T5
NIVELL IP

Xarxes de Computadors i Aplicacions

PAU ARTIGAS, DAVID CARRERA i JORDI TORRES
Departament d'Arquitectura de Computadors
UPC, setembre - 2009

Contenido

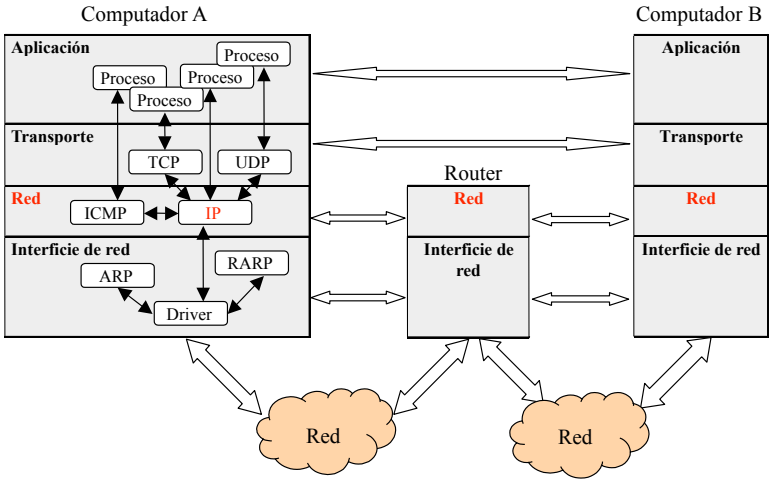
1. Modelo de operación y diseño
2. Formato datagrama IP
3. Direcciones IP mascaras y subredes
   - Classfull adresses
   - Subnetting
   - Classless addresses: CIDR
   - Interfície loopback
4. Funcionalidad de un router
5. Routing (encaminamiento)
   - Tabla de un router
   - Qué hace un router
   - Path determination y concepto de convergencia
   - Encaminamiento estático y dinámico
   - comando netstat y route
6. Resolución de direcciones (ARP)
7. Detección de errores en IP
8. Fragmentación y reensamblado
9. Internet Control Message Protocol
   - Herramientas de ayuda: ping, traceroute

transparències basades en el material docent dels professors José M. Barceló i Jordi Torres de l'assignatura STD del pla 91 de FIB.

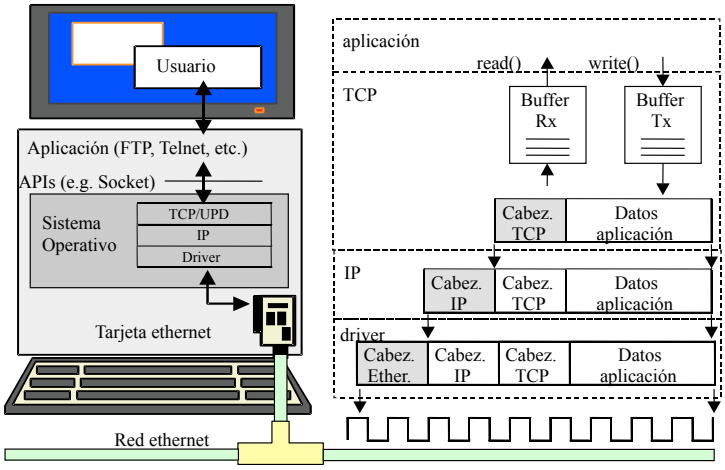
Contenido

10. Path determination
11. Sistemas Autónomos y protocolos de encaminamiento Externo
12. Detalles:
    - Modo promiscuo
    - Interficie loopback
    - MTU Path discovery
    - Ping
    - TTL
    - Traceroute
    - Cache ARP
13. NAT

Repàs: Arquitectura TCP/IP

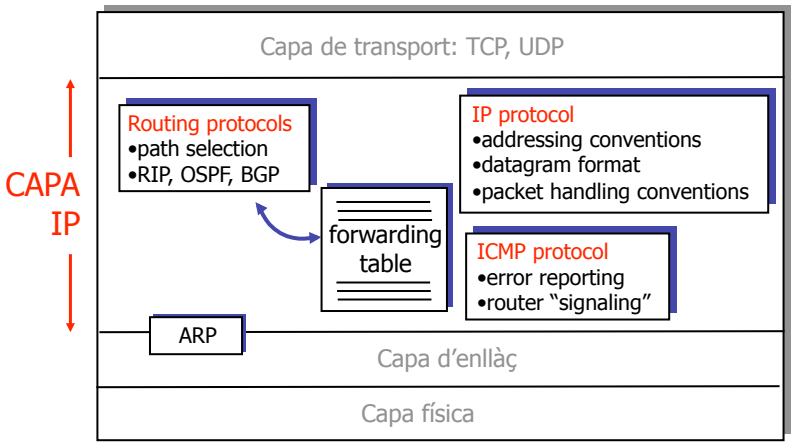


Repàs: Arquitectura TCP/IP



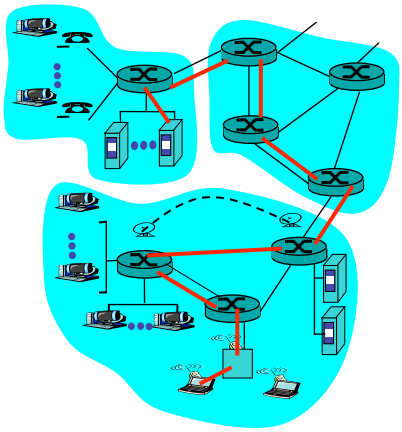
IP: La capa de xarxa d'internet

¿Què estudiarem en aquest tema?



1. Modelo de operación y diseño

- La aplicación remitente prepara datos
  - los pasa a su soft. TCP/IP.
  - se envía al router
  - El datagrama circula por routers
  - último router lo entrega
- La diferencia entre un router y un host: el host NO REENVIA



Modelo de operación

- La aplicación remitente prepara datos para su interlocutora y los pasa a su soft. TCP/IP.
- El datagrama se envía al router de la red (el destino está en otra red)
- El router decide a qué red envía el datagrama El datagrama circula por routers conectados a varias redes que van acercándolo a su destino hasta que llega al router de la red destinataria
- Este último router lo entrega a la máquina destinataria

Modelo de operación

- Un router es una máquina especializada en recibir datagramas y REENVIARLOS por una de las varias redes a las que puede estar conectado para acercarlos a su destino. Cualquier máquina podría ser configurada como router (activando una opción que permita reenviar datagramas -si el kernel lo permite-).
- **La diferencia entre un router y un host estriba en que el host NO REENVIA datagramas**. Si le llega alguno del que no es destinatario, lo ignora.

Modelo de operación

- No fiable y No orientado a conexión.
  - Ventajas:
    - **Flexibilidad**. IP requiere bien poco de las redes que interconecta.
    - **Robustez**: cada datagrama puede seguir una ruta diferente. Reacción a caídas en redes.
    - **Soporte** para aplicaciones no orientadas a conexión. Un servicio orientado a conexión hubiera hecho el conjunto más pesado.
- RFC 791. Septiembre de 1981.

Elementos de diseño

- Protocolos de nivel de red
  - **Routed Protocols** (protocolos encaminados):
    - Encapsulan información de nivel 4 (transporte)
    - Definen un esquema de direcciones jerarquizado
    - Usan un protocolo de nivel de enlace para transmitir la información a un dispositivo de nivel 3 (router)
    - E.g: IP, IPX, ...
  - **Routing Protocols** (protocolos de encaminamiento):
    - Buscan rutas óptimas para que los protocolos encaminados sepan a donde dirigir la información
    - E.g: RIP, IGRP, OSPF, EIGRP, BGP, ...
  - **Otros** (en pila TCP/IP)
    - ARP/RARP: mapeo de direcciones IP y MAC y viceversa
    - ICMP: control de mensajes de IP

Elementos de diseño

- Problemas que IP resuelve:
  - Encapsulamiento.
  - Direccionamiento.
  - Encaminamiento.
  - Fragmentación y reensamblado.
  - Vida de datagrama.
  - Control de error.
  - Control de flujo.

Elementos de diseño (explicación)

- Problemas que IP resuelve:
  - Encapsulamiento. Define un formato de cabecera de datagrama IP que encapsula la información que le ceda el protocolo que invoque su servicio.
  - Direccionamiento. Define un formato de dirección IP que identifica de forma no ambigua a cada interfaz de red de un host conectado a una de las redes (en un host conectado a varias redes por varios interfaces, cada uno de ellos tiene una dirección IP diferente) .
  - Encaminamiento. Define mecanismos (tablas de encaminamiento) para dirigir al datagrama en tránsito hacia su destino final.

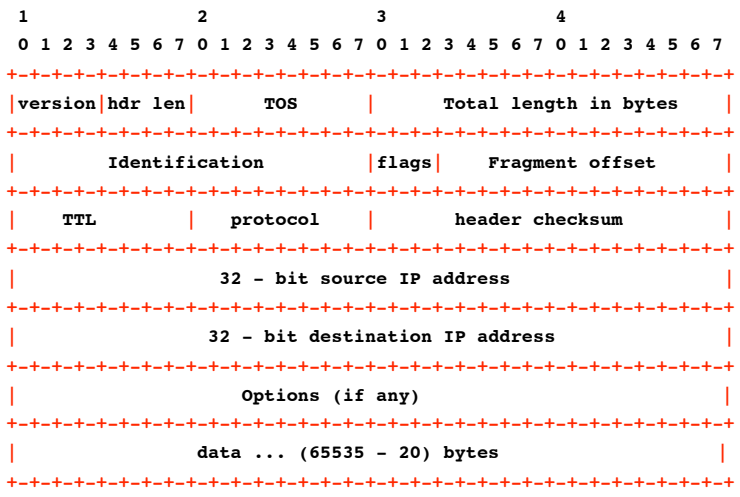
Elementos de diseño (explicación)

  - Fragmentación y reensamblado. Decide quién, cuándo y dónde se fragmentan y reensamblan los datagramas en su tránsito por las redes.
  - Vida de datagrama. Define un mecanismo que impide la existencia de datagramas que vaguen indefinidamente por las redes sin llegar nunca a sus destinos. (IMPORTANT!)
  - Control de error. Define un mecanismo de notificación de error (mediante el protocolo Internet Control Message Protocol) que sin embargo NO asegura al 100 % dicha notificación.
  - Control de flujo. Define un servicio limitado de control de flujo (también mediante ICMP)

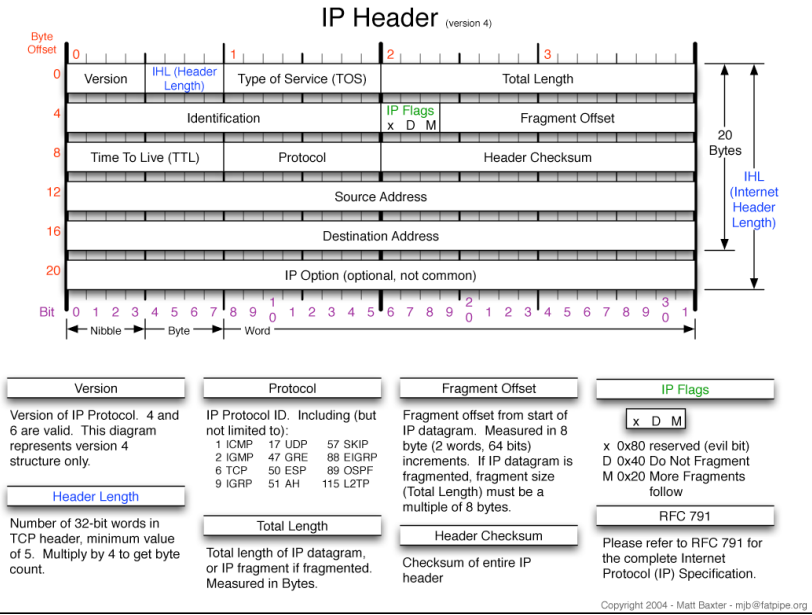
Elementos de diseño

- Unidad de información: datagrama IP
- conectionless Protocol
  - IP no mantiene estado entre sucesivos datagramas
  - No fases de establecimiento – mantemiento – cierre de la conexión
  - Datagramas IP pueden ser entregados sin un orden determinado
  - Control de errores: si algo va mal IP descarta los datagramas y envía un aviso al origen a través del protocolo ICMP
- IP es totalmente independiente de la tecnología de red

2. Formato datagrama IP



Formato datagrama IP



Formato datagrama IP

- **Version**: 4 (IPv4), 6 (IPv6)
- **Header Length**: en palabras de 32 bits (límite 60 bytes, 20), valor normal = 5
- **TOS** (Type of Service): 8 bits usados para un cierto grado de calidad de servicio.
- **Total length**: max is 65535 bytes, pero típico MTU (Maximum Transfer Unit) es 576 bytes de datagrama IP (viene de X.25) (incluye la cabezera)

Formato datagrama IP

- **Identification**: incrementado en uno por cada datag. enviado. Valor asignado por el generador. Junto con el offset se utiliza para ensamblar los posibles fragmentos que pueda sufrir en su tránsito hacia el destino.
- **Flags (3 bits):** control de fragmentación.
  - Un bit para marcar como fragmentable o no fragmentable.
  - Otro para indicar si el datagrama es el último fragmento o no.
- **Fragment offset**: Indica la parte del datagrama inicial en la que el fragmento está. Se indica en unidades de 8 octetos. Primer fragmento 0.

Formato datagrama IP

- **TTL** (Time To Live): Máximo tiempo que se permite al datagrama permanecer en el sistema internet. Se mide en saltos. Si llega a 0, el router que lo reciba lo destruye. Cada router que recibe el datagrama lo decrementa al menos en una unidad.
- **Protocol**: contenido del datag. IP, e.g datag. IP (0), mensajes ICMP (1), seg. TCP (6), datag. UDP (17), datag. IPv6 (41) ...
- **Header Checksum**: detector de errores. "Complemento a 1 de la suma complemento a 1 de todas las palabras de 16 bits de la cabecera. Para calcular el CRC, el valor del campo de checksum es 0". Recalculado en cada router que atraviesa (se cambia TTL y puede fragmentarse)."

Formato datagrama IP

- **Source/Destination IP addresses**
- **Options** (máximo 40 bytes). Lista variable de información opcional para el datagrama.
- **Data**: contenido del nivel superior u otros
  - Segmentos TCP, datagramas UDP, mensajes de otros protocolos de transporte como RSVP, mensajes ICMP, datagramas IP (tunneling) ...
  - Cada protocolo es identificado por el campo “protocolo”, e.g. TCP = 6

3. Direcciones IP

@IP = xxxx xxxx.xxxx xxxx. xxxx aaaa. aaaa aaaa

Mask = 1111 1111. 1111 1111. 1111 0000. 0000 0000

Donde **“xxxx”** identifica el **NetID**

y **“aaaa”** identifica el **HostID**

@IP = 197.35.187.138,

Mask= 255.255. 255.0

también se expresa como

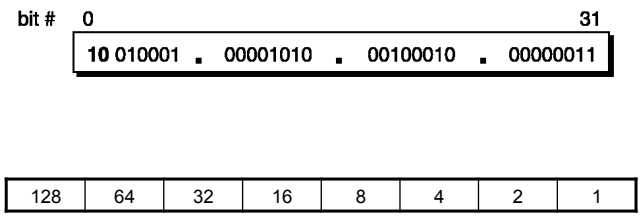
@IP = 197.35.187.138 /24

Direcciones IP

- Son de 32 bits divididos en 4 octetos expresados en decimal acompañados de una máscara de red
  - Contienen una parte que identifica la red (o subred) a la que pertenecen (NetID) y una parte que identifica la máquina (hostID)
  - La máscara identifica la longitud del prefijo de red
  - La máscara permite identificar el NetID (1s) del HostID (0s)

- Com s'expressa en binari aquesta adreça IP i la seva màscara?

145.10.34.3 /16

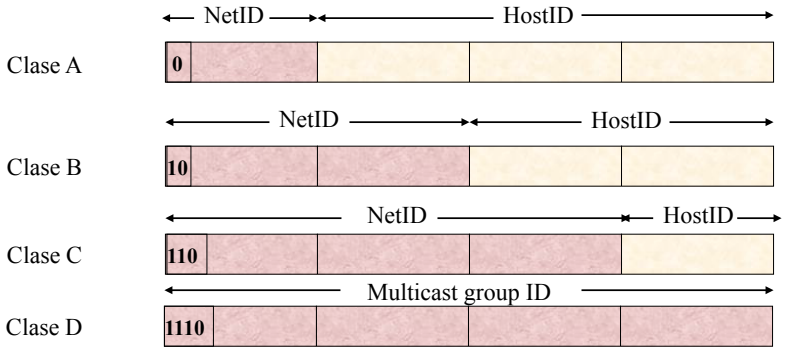


Direcciones IP

- Las direcciones IP son únicas.
- El organismo encargado de su asignación es IANA (Internet Assigned Numbers Authority), dependiente de ICANN .
- En Europa el organismo encargado de la asignación de direcciones IP es RIPE (Reseaux IP Europeenes).
- RIPE asigna grupos de direcciones IP a las Operadoras (Telefónica, BT, etc..) que a su vez las asignan a los ISP.

Direcciones IP

- **Classfull addresses**: son aquellas direcciones IP que definen una clase de tipo A (8 bits de Mask), tipo B (16 bits de Mask) y C (24 bits de Mask) y clase D para multicast



Direcciones IP

- **Direcciones con un significado especial**
  - **Dirección de red**: HostID = todo 0s, e.g. 147.83.0.0. Nunca se utilizan como dirección de dispositivos.
  - **Dirección broadcast**: HostID=todo 1s, e.g. 147.83.255.255
  - **Dirección 0.0.0.0** = este host en esta red (Nunca como dirección destino. Sólo sentido al arrancar el sistema como dirección origen), e.g. BOOTP
  - **Interficie loopback**: permite a un cliente comunicarse con un servidor dentro de la misma máquina sin tener que usar una tarjeta de red. Se usa la clase A 127.0.0.0 como dirección loopback
  - **Direcciones privadas**: son direcciones que no son enrutables en Internet, por tanto no son "vistas" por nadie fuera esa red
    - Clase A: 10.0.0.0
    - Clase B: 172.16.0.0 a 172.31.0.0
    - Clase C: 192.168.0.0 a 192.168.255.0

- Quantes xarxes de classe A es poden expressar en IPv4?
  - I quants hosts?
- Quin tant per cent aproximat representa el bloc d'adreces de la classe ***/8*** de l'espai de IPv4?
- I la classe ***/24***?

Direcciones IP

- Classfull addresses
  - **Clase A:** 0.0.0.0 a 127.255.255.255 con máscara 255.0.0.0
    - $2^{8-1} = 2^7 = 128$ redes
    - $2^{24} - 2 = 16.777.214$ hosts/red

    quan encara no havia crescut molt internet. grans empreses (e.g. GM)
  - **Clase B:** 128.0.0.0 a 191.255.255.255 con máscara 255.255.0.0
    - $2^{(16-2)} = 2^{14} = 16.384$redes
    - $2^{16} - 2 = 65.534$ hosts/red
  - **Clase C:** 192.0.0.0 a 223.255.255.255 con máscara 255.255.255.0
    - $2^{(24-3)} = 2^{21} = 2.097.152$ redes
    - $2^8 - 2 = 254$ host/red
  - **Clase D (Multicast):** 224.0.0.0 a 239.255.255.255
    - Hay direcciones especiales multicast definidas por IANA

- A quina classe corresponen les adreces IP següents?

145.32.59.24

200.42.129.16

14.82.19.54

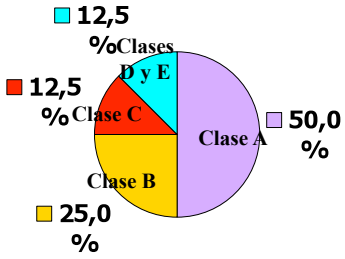
10010001.00100000.00111011.00011000

11001000.00101010.10000001.00010000

00001110.01010010.00010011.00110110

Agotamiento de direcciones?

- Crisis en el esquema de direcciones de IPv4



Clases A y B: suponen un 75 % de las direcciones pero sólo permiten 17000 organizaciones

Clases C permiten más organizaciones pero pocos hosts por red (254 hosts)

El hecho de que haya más clases A, B o C no mejora la situación (demasiadas direcciones en las tablas de routing)

Cada vez más gente y dispositivos conectados a Internet!

- SOLUCIÓN → IPv6

Agotamiento de direcciones?

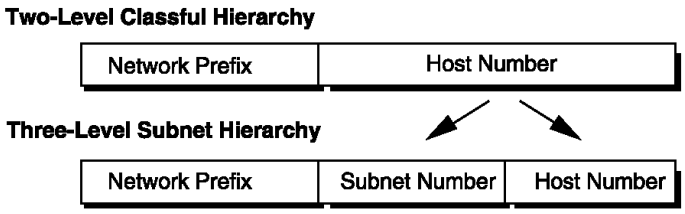
- Debido a la falta de direcciones IP es necesario encontrar soluciones
  - **Subnetting**: permite "dividir" redes de clase A o B en varias subredes B o C dentro de una "organización", mejora la distribución de direcciones pero no soluciona el problema a corto plazo
  - **VLSM** (Variable-Length Subnet Masks): significa "subnetting a subnet", es el mismo mecanismo que Subnetting, pero permitiendo subredes de tamaño diferente a B o C, sucede lo mismo que el punto anterior
  - **CIDR** (Classless Inter-Domain Routing): Eliminar el concepto de clases a nivel global y mejora la agregación de rutas para una mejor eficiencia del encaminamiento (CPU y memoria en los routers)
  - **@IP privadas + NAT (Network Address Translation):** usar direcciones privadas y efectuar una translación de direcciones privadas a públicas (solución provisional o a corto plazo)

Direcciones IP

– **Subnetting**

- Una red puede ser dividida en redes más pequeñas llamadas subredes
- Se da una clase con una mascara mayor que la de la clase

  "*se piden prestados bits de la clase para crear subredes"*



Direcciones IP

CIDR: Classless InterDomain Routing:
son aquellas direcciones que no mantienen el concepto de clase y por tanto tienen una máscara que puede ser cualquier número de bits

- Intentan resolver el problema de agotamiento de direcciones con clase que se considera que es un sistema que "desperdicia" direcciones
- E.g. 147.83.128.0 255.255.255.0 (147.83.128.0 /24) la máscara tiene 8+8+8 = 24 bits. Notar que es una clase B con máscara de clase C, (aunque aquí el concepto de clase deja de existir)
- Las máscaras no tienen porqué ser multiplos de 8 bits: 147.83.128.0 /22 (máscara:255.255.252.0)

CIDR

Classless Inter-Domain Routing (CIDR) is a replacement for the old process of assigning Class A, B and C addresses with a generalized network "prefix". Instead of being limited to network identifiers (or "prefixes") of 8, 16 or 24 bits, CIDR currently uses prefixes anywhere from 13 to 27 bits. Thus, blocks of addresses can be assigned to networks as small as 32 hosts or to those with over 500,000 hosts. This allows for address assignments that much more closely fit an organization's specific needs.

Direcciones IP

- IANA da una dirección con una mascara y permite que los administradores de red gestionen esta dirección de red creando tantas subredes como quieran, es decir, asignando nuevas mascaras de mayor valor a la otorgada (subnetting o lo que se llama VLSM: Variable Length Subnetting Mask)

- CIDR permite una assignación eficiente del espacio de direcciones IPv4.

Direcciones IP

- En el caso de usar dir. IP de clase A o B
  - El número de hosts en la red es grande.
- Imaginemos la compañía X que dispone de una clase A, pero sólo tiene 300 hosts en la compañía.
  - ¿Qué pasa con el resto de direcciones?
- Subneting permite a los administradores de red dividir una clase grande en varias subredes más pequeñas.

Direcciones IP

- **Exemple Subnetting (classfull)**

  Suposem que volem 8 subclasses de 132.45.0.0 /16.
  - ¿Cuantos bits requerimos para la NETID?
  - ¿Cual es la máscara?
  - ¿Cuales son las subredes?
  - ¿Cuales son las direcciones de host de la subred 132.45.96.0/19?
  - ¿Cual es la dirección de broadcast?

Direcciones IP

Exemple Subnetting (cont)

2^3=8 xarxes

/19 o 255.255.224.0

Subnet #0: 10000100.00101101. **000** 00000.00000000 = 132.45.0.0/19
Subnet #1: 10000100.00101101. **001** 00000.00000000 = 132.45.32.0/19
Subnet #2: 10000100.00101101. **010** 00000.00000000 = 132.45.64.0/19
Subnet #3: 10000100.00101101. **011** 00000.00000000 = 132.45.96.0/19
Subnet #4: 10000100.00101101. **100** 00000.00000000 = 132.45.128.0/19
Subnet #5: 10000100.00101101. **101** 00000.00000000 = 132.45.160.0/19
Subnet #6: 10000100.00101101. **110** 00000.00000000 = 132.45.192.0/19
Subnet #7: 10000100.00101101. **111** 00000.00000000 = 132.45.224.0/19

Direcciones IP

Exemple Subnetting (cont)

Subnet #3: 10000100.00101101.011 00000.00000000 = 132.45.96.0/19

Host #1: 10000100.00101101.011 **00000.00000001** = 132.45.96.1/19
Host #2: 10000100.00101101.011 **00000.00000010** = 132.45.96.2/19
Host #3: 10000100.00101101.011 **00000.00000011** = 132.45.96.3/19
:
Host #8190: 10000100.00101101.011 **11111.11111110** = 132.45.127.254/19

broadcast address =
10000100.00101101.011 **11111.11111111** = 132.45.127.255/19

Direcciones IP

– **Ejemplo Subnetting (CIDR)**

- IANA da red 132.45.0.0 /20. Calcular las subredes que se pueden crear

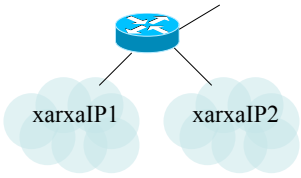
1001 0011. 0100 0011. 0000 0000. 0000 0000

1111 1111. 1111 1111. 1111 0000. 0000 0000

Clase B: 16 bits de máscara y nos dan 20 bits de máscara, luego hemos robado 4 bits para subnetting

Direcciones IP

- Imaginem una empresa amb 35 i 42 hosts. (suposant un /24)
  - **Quantes adreces IP desaprofitem?**
  - **quantes en podem recuperar amb subneting?**



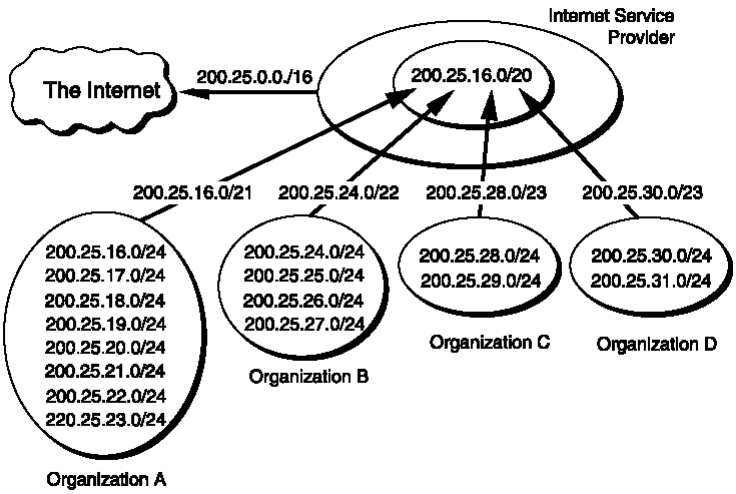
Direcciones IP

- **Agregación**
  - CIDR permite controlar el tamaño de las tablas de routing

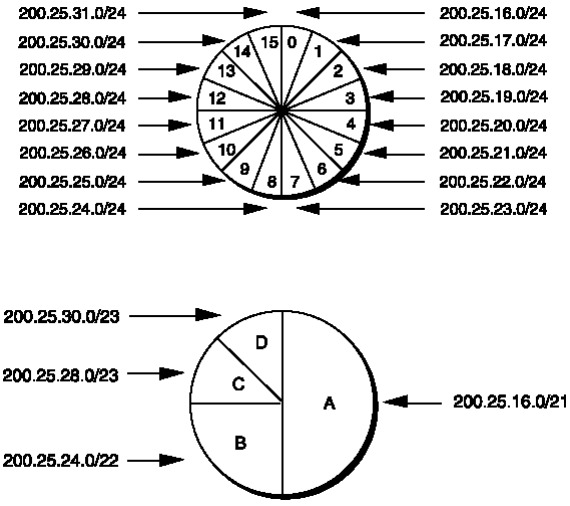
  p.ej: un ISP que dispone del bloque de direcciones 200.25.0.0/16 y quiere repartir la 200.25.16.0/20 entre 4 organizaciones (ver esquema)

  en un entorno Classful el ISP estaría obligado a usar la /20 como 16 /24 individuales

Direcciones IP



Direcciones IP



Direcciones IP

- Agregar

  212.56.132.0/24
  212.56.133.0/24
  212.56.134.0/24
  212.56.135.0/24

- 212.56.132.0/24 11010100.00111000.10000**00**.00000000
- 212.56.133.0/24 11010100.00111000.10000**01**.00000000
- 212.56.134.0/24 11010100.00111000.10000**10**.00000000
- 212.56.135.0/24 11010100.00111000.10000**11**.00000000
- Common Prefix: 11010100.00111000.**10000**100.00000000

→ The CIDR aggregation is: 212.56.132.0/22

Direcciones IP

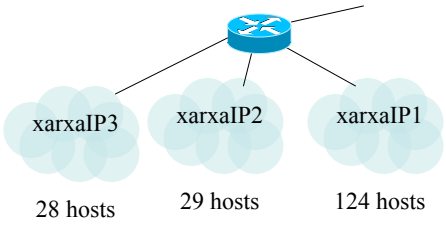
- Agregar

212.56.146.0/24
212.56.147.0/24
212.56.148.0/24
212.56.149.0/24

ull!

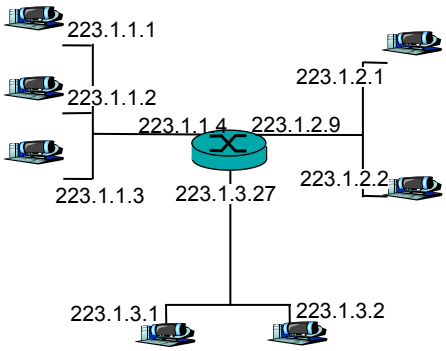
Direcciones IP

- Ajustar al màxim l'adreça 152.5.5.0/24 que ens han donat per



4. Funcionalidad de un router

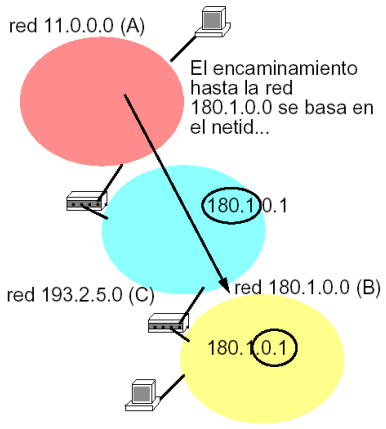
- Intermediario entre dos redes de nivel 2
  - Cualquier host A (@IP_A)
  - que quiera enviar un datagrama IP a otro host B (@IP_B)
  - que esté en una subred distinta (@NetIDA ≠ @NetIDB)
  - **debe hacerlo a través de un router**



Funcionalidad de un router

- El router deberá entre otras cosas realizar las siguientes funciones
  - "**Forwarding**": envíar datagramas de una subred a otra
  - "**Routing**": decidir a que subred debe enviar un datagrama que le llegue de otra subred (decidir interficies de salida del router)
  - "**Error messaging**": notificar al host origen con un mensaje ICMP de cualquier problema que le impida realizar un "forwarding" (descarta el datagrama pero informa al origen)

Funcionalidad de un router



Algunas direcciones tienen significados especiales.

0.0.0.0 = Este host (No dirección destino. Solo al arrancar el sistema)

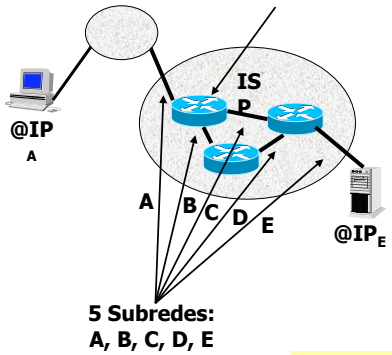
255.255.255.255 : Broadcast local (no dir. fuente)

netid, hostid=111...11 : Broadcast en la red indicada

127.0.0.1 : Loopback

Funcionalidad de un router

- Ejemplo:



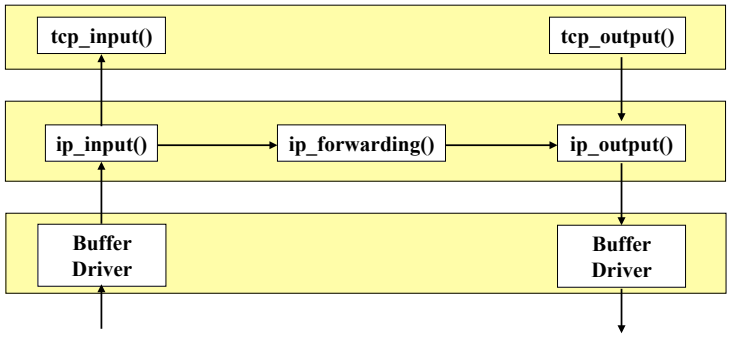
- **Routing: decidir la interficie de salida del router para cada datagrama que le llega**
- **Forwarding: usar la tecnología de nivel 2 para transmitir datagramas por una interficie de salida**
- **Error messaging: notificar cualquier problema que impida el forwarding de datagramas usando mensajes ICMP**

Cada router tiene una @IP con NetID distinto por interficie de salida

cada interficie es una red

5. Routing

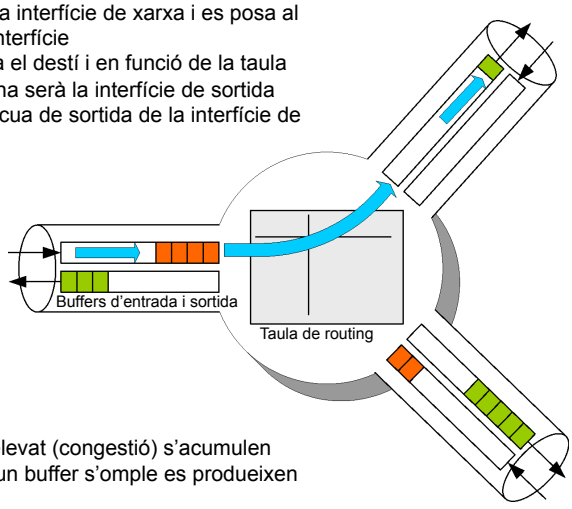
- Funcionalidad de un router
  - Si un **host** recibe un datagrama que no está dirigido a él, el host descarta el datagrama
  - Si un **router** recibe un datagrama que no está dirigido a él, intenta encaminarlo a un host o a otro router → FORWARDING



Routing

- Com és un router?

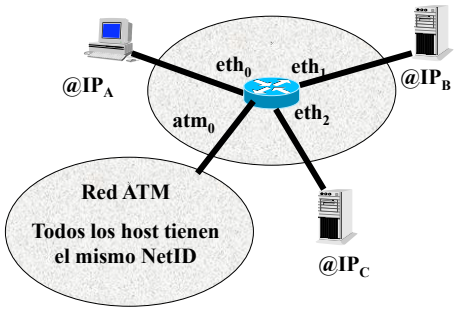
1. El paquet arriba per una interfície de xarxa i es posa al buffer d'entrada de la interfície
2. El router el llegeix, mira el destí i en funció de la taula de routing decideix quina serà la interfície de sortida
3. El paquet es posa a la cua de sortida de la interfície de xarxa escollida



- Quan el tràfic és molt elevat (congestió) s'acumulen paquets als buffers; si un buffer s'omple es produeixen pèrdues

Routing

- Tabla de un **router**
  - Cada router mantiene una tabla de encaminamiento que indica como llegar a un destino (dirección IP e interficie)
  - En vez de una dirección IP puede haber prefijos de red (NetID) que representen a redes enteras
  - La tabla es rellenada por un algoritmo de encaminamiento o manualmente



Ejemplo de Routing table

IP address	Interface
@IP$_A$	eth$_0$
@IP$_B$	eth$_1$
@IP$_C$	eth$_2$
NetID	atm0

Routing

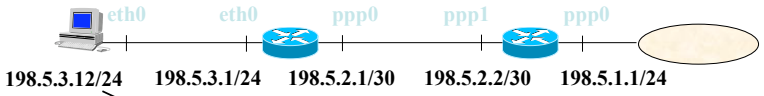
- ¿Qué hace un router cuando recibe un datagrama?
  - Extraer la @IPdest del datagrama recibido
  - Accede a la tabla de encaminamiento donde va realizando una operación de busqueda (matching)
    - Si *@IPdest AND Mask_tabla = IP destination_tabla* entonces encaminar por interficie indicada por la tabla
    - última entrada en la tabla: router por defecto
    - Sino coincide con ninguna entrada entonces enviar ICMP con error *network unreachable*.

Routing

- **Default Router (Router por defecto):** router dentro de una red a la que se le envían aquellos datagramas que un host u otro router no saben donde encaminar. El router por defecto tienen más información que le permite encaminar
- **Longest Prefix Match**: proceso por el cual debemos encontrar la entrada en la que coincide el prefijo más largo

Routing

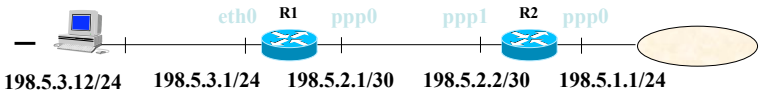
- Tabla de un **host**
  - Debe indicar como llegar a su propia subred (haciendo una ARP)
  - Debe indicar como salir de su subred (usando el gateway por defecto)



@IPdestino	Máscara	Gateway	Interficie
198.5.3.0	255.255.255.0	0.0.0.0	eth0
0.0.0.0	0.0.0.0	198.5.3.1	eth0

IP Routing

- Tabla de un **router**
  - Si existe una “correspondencia explicita” a una entrada en la tabla, el router sabe por donde sacar la trama
  - Si no existe una correspondencia explicita, normalmente habrá una salida por defecto (**gateway** o **router por defecto**)
  - Por consiguiente nos falta en la tabla información de a que destino queremos enviar el paquete y porqué interficie sacarla

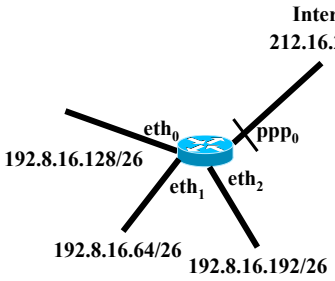


R1

@IPdestino	Máscara	Gateway	Interficie
198.5.3.12	255.255.255.255	0.0.0.0	eth0
198.5.3.0	255.255.255.0	0.0.0.0	eth0
0.0.0.0	0.0.0.0	198.5.2.2	ppp0

IP Routing

- Ejemplo de cómo se las arregla el router:



Routing table

IP address	Mask	Gateway	Interface
192.8.16.192	255.255.255.192	0.0.0.0	eth2
192.8.16.64	255.255.255.192	0.0.0.0	eth1
192.8.16.128	255.255.255.192	0.0.0.0	eth0
0.0.0.0	0.0.0.0	212.16.20.2	ppp0

- Supongamos que le llegan los siguientes paquetes:

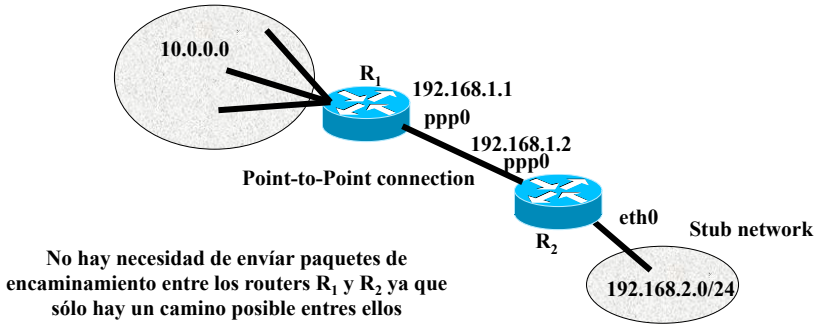
@IPorg	@IPdst
6.6.6.1	192.8.16.195
6.6.6.1	192.8.16.67
192.8.16.67	6.6.6.1 /*→ quiero salir */

IP Routing

- Ejemplo de encaminamiento estático

Routing table of R_1			
192.168.2.0	/24	192.168.1.2	ppp0
192.168.1.0	/24	0.0.0.0	ppp0
10.0.0.0	.....		

Routing table of R_2			
192.168.2.0	/24	0.0.0.0	eth0
192.168.1.0	/24	0.0.0.0	ppp0
0.0.0.0	0.0.0.0	192.168.1.1	ppp0



No hay necesidad de envíar paquetes de encaminamiento entre los routers R_1 y R_2 ya que sólo hay un camino posible entres ellos

IP Routing

- Ejemplo de encaminamiento estático: comando route (modo root)

route [-v] [-A family**]** add **[-net|-host]** target **[netmask** Nm**] [gw** Gw**] [metric** N**] [mss** M**] [window** W**] [irtt** I**]**
[reject**] [**mod**] [**dyn**] [**reinstate**] [[**dev**]** If**]**

route [**-v**] [**-A** family] del [**-net**| **-host**] target [**gw** Gw]
[**netmask** Nm] [**metric** N] [[dev] If]

  - Ejemplo: ver la tabla de routing

•route -v

Kernel IP routing table

Destination	**Gateway**	**Genmask**	**Flags**	**Metric**	**Ref**	**Use**	**Iface**
147.83.35.0	*	255.255.255.0	U	0	0	0	eth0
loopback	*	255.0.0.0	U	0	0	0	lo
0.0.0.0	arenys5.ac.upc.	0.0.0.0	UG	0	0	0	eth0

IP Routing

- Comando “netstat” (modo usuario)
  - Permite observar entre otras cosas la tabla de encaminamiento además del estado de las conexiones, estadísticas, ...

    netstat { **--route**|**-r**} [address_family_options]
    [**--extend**| **-e**[**--extend**| **-e**]] [**--verbose**| **-v**] [**--numeric**| **-n**]
    [**--continuous**| **-c**]
  - Ejemplo: ver la tabla de routing

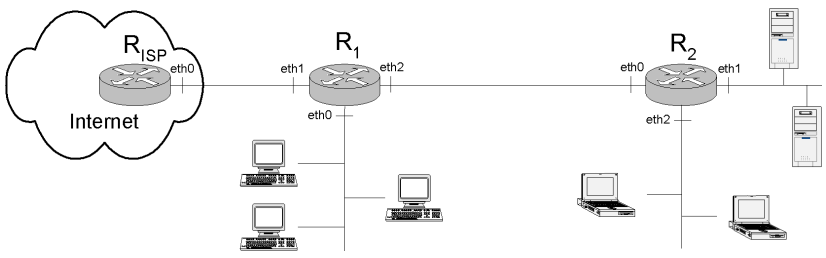
    • **netstat -rn**

    Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
147.83.35.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
loopback	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	arenys5.ac.upc.es	0.0.0.0	UG	0	0	0	eth0

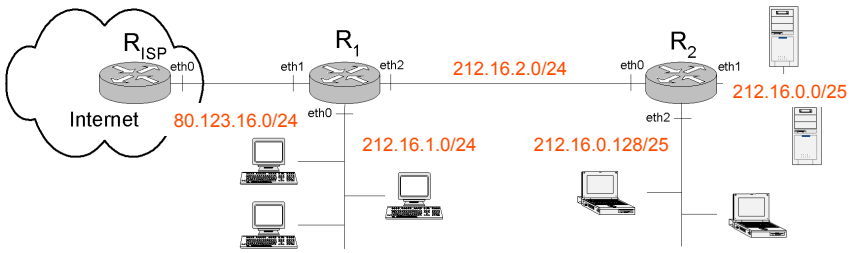
Exemple Routing: Presentació

- Exemple d’una xarxa d’una empresa
  - 2 Routers
  - 3 xarxes: PCs, portàtils i servidors
    - Fals! en té més, ja ho veurem
  - Connexió a internet a través de l’ISP



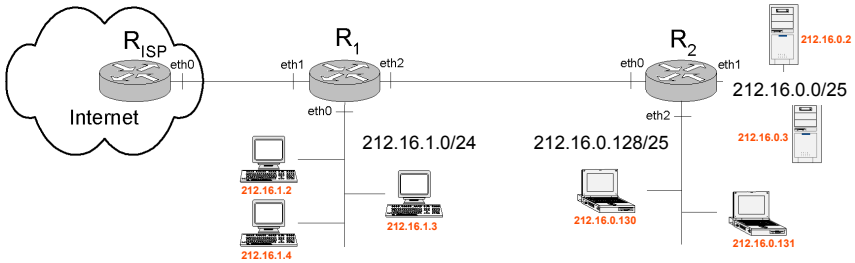
Exemple Routing: Xarxes

- Un router separa 2 o més xarxes
  - Tenim 5 xarxes
    - PCs, portàtils i servidors
    - Entre R_1 i R_2!
    - Entre R_1 i R_{ISP}!



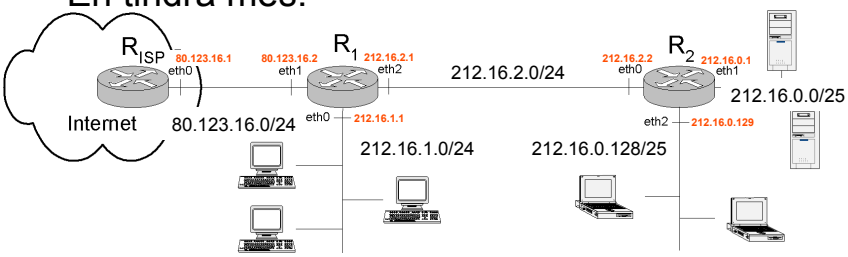
Exemple Routing: IPs

- Els PCs tenen IPs entre la 212.16.1.1 i la 212.16.1.254
  - 212.16.1.0 és la de xarxa i 212.16.1.255 la de broadcast
- Els servidors tenen IPs entre la 212.16.0.1 i la 212.16.0.126
  - 212.16.0.0 és la de xarxa i 212.16.0.127 la de broadcast
- Els portàtils tenen IPs entre la 212.16.0.129 i la 212.16.0.254
  - 212.16.0.128 és la de xarxa i 212.16.0.255 la de broadcast



Exemple Routing: IPs

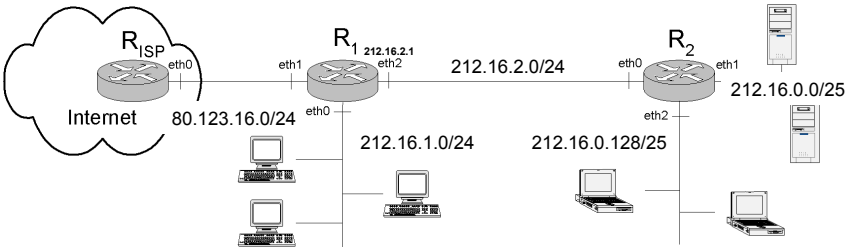
- R_1 i R_2 tenen 3 IPs cadascun:
  - Una per cada interfície de xarxa
  - A cada interfície una IP de la xarxa a la qual estan connectats
- R_{ISP} té una IP de la xarxa que fa servir per donar connexió a l'empresa
  - En tindrà més!



Exemple Routing: R_2

- Taula de routing de R_2
  - Entrades ordenades per la mida de la màscara

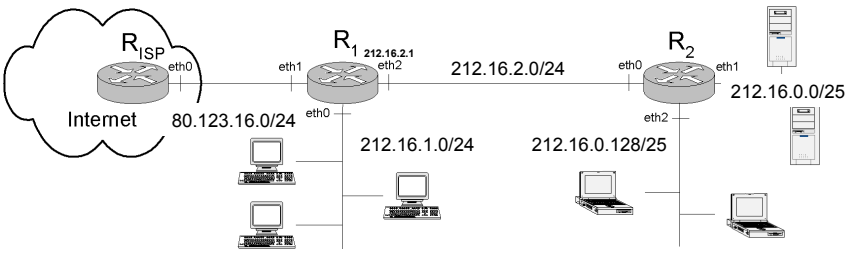
Direcció	Màscara	GW	Interfície	
212.16.0.0	255.255.255.128		eth1	Xarxes a les que està connectat directament
212.16.0.128	255.255.255.128		eth2	
212.16.2.0	255.255.255.0		eth0	
212.16.1.0	255.255.255.0	212.16.2.1	eth0	L'altre xarxa de l'empresa
0.0.0.0	0.0.0.0	212.16.2.1	eth0	Default gateway



Exemple Routing: R_2

- Taula de routing de R_2 millorada
  - L'entrada per l'altre xarxa de l'empresa (a la que no està connectat directament) no és necessària, amb el default gateway ja és suficient
    - L'altre taula no és incorrecta, només una mica redundant

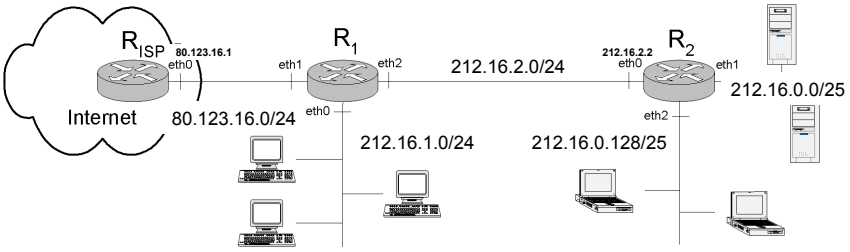
Direcció	Màscara	GW	Interfície	
212.16.0.0	255.255.255.128		eth1	Xarxes a les que està connectat directament
212.16.0.128	255.255.255.128		eth2	
212.16.2.0	255.255.255.0		eth0	
0.0.0.0	0.0.0.0	212.16.2.1	eth0	Default gateway



Exemple Routing: R_1

- Taula de routing de R_1
  - Entrades ordenades per la mida de la màscara

Direcció	Màscara	GW	Interfície	
212.16.0.0	255.255.255.128	212.16.2.2	eth2	Xarxes de l'empresa a les que NO està connectat directament
212.16.0.128	255.255.255.128	212.16.2.2	eth2	
212.16.2.0	255.255.255.0		eth2	Xarxes a les que està connectat directament
212.16.1.0	255.255.255.0		eth0	
80.123.16.0	255.255.255.0		eth1	
0.0.0.0	0.0.0.0	80.123.16.1	eth1	Default gateway



Exemple Routing: R_1

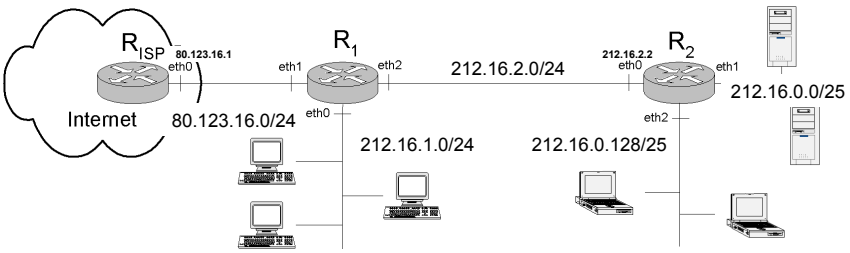
- Taula de routing de R_1 millorada
  - Es pot fer agregació!

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.0	212.16.2.2	eth2
212.16.2.0	255.255.255.0		eth2
212.16.1.0	255.255.255.0		eth0
80.123.16.0	255.255.255.0		eth1
0.0.0.0	0.0.0.0	80.123.16.1	eth1

Xarxes de l'empresa a les que NO està connectat directament

Xarxes a les que està connectat directament

Default gateway



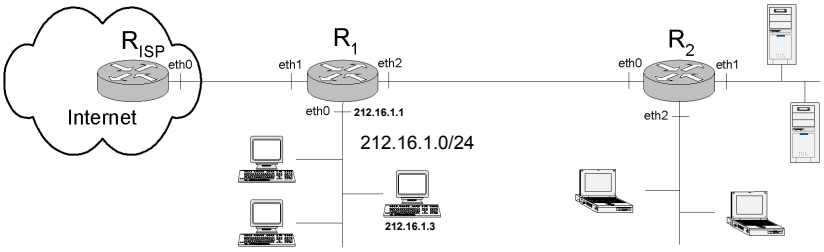
Exemple Routing: Hosts

- Per exemple el 212.16.1.3
  - Les altres seguiran el mateix esquema
    - Xarxa a la que estan connectats
    - Default gateway

Direcció	Màscara	GW	Interfície
212.16.1.0	255.255.255.0		eth0
0.0.0.0	0.0.0.0	212.16.1.1	eth0

Xarxa a la que està connectat directament

Default gateway



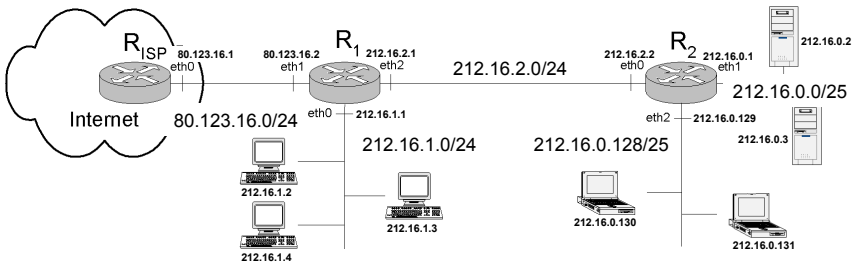
Exemple Routing: Tot junt

R_1

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.0	212.16.2.2	eth2
212.16.2.0	255.255.255.0		eth2
212.16.1.0	255.255.255.0		eth0
80.123.16.0	255.255.255.0		eth1
0.0.0.0	0.0.0.0	80.123.16.1	eth1

R_2

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.128		eth1
212.16.0.128	255.255.255.128		eth2
212.16.2.0	255.255.255.0		eth0
0.0.0.0	0.0.0.0	212.16.2.1	eth0



Exemple Routing: Exemple 1

212.16.1.3

⇩

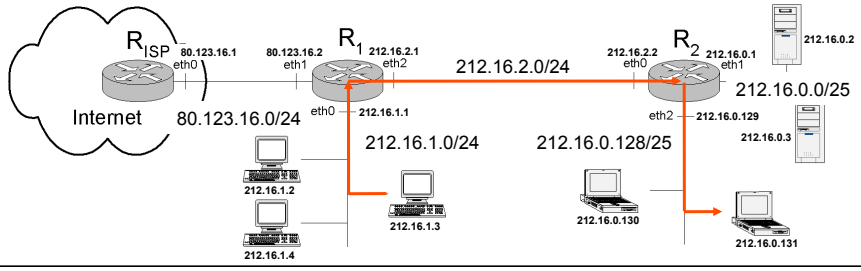
212.16.0.131

R_1

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.0	212.16.2.2	eth2
212.16.2.0	255.255.255.0		eth2
212.16.1.0	255.255.255.0		eth0
80.123.16.0	255.255.255.0		eth1
0.0.0.0	0.0.0.0	80.123.16.1	eth1

R_2

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.128		eth1
212.16.0.128	255.255.255.128		eth2
212.16.2.0	255.255.255.0		eth0
0.0.0.0	0.0.0.0	212.16.2.1	eth0



Exemple Routing: Exemple 2

212.16.1.3

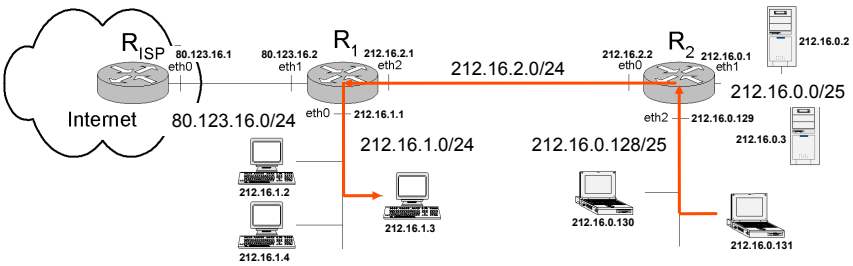
212.16.0.131

R_1

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.0	212.16.2.2	eth2
212.16.2.0	255.255.255.0		eth2
212.16.1.0	255.255.255.0		eth0
80.123.16.0	255.255.255.0		eth1
0.0.0.0	0.0.0.0	80.123.16.1	eth1

R_2

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.128		eth1
212.16.0.128	255.255.255.128		eth2
212.16.2.0	255.255.255.0		eth0
0.0.0.0	0.0.0.0	212.16.2.1	eth0



Exemple Routing: Exemple 3

212.16.0.131

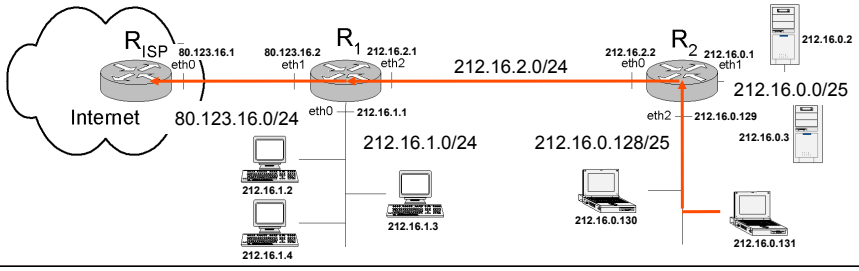
147.83.35.228

R_1

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.0	212.16.2.2	eth2
212.16.2.0	255.255.255.0		eth2
212.16.1.0	255.255.255.0		eth0
80.123.16.0	255.255.255.0		eth1
0.0.0.0	0.0.0.0	80.123.16.1	eth1

R_2

Direcció	Màscara	GW	Interfície
212.16.0.0	255.255.255.128		eth1
212.16.0.128	255.255.255.128		eth2
212.16.2.0	255.255.255.0		eth0
0.0.0.0	0.0.0.0	212.16.2.1	eth0



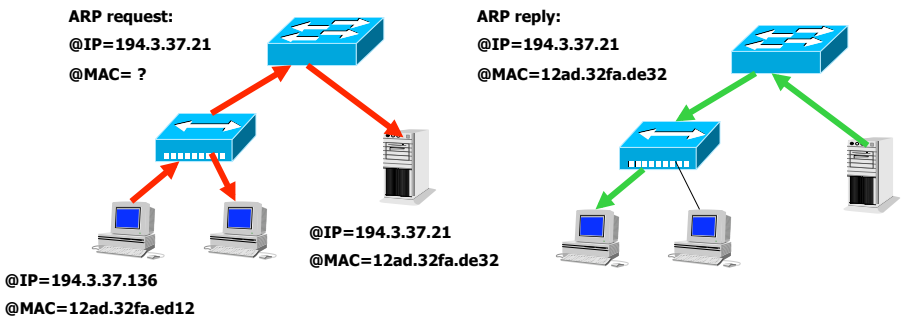
6. ARP (Address Resolution Protocol)

- 32-bit IP address:
  - network-layer address
  - used to get datagram to destination IP network (recall IP network definition)

- LAN (or MAC or physical or Ethernet) address:
  - used to get datagram from one interface to another physically-connected interface (same network)
  - 48 bit MAC address (for most LANs) burned in the adapter ROM
  - MAC address allocation administered by IEEE
  - manufacturer buys portion of MAC address space (to assure uniqueness)

ARP (Address Resolution Protocol)

- (RFC 826)
- Se encarga de mapear @IP con @MAC
- Host A envía una trama broadcast a la subred, el servidor responde con la respuesta



ARP (Address Resolution Protocol)

- **ARP cache**:
  - tabla que mantiene cada dispositivo con los mapeos más recientes entre @IP y @MAC
    - Recordar que las @IP son dinámicas y pueden cambiar, en cambio las @Físicas son permanentes a las tarjetas
  - La duración de una entrada es de 20 minutos (depende del SO)
  - En UNIX: se puede usar el comando arp –a para obtener el contenido de la tabla ARP

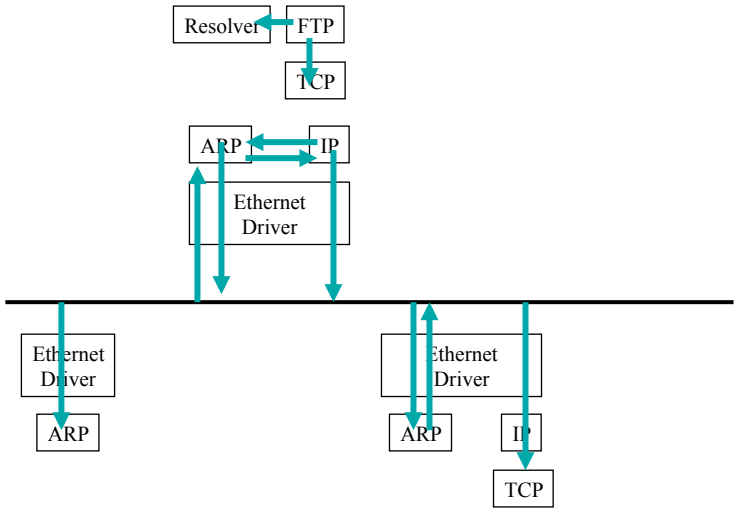
- **arp [-vn] [-H type] [-i if] -a [hostname]**

- **aucanada% arp –a**

teix.ac.upc.es (147.83.35.110) at 00:20:E1:10:4f:34 [ether] on eth0

arenys5.ac.upc.es (147.83.35.2) at 00:10:F8:B3:E4:00 [ether] on eth0

ARP (Address Resolution Protocol)

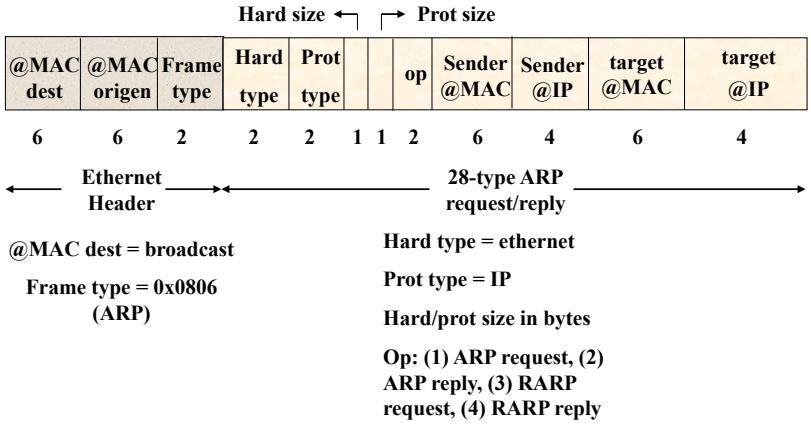


ARP (Address Resolution Protocol)

- FTP uses **gethostbyname** to determine the IP address of an FTP server
- FTP asks TCP to establish a connection
- TCP send a connection request to that IP address, which is on the local network
- The O/S uses ARP to determine the Ethernet MAC address
- The destination O/S replies & the reply is received
- The IP layer can now send the packet

ARP (Address Resolution Protocol)

- Formato del paquete ARP
  - Usa tramas de nivel 2 (e.g. Ethernet)



ARP (Address Resolution Protocol)

- ¿Qué ocurre si el host buscado no existe?
  - Se reintenta porqué TCP reintenta la conexión varias veces hasta que salte su TimeOut (ver tema 3WHS de TCP)
- Gratuitous ARP
  - Un host hace un ARP request para averiguar su propia @IP
  - Para qué?
    - Para saber si otro host tiene configurada la misma @IP
    - Hacer un update de ARP caches cuando cambias la @Física

7. Detección de errores en IP

- **Checksum** en la cabecera de cada datagrama IP
- Los datos están protegidos por otro checksum que se aplica en el nivel de transporte (sobre todo el segmento TCP o datagrama UDP)
- El checksum **SOLO protege la cabecera IP**
- Acción si error detectado: descarta la trama y envía un mensaje ICMP al origen
- Checksum (es software) = Σ Words i
  - Alinear en palabras de 16 bits
  - Inicializar el checksum a 0
  - Sumar palabras de la cabecera en complemento a 1s
  - Calcular el complemento a 1 del resultado
  - Rellenar el campo del checksum con el valor calculado
  - Se tiene que recalcular cada vez que se atraviesa un router (ya que hay campos de la cabecera que son mutables, e.g. TTL)

Detección de errores en IP

- Detección de errores en IP:
  - E.g. Checksum de una cabecera:
  - a% tcpdump –x –s 512 –i eth0

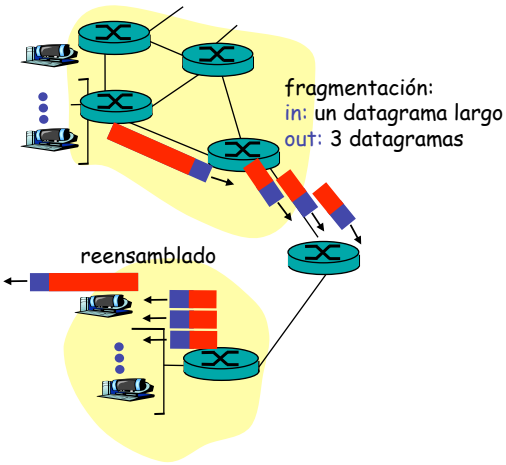
45 10
05 dc
64 78
00 00
40 06
00 00
93 53
23 50
93 53
23 51
2 5c b1 → acumular el 2 y hacer complemento a 1
5c b3 → checksum = a3 4c

Si añadimos el checksum, la suma dará 0

- **Version:** 4 (0x04)
- **Hdr Length:** 20 bytes (0x05)
- **TOS:** (0x10)
- **Total length:** 1500 bytes (0x05dc)
- **Ident:** (0x6478)
- **Flags:** 000 (3 bits)
- **offset:** 0 (13 bits)
- **TTL:** 64 (0x40)
- **Protocolo:** el 6 es TCP (0x06)
- Hdr checksum: (0x0000)
- **@IPorg:** 147.83.35.80 (0x9353 2350)
- **@IPdest:** 147.83.35.81 (0x9353 2351)

8. Fragmentación y reensamblado

- Los datagramas IP largos son fragmentados en la red y reensamblados al final del destino



Fragmentación y reensamblado

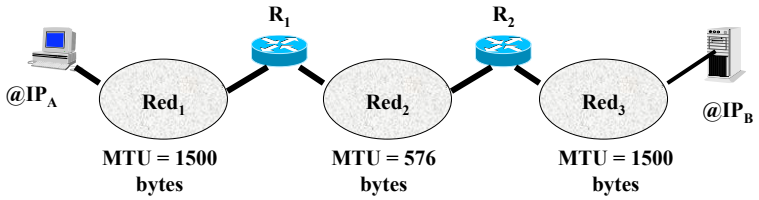
- MTU (Maximum Transfer Unit)
  - Número máximo de bytes de datos que pueden aparecer encapsulados en una trama de red
    - | CAB | ←MTU → |crc|
  - Cada red (Ethernet, ATM, X.25 ...) tiene su propia MTU
  - "**Path MTU**": se define como el **mínimo MTU** de entre todas las redes que hay entre dos hosts conectados a Internet

Network	MTU (Bytes)
Punto a Punto	296
X.25	576
PPP	1500
Ethernet	1500
IEEE 802.3/802.2	1492
FDDI	4352
IEEE 802.5 (4 Mbps TR)	4464
IBM (16 Mbps TR)	17914

Fragmentación y reensamblado

→ Cada día menos usado con el uso de MTU Path Discovery

- Ejemplo



→ R_1 fragmentará datagramas enviados por host A debido a que la MTU de la Red_2 es menor que la de la Red_1 (Path MTU = 576 bytes)

→ El datagrama no se reensambla en R_2, sino que lo hará el destino ($Host_B$)

→ R2 reenvía los fragmentos como si fuesen datagramas independientes (podrían llegar desordenados o que alguno de los fragmentos no llegase)

→ Usa los campos "flags", "fragment offset", "total length" de la cabecera IP para fragmentar y reensamblar

Fragmentación y reensamblado

- "**Flags**" del datagrama IP (campo de 3 bits)
  - El segundo y tercer bit se usan para fragmentar:
    - Flag M "more fragments": activo cuando se fragmenta excepto en el último fragmento que se desactiva
    - Flag D "don't fragment": si activo un router NO fragmentará el datagrama (devolverá un mensaje ICMP indicando que no puede enviar el datagrama ya que no se le permite fragmentar)
  - "**Fragment offset**": campo de 13 bits que indica el offset (en bytes) que tranporta este fragmento desde el origen del datagrama
  - "**Total length**" del fragmento contiene la longitud total del fragmento y NO la del datagrama original

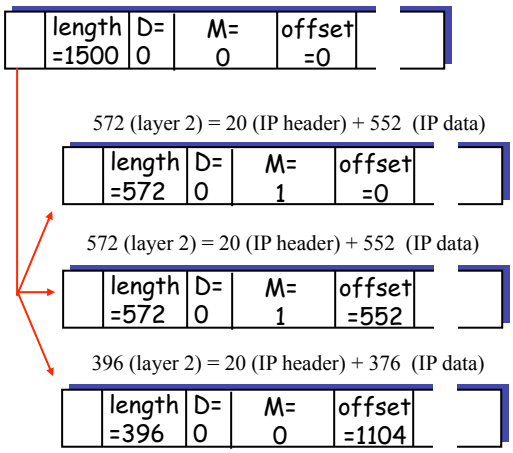
Fragmentación y reensamblado

Ejemplo:

- MTU=576, entonces busca el 1r multiplo de 8.
- Todos los fragmentos excepto el último deben ser **multiplos de 8 bytes** (en su campo de datos)
- Las **direcciones IP** origen y destino NO se modifican
- Si un **fragmento se pierde**, todos los fragmentos del datagrama se descartarán (esto se descubre en destino que es el que reensambla los fragmentos)

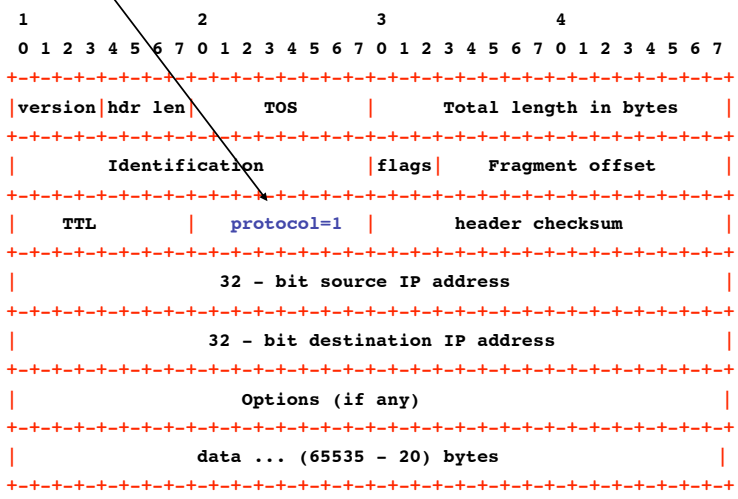
1500 (Ethernet data) = 20 (IP header) + 1480 (IP data)

1480 bytes = 552 (multiplo de 8) + 552 (multiplo de 8) + 376



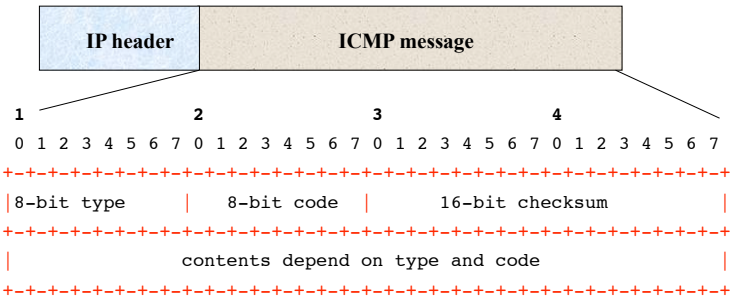
9.ICMP: Internet Control Message Protocol

mensaje ICMP



ICMP: Internet Control Message Protocol

- usado por hosts, routers, gateways para comunicar información de nivel de red
  - ICMP communica mensages de error y otras condiciones que requiera atención por parte de un router o host
  - Los mensajes van encapsulados en datagramas IP



ICMP: Internet Control Message Protocol

- Checksum cubre todo el mensaje ICMP
- Hay 15 tipos de mensajes definidos por el campo “type”
- Un mismo tipo puede emplear el campo “code” para especificar cierta condición del mensaje

Type	Code	Description	Query	Error
0	0	Echo reply (Ping reply)	x	
3	0	Network unreachable		x
	1	Host unreachable		x
	2	Protocol unreachable		x
	3	Port unreachable		x
		.....		
8	0	Echo request (Ping request)	x	
9	0	Router advertisement	x	
11	0	time-to-live exceeded		x
		.....		

10. Path determination

- “Path determination” o mejor ruta
  - Proceso por el cual un router determina los posibles caminos por los que puede reenviar un datagrama para que este llegue a su destino
    - El camino puede determinarse a partir de:
      - información introducida por el administrador de red (**estático**)
      - o a partir de información (métricas) intercambiada por los routers (**dinámico**)
    - Las métricas pueden ser muy variadas: saltos (“hops”), retardos, cargas, ancho de banda, fiabilidad del enlace, ...

IP Routing

- "Path determination" o mejor ruta (cont.)
  - La información que se intercambia los routers para permitir la determinación de un camino es particular a cada protocolo de encaminamiento, que define
    - La periodicidad con que se intercambian los paquetes de encaminamiento
    - El formato y contenido de estos paquetes de encaminamiento
    - Algoritmos asociados que permiten calcular el camino óptimo, y por tanto decidir la interfície de salida (e.g algoritmos de mínimo coste)

IP Routing

- Concepto de "convergencia" en un protocolo de encaminamiento
  - Cuando la topología de la red cambia (o cae un enlace), los routers deben recalcular las routas y actualizar las tablas de encaminamiento
  - El tiempo en que todos los routers alcanzan un conocimiento homogéneo de la red se le llama "tiempo de convergencia"
  - Tiempos de convergencia grandes implican que los routers tendrán mayor dificultad para envíar los datagramas por la interficie más adecuada
  - Convergencia depende
    - Distancia en hops desde el punto en que se produjo el cambio
    - Cantidad de routers que usan el protocolo dinámico
    - El ancho de banda y la carga de tráfico de la red
    - La carga del router (CPU)
    - El protocolo de encaminamiento usado (el algoritmo)

IP Routing

- Protocolos de encaminamiento
  - Estáticos
  - Dinàmicos

  – **Estáticos**:
    - son aquellos en los que el administrador de sistemas introduce manualmente las entradas de la tabla de encaminamiento (puertos predeterminados)
    - Util si la red es muy pequeña o cuando una red sólo puede ser alcanzado por un solo camino ("**stub network**")
    - E.g.; en UNIX con el comando "**route add/del**" se modifica la tabla y con el comando "**netstat –rn**" se observa el contenido de la tabla

IP Routing

- Protocolos de encaminamiento
  – **Dinámicos**
    - Hablan con los routers adyacentes pasando información.
    - Son aquellos que rellenan la tabla de encaminamiento de forma automática
    - Permite que la tabla cambie automáticamente cuando hay cambios en topología de la red, por tanto útil en redes grandes
    - Se pueden agrupar en 3 grandes grupos
      – **Vector-distance protocols**: determinan la dirección y distancia a que se encuentra cualquier enlace de la red ,(e.g. RIP, IGRP de cisco, BGP, ...)
      – **Link-state protocols**: recrean la topología exacta de la red (e.g.; OSPF, IS-IS)
      – **Híbridos**: combinan aspectos de los algoritmos de distancias y de los de estado del enlace

RIP (Routing Information Protocol)

- RIP
  - cada router envía periódicamene (cada 30 segundos) un datagrama de encaminamiento a cada uno de SUS VECINOS con TODA su tabla de encaminamiento
  - esta tabla indica el coste (métrica son "**hops**") para llegar a cada uno de los destinos (@IP) desde ese router
  - el router calcula con algoritmo de mínimo coste (**Algoritmo de Bellman-Ford**) la mínima distancia para llegar a los destino y actualiza su tabla (**convergencia**: debe ser rápido)

RIP (Routing Information Protocol)

- **RIP versión 1**: no anuncia máscaras (RFC1058)
  - Aplica la máscara de la interficie
  - Sino tiene, aplica la mascara de la clase por defecto de esa @IP
- **RIP versión 2**: anuncia máscaras (RFC2453)
- UNIX routing **daemons**
  - Routed (RIP v1)
  - Gated (RIP v1, v2, v3, OSPF v2, BGP v1, v2)

RIP (Routing Information Protocol)

- RIP
  - Ventajas:
    - Si la red tiene rutas redundantes, RIP es capaz de detectarlas y escoger la mejor (routing estático no)
    - Corrige fallos de la red automáticamente
    - Protocolo fácil de configurar, usar y mantener
    - Útil si la red es sencilla y sin fuertes requerimientos respecto a la buena eficiencia de la red
  - Desventajas
    - Converge muy lentamente ante fallos de la red
    - Puede crear ciclos (loops) infinitos que hagan que la red sea inconsistente
    - Debido a la vulnerabilidad ante la lenta convergencia hace que sea muy poco útil en WANs
  - "Útil para redes pequeñas y poco complejas"

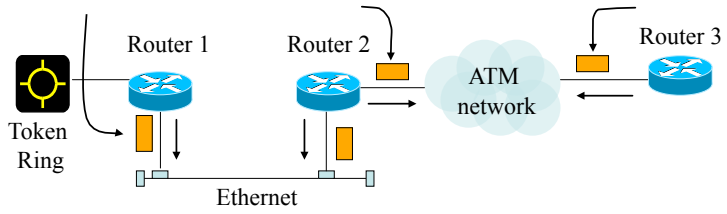
Ejemplo de RIP

- Routers cambian inform. cada 30 sec. (no están sincronizados).
- Si no, un Tout expira y el router informa a la red del cambio
- El contador de saltos para redes conectadas directamente es de 1 hop (ya que coste de Gateway consigo mismo es 0)

Ethernet:	1 hop	eth0
ATM Network:	2 hop	eth0
Token Ring	1 hops	tr0

Ethernet:	1 hop	eth0
ATM Network:	1 hop	atm0
Token Ring	2 hops	eth0

ATM Network:	1 hop	atm0
Ethernet:	2 hops	atm0
Token Ring	3 hops	atm0



RIP (Routing Information Protocol)

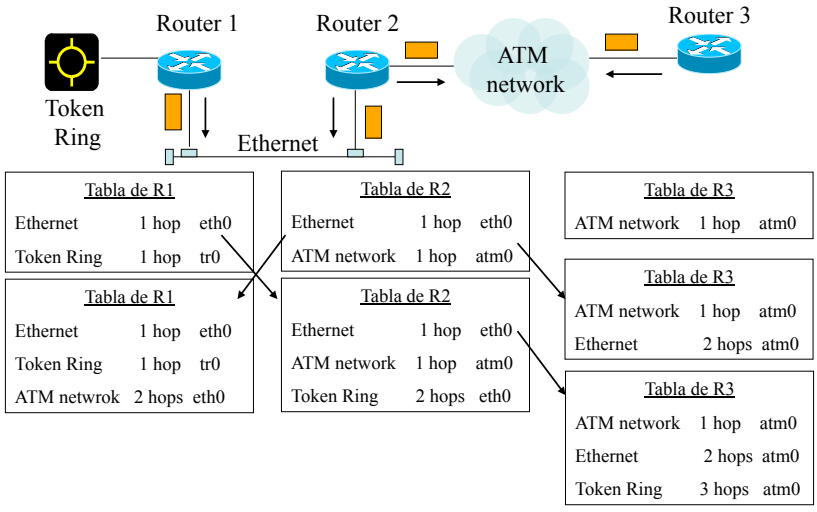
- Algoritmo de Bellman - Ford
  - d(i,j) es el coste para llegar directamente de la Red *i* a la *j*, y vale infinito si no es posible llegar directamente a una red vecina
  - D(i,j) representa la métrica de la mejor ruta entre dos redes
  - Entonces la mejor métrica se puede describir como:
    - El mínimo para llegar a la red j-sima a través de mi red vecina k-sima y se calcula como el mínimo de la suma entre el coste de llegar a mi red vecina k-sima y la métrica de llegar desde la red k-sima a la j-sima

$$D(i,i) = 0 \qquad \text{all } i$$

$$D(i,j) = \min_{k} [d(i,k) + D(k,j)] \qquad \text{otherwise}$$

Ejemplo de RIP

- Algoritmo de Bellman-Ford



RIP (Routing Information Protocol)

- **Cambios en la topología de la red**
  - Un router con RIP suele enviar mensajes de refresco cada T_{out} = 30 segundos con la tabla de encaminamiento a todos sus vecinos
  - Si hay un cambio en la topología de la red, e.g.; router cae, este no puede notificar el cambio con un mensaje de refresco (“update”)
  - Si transcurridos 6*Tout = 180 segundos, un router no ha recibido un update de su vecino, el router marcará la ruta a través de ese router como invalida
  - Una métrica de valor infinito (=16) indica que una ruta NO es válida

RIP (Routing Information Protocol)

- Limitaciones del RIP
  - RIP sólo permite 15 saltos. Considera métrica infinita cualquier router que esté más lejos de 15 saltos (**Límite de la red = 15**)
  - No tiene en cuenta métricas importantes como puede ser el retardo o el ancho de banda
  - v.1 No permite intercambiar información entre subredes (paquete RIP no informa de las máscaras de red)

RIP (Routing Information Protocol)

counting to infinite:

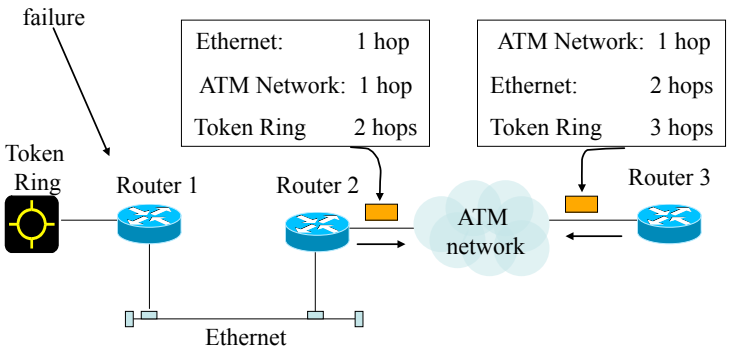
- Efecto ping-pong después de un fallo en una red
- Se cruzan mensajes entre routers en un bucle
- No termina hasta que la red caída pasa a tener distancia infinita (16 en RIP) en todos los routers

Soluciones:

- **Split horizon**: consiste en ser más selectivo haciendo que los routers que advierten omitan información de refresco que pueda "confundir" a los routers
- **poison reverse**: no omitas, advierte pero con un coste infinito

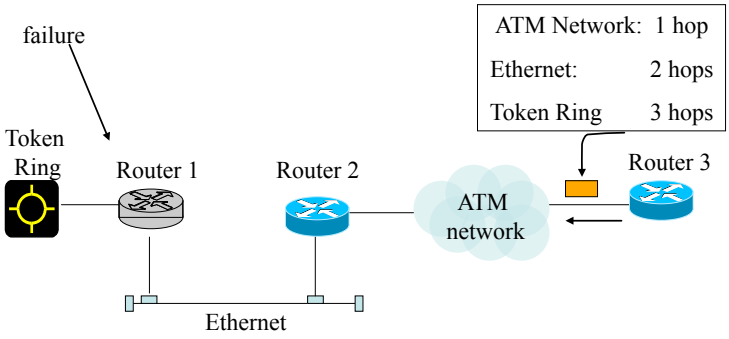
RIP (Routing Information Protocol)

- Un fallo de red en Router 1 dispara el "counting to infinite" (1):
  - Router 1 falla. Touts de 180 seg en Router 2 todavía no ha expirado



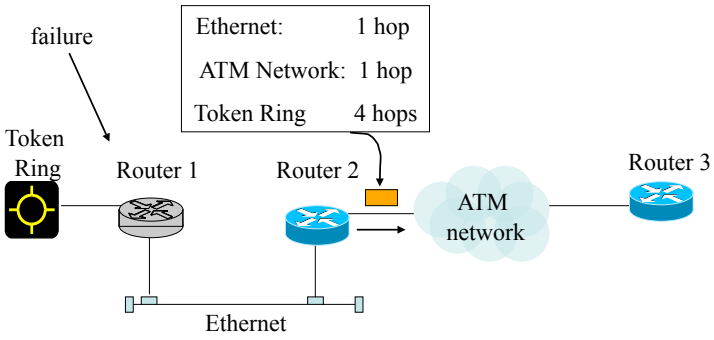
RIP (Routing Information Protocol)

- Routers comienzan el “counting to infinite” (2):
  - El Tout expira en Router 2
  - Router 2 debe refrescar su database antes de enviar un mensaje



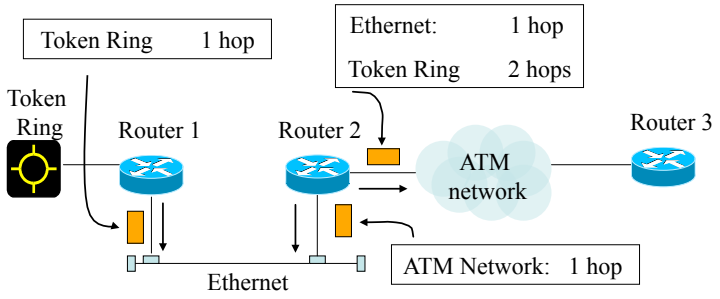
RIP (Routing Information Protocol)

- Counting to infinite (3):
  - Router 2 envía un nuevo mensaje al Router 3
  - Ambos routers deciden que Token Ring es inalcanzable cuando infinite =16



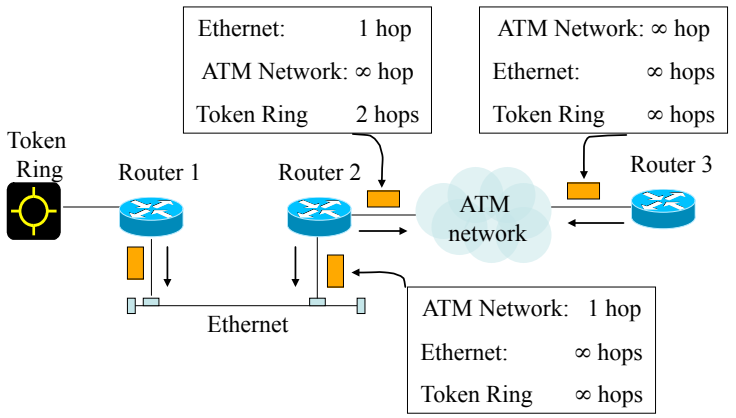
RIP (Routing Information Protocol)

- Split Horizon: evitar bucles (counting to infinite)
  - Un router NO envía información a otro router de las redes que le son comunicadas por ese otro router
  - Router 3 no envía refrescos de la tabla
  - Router 2 informa al Router 3 solo de la Ethernet y Token Ring.
  - Router 2 informa al Router 1 solo de la ATM
  - Router 1 sólo informa de Token Ring



RIP (Routing Information Protocol)

- Poison Reverse: advertir costes infinitos



RIP (Routing Information Protocol)

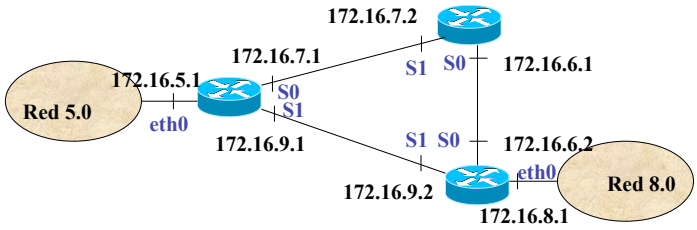
- Poison Reverse: advertir costes infinitos
  - Cuando detecta que su antigua ruta por un interface no es válida envía un mensaje a los otros interfaces en el que se indica que su coste es infinito, por lo que los routers que reciben el mensaje tardan menos en saber que esta ruta no es válida. El problema que tenemos es que incrementa el tamaño de los mensajes de encaminamieno.

RIP (Routing Information Protocol)

- **Triggered updates**
  - Consiste en que si un router ha cambiado su tabla debido a un cambio en la topología tardará 30 segundos en el peor de los casos en avisar del nuevo cambio a un router vecino
  - Eso hace que un el tiempo de convergencia pueda ser muy alto (minutos) ante cambios en la topología de la red
  - "triggered update" consiste en enviar la tabla enseguida de que se produzca un cambio en la red sin tener que esperar los 30 segundos, mejorando por tanto el tiempo de convergencia
- Actualmente se usa por un lado split-horizon para evitar bucles (loops) y por otra triggered updates conjuntamente con Poisson reversed para converger rápidamente en caso de que haya cambios en la topología de la red.

Ejercicio

- Indica la información que se comunicarían los routers de la siguiente figura en los casos siguientes:
  - No hay split-horizon ni poisson-reverse
  - Hay split-horizon, pero no poisson-reverse
  - No hay split-horizon y si que hay poisson-reverse y la red 8.0 deja de ser alcanzable



OSPF (Open-Short Path First)

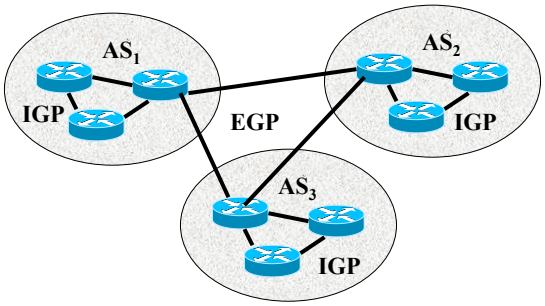
- Se dibuja un mapa con toda la topología de la red
- Cada router envía información a TODOS los routers de la red cuando se produzca un cambio en la topología de la red
- A partir de esa información se recalcula la tabla de encaminamiento usando el algoritmo de Dijkstra
- OPSF se basa en:
  - Enviar LSAs (Link State Advertisements) con los cambios que se producen en la red (LSAs van encapsulados en IP)
  - Mantener una base de datos con la topología de la red (Link State Database) en cada router
  - Mantener una tabla de encaminamiento con los caminos y puertos
  - Un algoritmo de encaminamiento (Dijkstra) que rellena la tabla a partir del contenido de la base de datos

11. Sistemas Autónomos (AS)

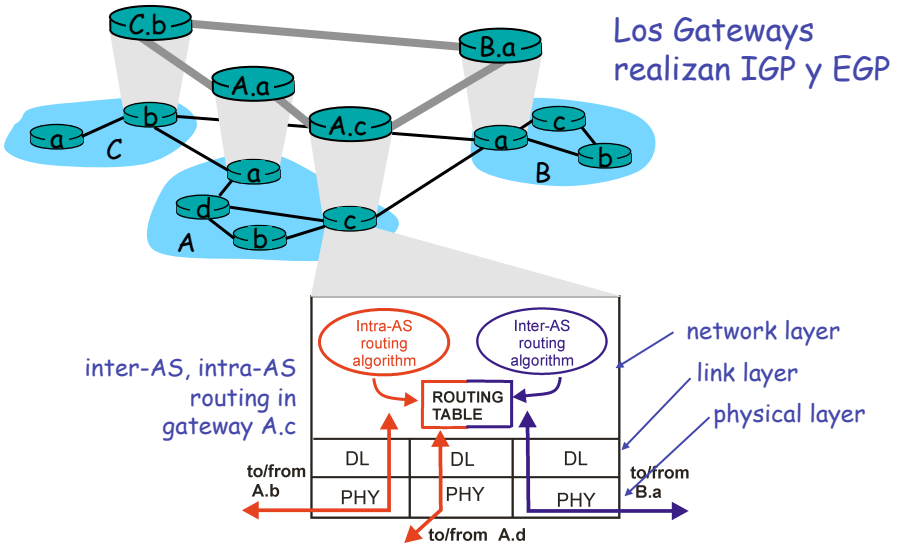
- Internet: con más de 200 millones de destinaciones: no se pueden almacenar en las tablas de routing.
  → Internet = red de redes
  – Internet se organiza como una colección de AS, cada uno de ellos administrado por una única entidad
    - cada uno su propia política de routing
    - routers en el mismos AS ejecutan el mismo protocolo
    - gateway routers: routers especiales que son los responsables para comunicarse con el exterior.

Sistemas Autónomos (AS)

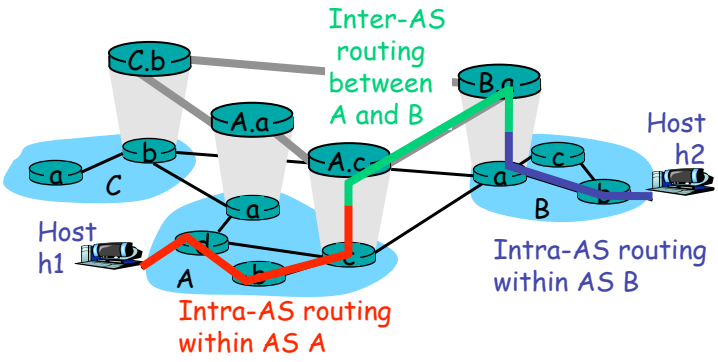
– El protocolo de encaminamiento que comunica routers dentro de un AS se le llama IGP (**Interior Gateway Protocol**)(e.g.; RIP, OSPF, IGRP, EIGRP)
– El protocolo de encaminamiento que comunica routers de distintos AS se le llama EGP (**Exterior Gateway Protocol**) (e.g.; EGP, BGP)



Sistemas Autónomos (AS)



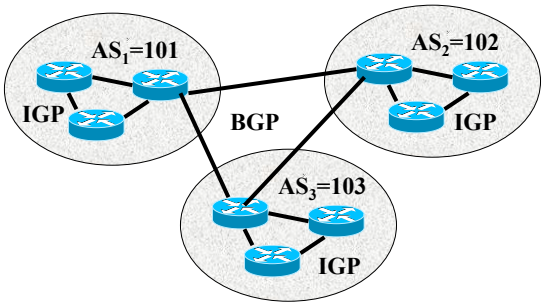
Sistemas Autónomos (AS)



- Veremos a continuación:
  - **Interior Gateway Protocol**: RIP, OSPF
  - **Exterior Gateway Protocol**: BGP

BGP (Border Gateway Protocol)

- Es un EGP (Exterior Gateway Protocol) usado para comunicar AS
  - Inicialmente se usó bastante un protocolo llamado EGP (versión 3), pero fue rechazado por su gran ineficiencia
  - Hoy en día se usa BGP (versión 4) como EGP
- BGP permite conectar Sistemas Autónomos (AS) que pertenezcan a distintas organizaciones



BGP (Border Gateway Protocol)

- BGP es un protocolo de encaminamiento que se basa en políticas de red (organizaciones) y no en métricas
  - Un AS multihomed puede rechazar actuar como AS de tránsito a otro AS
  - Un AS multihomed puede convertirse en un AS de tránsito para un conjunto restrictivo de AS
  - Un AS de tránsito puede desfavorecer ciertos AS para llevar tráfico él mismo
- Elección de rutas: elegir un camino basándose en la preferencia
  - Atravesar menor número de AS
  - Consideraciones administrativas
  - Presencia o ausencia de ciertos AS en el camino
  - Origen de la ruta
  - Dinámica de los enlaces

11. Detalles:

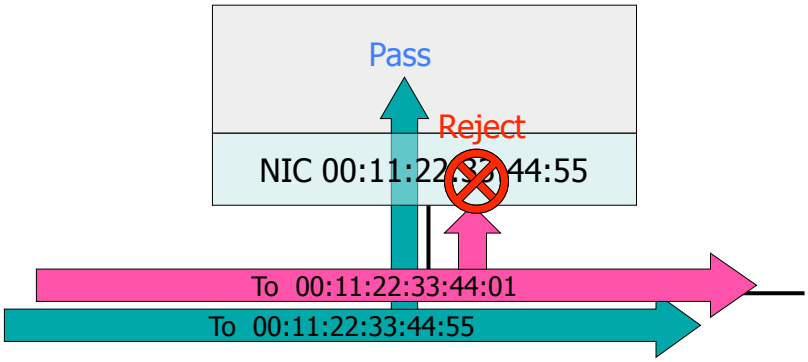
- Modo promiscuo
- Interficie loopback
- MTU Path discovery
- Ping
- TTL
- Traceroute
- Cache ARP

Modo promiscuo

"Tcpdump es un programa de red que pone la tarjeta de red en ***modo promiscuo****:* recibe todos los paquetes/tramas que pasan por esa interficie independientemente de que la dirección MAC destino sea la de la tarjeta (normalmente la tarjeta solo captura aquellas tramas que contienen su dirección destino). Eso significa que tcpdump captura todas las tramas que pasan por la interficie a la que esta conectada.

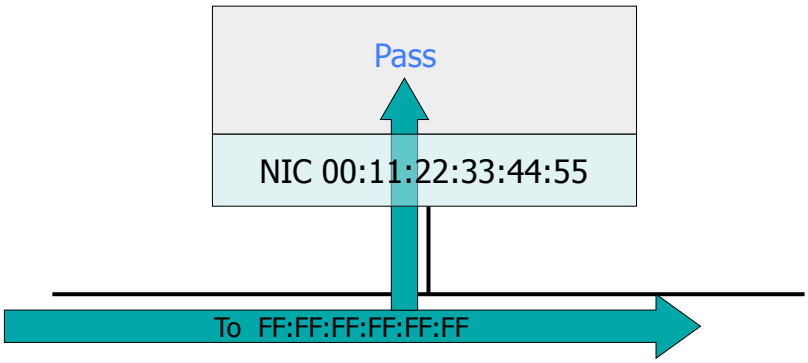
¿modo promiscuo ?

- Unicast: The packet to the HW address of the device is passed.



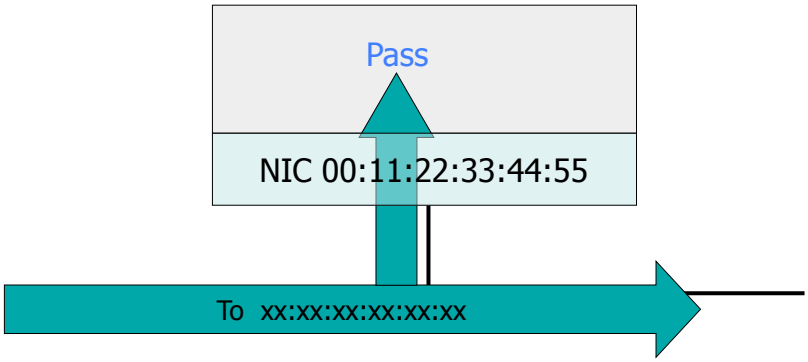
¿modo promiscuo ?

- Broadcast : Packet to broadcast (FF:FF:FF:FF:FF:FF) is passed



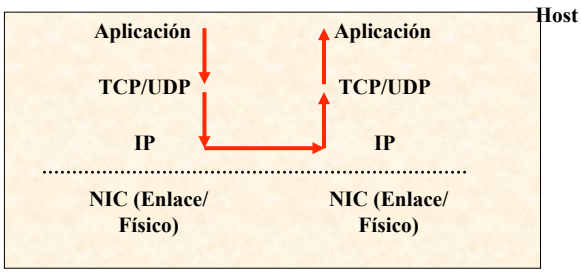
¿modo promiscuo ?

- Promiscuous: All packets are passed.



Interficie Loopback

- Interficie que permite comunicarse a dos procesos dentro de un mismo host
- Los datos de la aplicación bajan los niveles de transporte y de red (niveles 4 y 3) pero no llegan al nivel de enlace (nivel 2)
- En el caso de IP, la dirección IP loopback está estandarizada y es la @IP = 127.0.0.1



MTU Path Discovery (RFC 1191)

- Objetivo: evitar la fragmentación de datagramas averiguando cual es la Mínima MTU entre el origen y el destino
- ¿Cómo conseguirlo?
  - Se envía un datagrama con MTU la del enlace y con el bit Don't Fragment activo
  - Cuando un router se encuentre que tiene una MTU menor que la que le llegue no fragmentará y enviará un mensaje ICMP (type = 3, code = 4 "fragmentation needed but don't fragment bit set")
  - Este mensaje ICMP advierte cual es la MTU del enlace que necesita fragmentar (sino soporta esta opción, advierte MTU = 0)
  - El origen vuelve a empezar con la nueva MTU hasta que averigue la mínima MTU, si la MTU advertida es 0, lo intenta con MTU conocidas más pequeñas

  → en IPv6 és obligat implementar aquest protocol

Ping command

- Se usa como herramienta de diagnóstico para averiguar
  - Si un host está conectado y es accesible
  - Si los routers intermedios son operativos
  - Tu propio host (software IP) funciona correctamente
  - Ping envía "echo requests" a un host determinado. Este le devuelve un "echo reply"
  - Los echo request/reply son mensajes ICMP
  - Ping devuelve información del tipo "retardo desde cliente a servidor", valor del TTL, cantidad de paquetes ICMP perdidos

ping [**-dfLnqRrv**] [**-c** count] [**-I** ifaddr] [**-i** wait] [**-l** preload] [**-p** pattern] [**-S** ifaddr] [**-s** packetsize] [**-t** ttl] [**-w** maxwait] host

Ping command

ping –c3 aucanada

PING aucanada.ac.upc.es (147.83.35.24): 56 data bytes

64 bytes from 147.83.35.24: icmp_seq=0 ttl=255 time=0.093 ms

64 bytes from 147.83.35.24: icmp_seq=1 ttl=255 time=0.074 ms

64 bytes from 147.83.35.24: icmp_seq=2 ttl=255 time=0.079 ms

--- aucanada.ac.upc.es ping statistics ---

3 packets transmitted, 3 packets received, 0% packet loss

round-trip min/avg/max = 0.074/0.082/0.093 ms

ping -c3 -s512 rogent

PING rogent.ac.upc.es (147.83.31.7): 512 data bytes

520 bytes from 147.83.31.7: icmp_seq=0 ttl=254 time=1.530 ms

520 bytes from 147.83.31.7: icmp_seq=1 ttl=254 time=1.582 ms

520 bytes from 147.83.31.7: icmp_seq=2 ttl=254 time=1.584 ms

--- rogent.ac.upc.es ping statistics ---

3 packets transmitted, 3 packets received, 0% packet loss

round-trip min/avg/max = 1.530/1.565/1.584 ms

Comando Ping

- Ping envía “echo requests” a un host determinado. Este le devuelve un “echo reply”
- Los echo request/reply son mensajes ICMP
- Ping devuelve información del tipo “retardo desde cliente a servidor”, valor del TTL, cantidad de paquetes ICMP perdidos

Ejemplo de ping

```
[foobar-32] ping ac.upc.es
PING ac.upc.es (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: icmp_seq=0 ttl=240 time=37 ms
64 bytes from 192.168.0.1: icmp_seq=1 ttl=240 time=35 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=240 time=32 ms
64 bytes from 192.168.0.1: icmp_seq=4 ttl=240 time=35 ms
64 bytes from 192.168.0.1: icmp_seq=6 ttl=240 time=37 ms
64 bytes from 192.168.0.1: icmp_seq=7 ttl=240 time=33 ms
64 bytes from 192.168.0.1: icmp_seq=8 ttl=240 time=37 ms
64 bytes from 192.168.0.1: icmp_seq=9 ttl=240 time=54 ms
64 bytes from 192.168.0.1: icmp_seq=10 ttl=240 time=63 ms
64 bytes from 192.168.0.1: icmp_seq=11 ttl=240 time=34 ms
64 bytes from 192.168.0.1: icmp_seq=12 ttl=240 time=36 ms

----ac.upc.es PING Statistics----
13 packets transmitted, 11 packets received, 15% packet loss
round-trip (ms)  min/avg/max = 32/40/63 ms
```

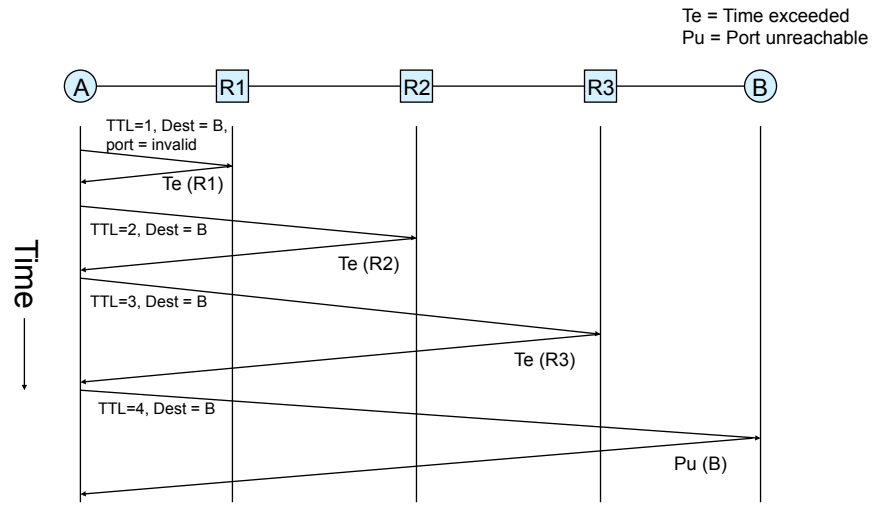
TTL (Time To Live)

- Campo dentro de cabecera IP
  - Campo que indica el límite de routers que puedes atravesar en Internet
  - Se inicializa en cada datagrama con un valor como máximo de 255 (8 bits de campo)
  - Cada vez que el datagrama atraviesa un router se decrementa en 1
  - Si un datagrama llega a un router y su TTL = 0, el router descarta el datagrama y envía un mensaje ICMP (mensaje con tipo = 11)
  - Este campo se utiliza en el programa “traceroute” para averiguar la ruta que atraviesa un datagrama cuando viaja por Internet, también el programa “ping” suele indicar el TTL

Traceroute command

- Programa que permite averiguar la ruta que ha seguido un datagrama cuando viaja por Internet
- Se aprovecha de que cuando un datagrama llega a un router con el campo TTL = 0 este es descartado y el origen recibe un mensaje ICMP (type = 11, code = 0)
- Programa envía:
  - datagramas UDP con la cabecera IP con el TTL =1, 2, 3, 4, ... (envía 3 datagramas con cada TTL) hasta que se llegue al destino
  - Como el puerto UDP destino es desconocido, el host destino devuelve un error ICMP de destino no alcanzable (unreachable port, type=3, code=3)
  - Además el datagrama lleva en su campo de datos un número de secuencia, una copia del TTL y un timestamp con el tiempo en que se envió el datagrama para dar estadísticas

Traceroute command



Traceroute command

traceroute [**-l**] [**-m** max_ttl] [**-n**] [**-p** port] [**-q** nqueries] [**-r**]
[**-s** src_addr] [**-t** tos] [**-w** waittime] host [packetsize]

• **traceroute fonoll**

traceroute to fonoll.ac.upc.es (147.83.31.14), 30 hops max, 40 byte packets
1 arenys5.ac.upc.es (147.83.35.2) 1 ms 1 ms 2 ms
2 fonoll.ac.upc.es (147.83.31.14) 1 ms * 1 ms

• **traceroute -q 4 fonoll 512**

traceroute to fonoll.ac.upc.es (147.83.31.14), 30 hops max, 512 byte packets
1 arenys5.ac.upc.es (147.83.35.2) 2 ms 2 ms 1 ms 2 ms
2 fonoll.ac.upc.es (147.83.31.14) 1 ms * 1 ms *

Comando arp

La comanda arp permet veure i modificar manualment la taula que manté el mòdul ARP (*Address Ressolution Protocol*). En aquesta taula hi ha la correspondència entre les adreces IP i les adreces *hardware*. Les invocacions típiques són:

arp -s adreça_IP adreça_hw

Assigna l'adreça *hardware* adreça_hw a l'adreça IP adreça_IP.

Promiscuo?

Volem saber si l'adreça 170.34.6.2 està en modalitat promiscua.

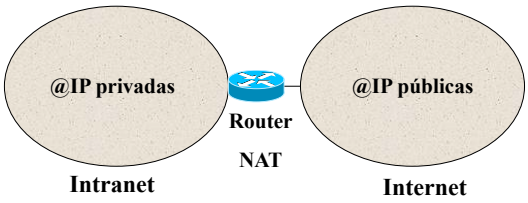
podem fer-ho amb arp i ping?

13. NAT

- Direcciones privadas (RFC 1918)
  - Direcciones privadas definidas por IANA: son direcciones que no son enrutables en Internet
    - Clase A: 10.0.0.0 – 10.255.255.255 → CIDR 10.0.0.0/8
    - Clase B: 172.16.0.0 – 172.31.255.255 →CIDR 172.16.0.0/12
    - Clase C: 192.168.0.0 – 192.168.255.255
    - → CIDR 192.168.0.0/16
  - Ideales para Labs o Test-home networks
  - Ideal en Intranets
  - Ideal en WAN links (core backbones) para ahorrar direcciones globales (curioso, no?)
  - Problema: no son enrutables por Internet

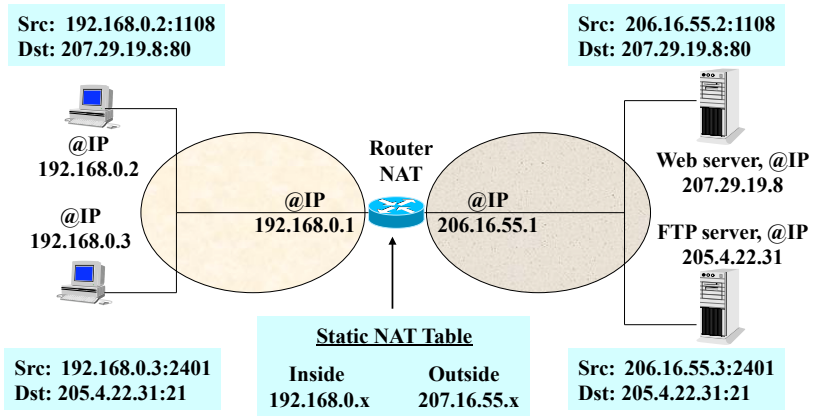
NAT

- NAT (Network Address Translation) (RFC 1631)
  - Definition: Maps @IP from one realm to another
  - Mecanismo que permite la traducción de direcciones privadas a públicas para poder acceder a internet desde una intranet
  - Necesitamos un Router NAT en la frontera entre las redes que queremos traducir
  - El mecanismo debe ser transparente a los usuarios finales
  - Compatibilidad con firewalls y con seguridad en Internet



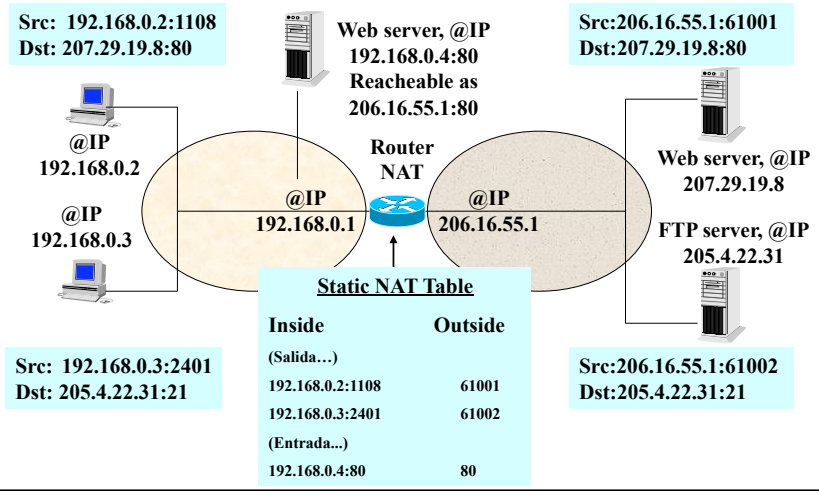
NAT estático

- consiste en substituir la parte de host de la @IP privada en el host de la @IP pública



NAT dinámico

- (PAT: Port Address Translation): el router tiene una sóla @IP pública, y elige un nuevo puerto origen y mapea las @IP privadas a partir del puerto designado

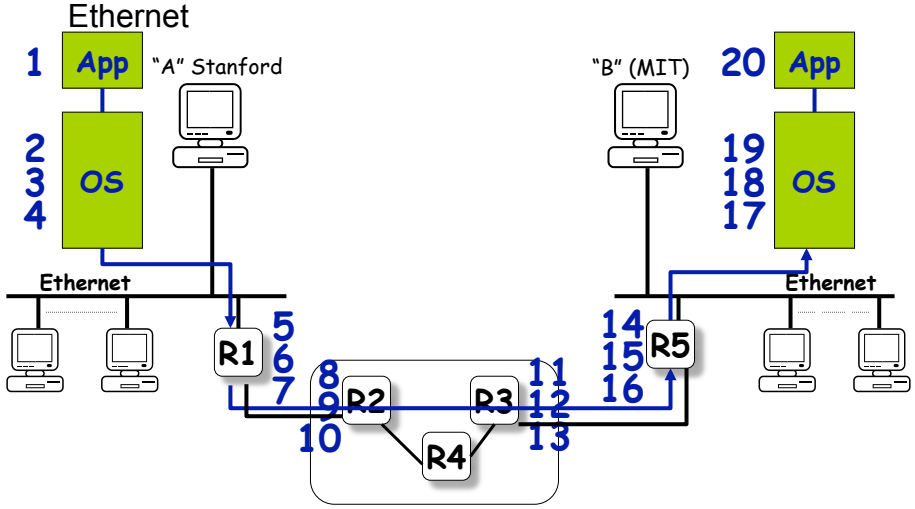


Protocolos sensibles a NAT

- NAT modifica cabecera IP → recalcular el checksum IP y TCP
- Protocolos que llevan embebida la @IP → también debe ser modificada → ALG (Application-Level Gateway)
  - ICMP: “Destination unreachable messages” llevan @IP embebidas
  - Comandos FTP llevan @IP embebidas como “strings” (cambiarlas además implica que cambia la longitud del segmento TCP)
  - SNMP (Simle Network Management Protocol)
  - NetBIOS over TCP/IP (NBT)
- NAT + Firewalls (Algunos protocolos no funcionan)
  - DNS, Kerberos, X-Windows, remote-shell, SIP, ... (ver Internet Draft “Protocol Complications with the IP Network Address Translation”)

Repaso: pasos de un FTP

- Example: FTP over the Internet Using TCP/IP and Ethernet



font: curs CS244a de http://www.stanford.edu/~nickm

T6
NIVELL TRANSPORT

Xarxes de Computadors i Aplicacions

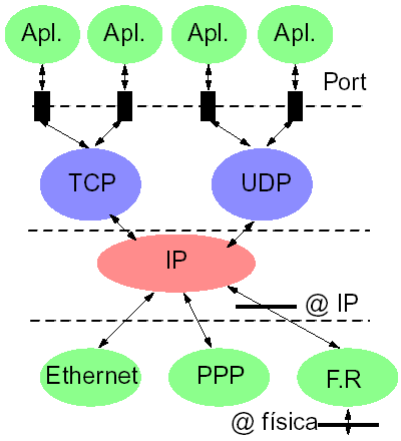
PAU ARTIGAS, DAVID CARRERA i JORDI TORRES
Departament d'Arquitectura de Computadors
UPC, setembre - 2009

Contenido

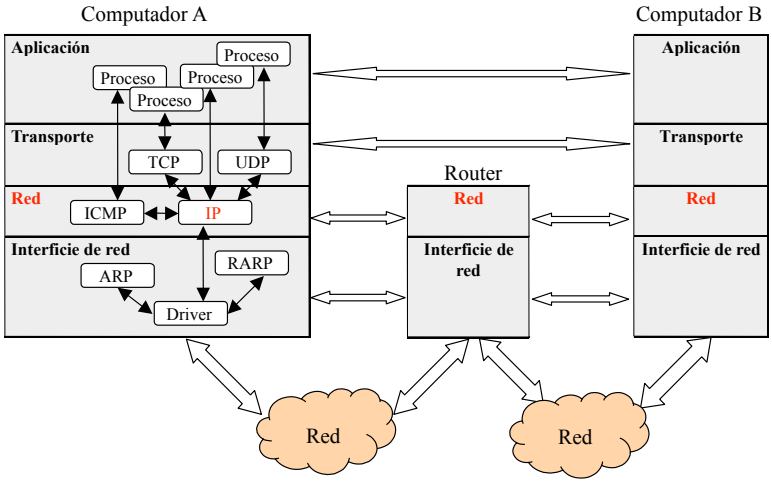
transparències basades en el material docent dels professors José M. Barceló i Jordi Torres de l'assignatura STD del pla 91 de FIB.

1. Introducció: nivel 4 - transport

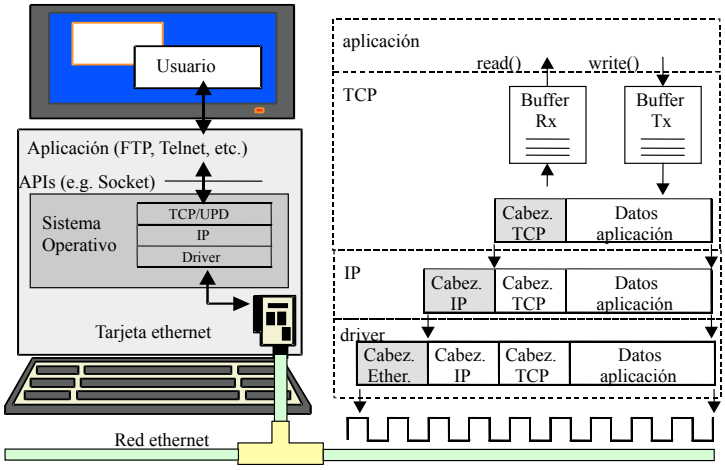
- Entre IP y las aplicaciones tenemos dos posibles protocolos:
  - UDP (No orientado a la conexión)
  - TCP (Orientado a la conexión)



1. Introducció: Arquitectura TCP/IP



1. Introducció: Arquitectura TCP/IP

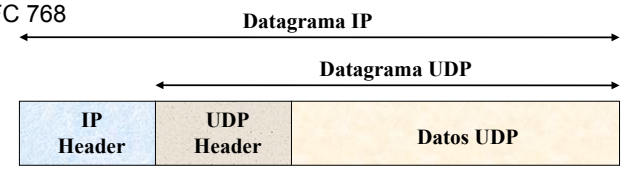


Contenido T5

1. Introducción
2. UDP (User Datagrama Protocol)
3. TCP (Transmission Control Protocol)
4. Cabecera TCP
5. Establecimiento (3wHS) y cierre de la conexión TCP
6. Grafo de estados TCP
7. Control de flujo en TCP
8. Control de errores en TCP
9. Tipos de aplicaciones que usan TCP
10. Control de congestión en TCP
11. Sockets

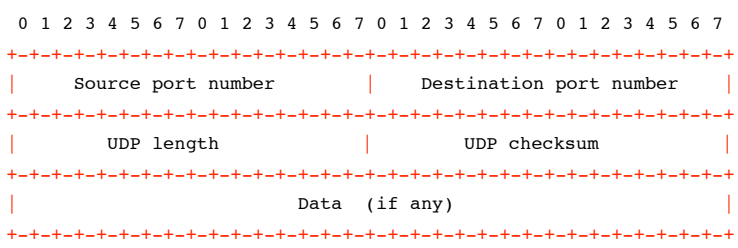
2. UDP (User Datagram Protocol)

- Protocolo de transporte no-orientado a la conexión, cuya unidad de encapsulamiento es el datagrama UDP
- Ideal para comunicaciones en tiempo real
- Cada escritura por parte de la aplicación provoca la creación de un Datagrama UDP
- Cada datagrama UDP creado provoca la creación de un datagrama IP en el nivel 3 (lo veremos en el siguiente tema)
- Si se pierde el datagrama IP o UDP es problema de la aplicación remota incorporar mecanismos de retransmisión
- RFC 768



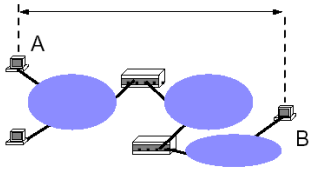
2.UDP (User Datagram Protocol)

- Datagrama UDP (8 bytes de cabecera)
  - **Puertos:** identifican a la aplicación origen y destino
  - **UDP length**: longitud total del datagrama UDP (campo redundante ya que IP lleva la longitud también)
  - **UDP checksum**: detector de errores que aplica a TODO el datagrama (checksum IP sólo cubría la cabecera IP)



3.TCP (Transmission Control Protocol)

- Es un protocolo extremo a extremo



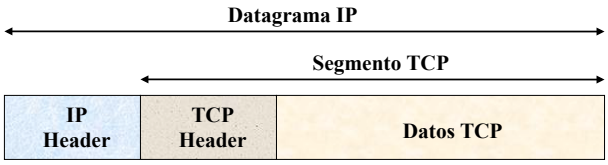
- Utilizando TCP aseguramos que
  - La información llega de forma correcta.
    (es decir, la información incorrecta es reenviada)
  - Y en el orden correcto.
  - Además
    - permite al consumidor no ser inundado por la información del productor.
    - y protege a la red de congestión.

3.TCP (Transmission Control Protocol)

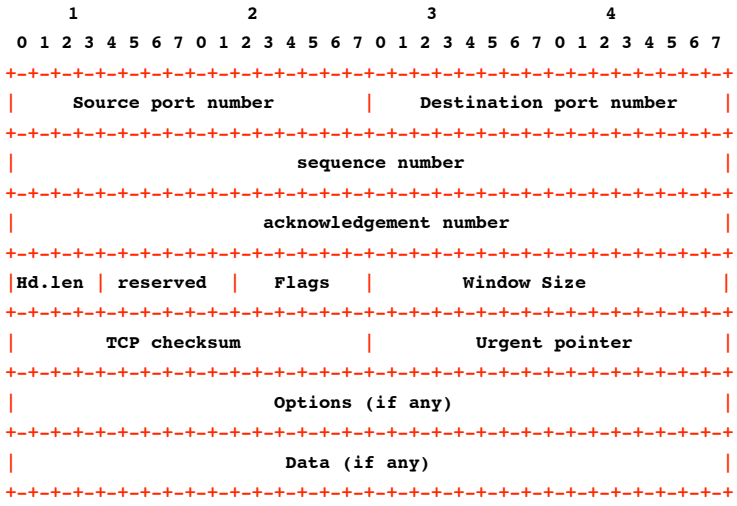
- Protocolo orientado a la conexión:
  - estableciemiento de la conexión,
  - envío de datos
  - y cierre de la conexión
- Tal vez el protocolo MAS complejo e importante de la pila de protocolos.
- **Unidad de datos es el "segmento TCP"**
- Proporciona fiabilidad mediante
  - el control de flujo (**ventana advertida**)
  - control de errores (**ARQ**)
  - y control de la congestión (**ventana de congestión**)

3.TCP (Transmission Control Protocol)

- Los segmentos pueden llegar fuera de orden (debajo hay IP que es no orientado a la conexión, o sea, datagrama),
- por tanto, TCP debe reordenar los segmentos antes de pasarlos a la aplicación



4. Cabecera TCP



4. Cabecera TCP

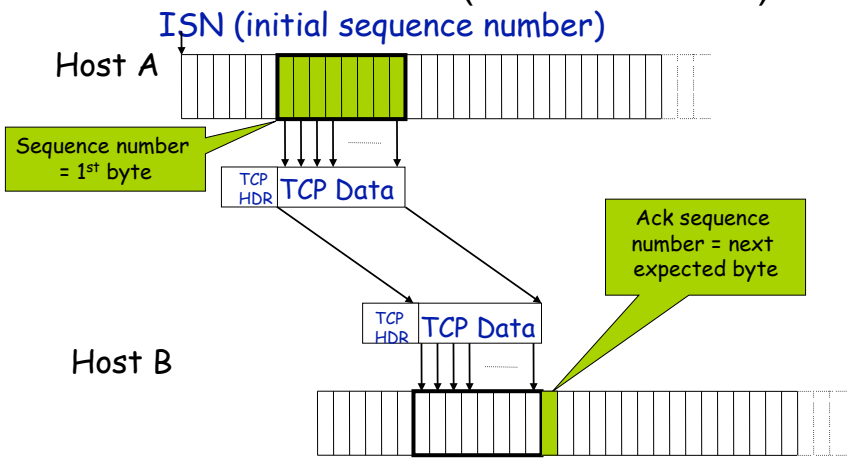
– **Puertos**: identifican las aplicaciones

– **Sequence number**: identifica el primer byte dentro de ese segmento de la secuencia de bytes enviados hasta ese momento.

- ISN (Initial Seq. Number): primer número de secuencia escogida por el protocolo TCP
- Seq. Number será un número a partir de ISN. Por tanto para saber cuantos bytes llevamos enviados hay que hacer “Seq. Number – ISN”

4. Cabecera TCP

- Números de secuencia (finestra lliscant)



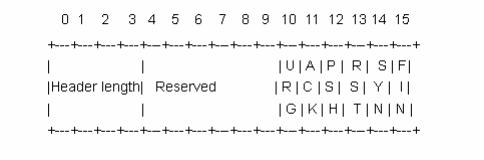
4. Cabecera TCP

– **Ack Number**: contiene el próximo número de seq. que el transmisor del ACK espera recibir
  - Por tanto es el “Seq. Number+1” del último byte recibido correctamente
  - Cuidado !!! TCP es FULL-DUPLEX: cada extremo mantiene un Seq. Number y un Ack Number

– **Header length**: longitud total de la cabecera (opciones variable) en múltiplos de 4 bytes

4. Cabecera TCP

– **Flags**: hay 6 flags (bits) en la cabecera
  - URG: Urgent Pointer field valido
  - ACK: Ack Number es valido
  - PSH: el receptor debe pasar los datos a la aplicación tan rápido como sea posible (por ahora no hay más datos)
  - RST: “Reset” la conexión
  - SYN: Sincronización de los números de secuencia al iniciar la conexión
  - FIN: termina la conexión



4. Cabecera TCP

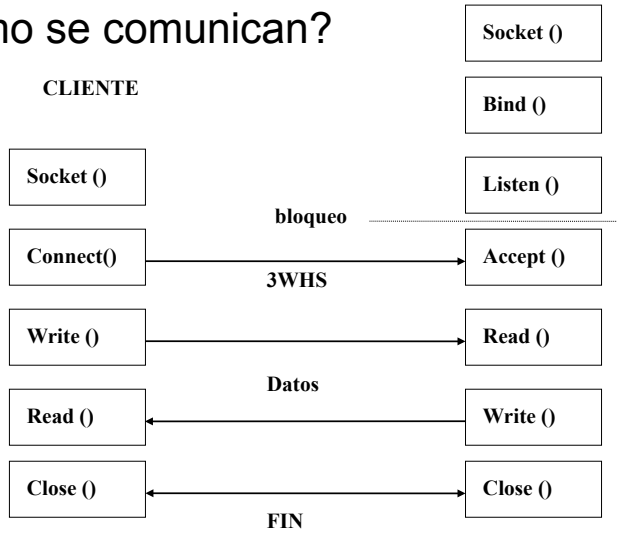
- **Window Size**: tamaño de la ventana advertida por el receptor al transmisor (Sliding Window). Máxima ventana = 65535 bytes
- **Checksum**: de todo el segmento TCP (igual que UDP)
- **Urgent Pointer**: puntero al Seq. Number que indica la parte de datos urgentes dentro del campo de datos
- **Options**: opción de anunciar el MSS (Maximum Segment Size)

Contenido T5

1. Introducción
2. UDP (User Datagrama Protocol)
3. TCP (Transmission Control Protocol)
4. Cabecera TCP
5. Establecimiento (3wHS) y cierre de la conexión TCP
6. Grafo de estados TCP
7. Control de flujo en TCP
8. Control de errores en TCP
9. Tipos de aplicaciones que usan TCP
10. Control de congestión en TCP
11. Sockets

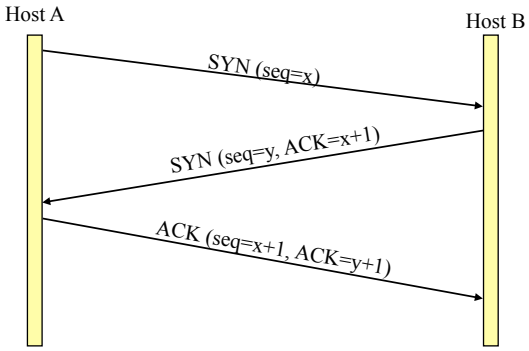
5. Programa cliente-servidor (repaso)

- ¿como se comunican?



5. Establecimiento de la conexión TCP

– Usa el 3-Way Handshake Algorithm:

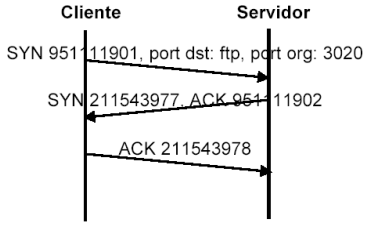


5. Establecimiento de la conexión TCP

- Usa el 3-Way Handshake Algorithm:
  - El cliente envía un **segmento SYN** especificando el puerto destino del servidor, su puerto origen (escogido por el Kernel) y el ISN (Initial Seq Number) escogido al azar por el Kernel
  - El receptor (servidor) devuelve un **segmento SYN+ACK** reconociendo el segmento SYN e indicando su ISN
  - El cliente responde con un **segmento ACK** reconociendo el SYN+ACK
- Una vez establecido la conexión se pasa a la fase de envio de datos
- Es posible negociar ("**indicar**") opciones (e.g.; indicar el MSS)
  - **MSS (Maximum Segment Size):** tamaño máximo de un segmento TCP.
    - Viene fijado por el Kernel: default = 536 bytes para un datagrama IP de 576 bytes (histórico X.25)
    - Sino, fijado por el MTU (Maximum Transfer Unit) menos cabeceras

Ejemplo de conexión TCP

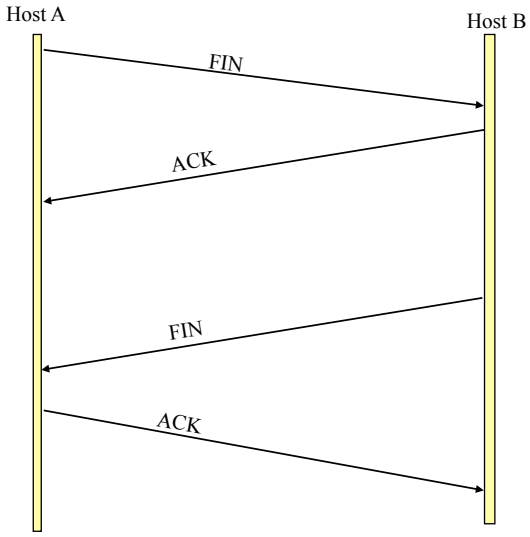
- Ejemplo de establecimiento de la conexión TCP: (recordar que usa el 3-Way Handshake Algorithm):
  - representación "time line":



  - TCP output simplificado:

```
11:27:13.771041        147.83.35.18.3020 > 147.83.32.14.ftp: S
                              951111901:951111901(0)
11:27:13.771491        147.83.32.14.ftp > 147.83.35.18.3020 : S
                              211543977:211543977(0) ack 951111902
11:27:13.771517        147.83.35.18.3020 > 147.83.32.14.ftp: .
                              ack 211543978
```

5. Cierre de la conexión TCP

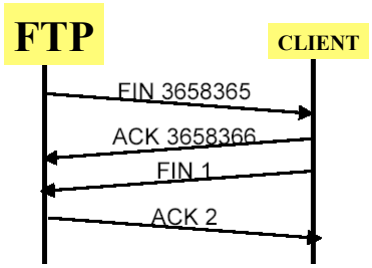


5. Cierre de la conexión TCP

- El cierre de la conexión puede ser debido a varias causas:
  - El cliente o el servidor cierran la conexión (e.g.; LLS close())
  - Por alguna razón se envía un reset de la conexión (flag activo RST)
  - Cierre debido a una interrupción, e.g.; ^D o un ^C, etc, ...
- El cierre normal es debido a un close del cliente lo que provoca el envío de 4 segmentos TCP
- Como la conexión TCP es FDX cada dirección debe cerrar la conexión (envío de segmento FIN y de su correspondiente ACK)
- Es posible que un extremo cierre su lado de la conexión y el otro no. En ese caso, el extremo que no ha cerrado puede enviar datos y el otro extremo los reconocerá (ACKs) aunque haya cerrado su conexión

Ejemplo de conexión TCP

- Cierre de la conexión TCP




```
11:27:19.397349        147.83.32.14.ftp > 147.83.35.18.3020 :
                       F 3658365:3658365(0)

11:27:19.397370        147.83.35.18.3020 > 147.83.32.14.ftp:
                       . 1:1(0) ack 3658366

11:27:19.397453        147.83.35.18.3020 > 147.83.32.14.ftp:
                       F 1:1(0)

11:27:19.398437        147.83.32.14.ftp > 147.83.35.18.3020 :
                       . 3658366:3658366(0) ack 2
```

Cabecera TCP

paquete:

4500 05dc e597 0000 4006 2687 9353 2350
9353 1f07 **0017 82c0 99a5 68e3 4693 e23b**
5018 3fe0 338c 0000 203e ...

cabecera TCP:

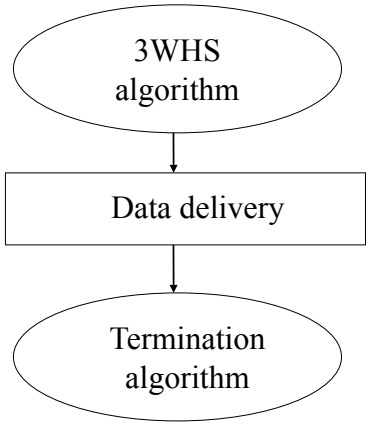
```
0000 0000 0001 0111 1000 0010 1100 0000
1001 1001 1010 0101 0110 1000 1110 0011
0100 0110 1001 0011 1110 0010 0011 1011
0101 0000 0001 1000 0011 1111 1110 0000
0011 0011 1000 1100 0000 0000 0000 0000
```

- port origen?
- port dest.?
- num. sec.?
- num. ack?
- hlen?

- urg?; ack?; psh?; srt?; syn?; fin?
- tam. vent?
- checksum?
- punt. urg?

6. Grafo de estados en una conexión TCP

– En total 11 estados distintos: CLOSE, ESTABLISHED, LISTEN, LAST_ACK, ...



6. Grafo de estados en una conexión TCP

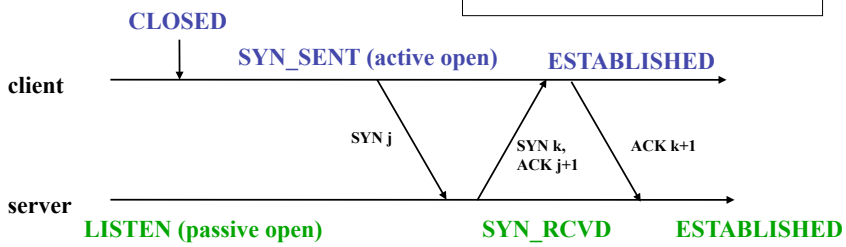
- Estados posibles en una conexión TCP

Active open	**CLOSED**	The socket is not being used.
	SYN_SENT	Actively trying to establish connection.
Passive open	**LISTEN**	Listening for incoming connections.
	SYN_RECEIVED	Initial synchronization of the connection under way.
	ESTABLISHED	Connection has been established.
Active close	**FIN_WAIT_1**	Socket closed; shutting down connection.
	CLOSING	Closed, then remote shutdown; awaiting acknowledgment.
	FIN_WAIT_2	Socket closed; waiting for shutdown from remote.
	TIME_WAIT	Wait after close for remote shutdown retransmission.
Passive close	**CLOSE_WAIT**	Remote shutdown; waiting for the socket to close.
	LAST_ACK	Remote shutdown, then closed; awaiting acknowledgment.

6. Grafo de estados en una conexión TCP

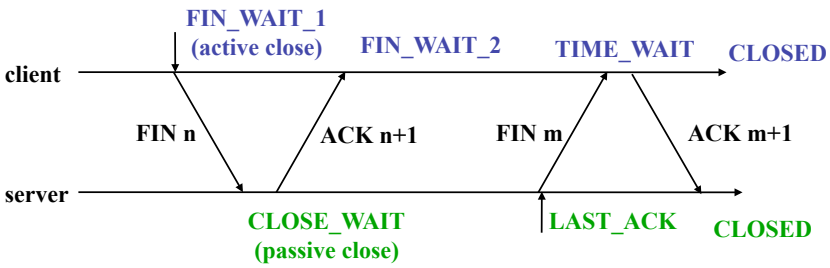
- Estados en el establecimiento de la conexión TCP:

"Time Line horizontal"
AZUL: estados que pasa el cliente
VERDE: estados que pasa el servidor



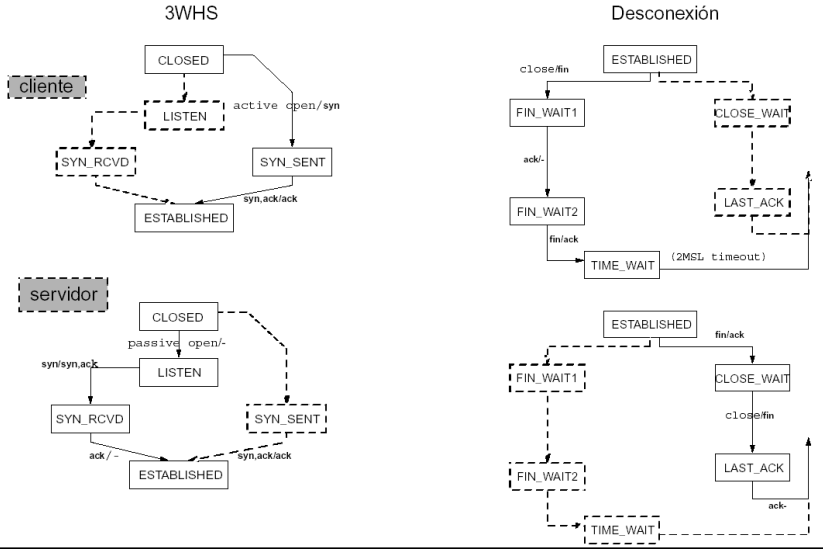
6. Grafo de estados en una conexión TCP

- Estados en el cierre de la conexión TCP:



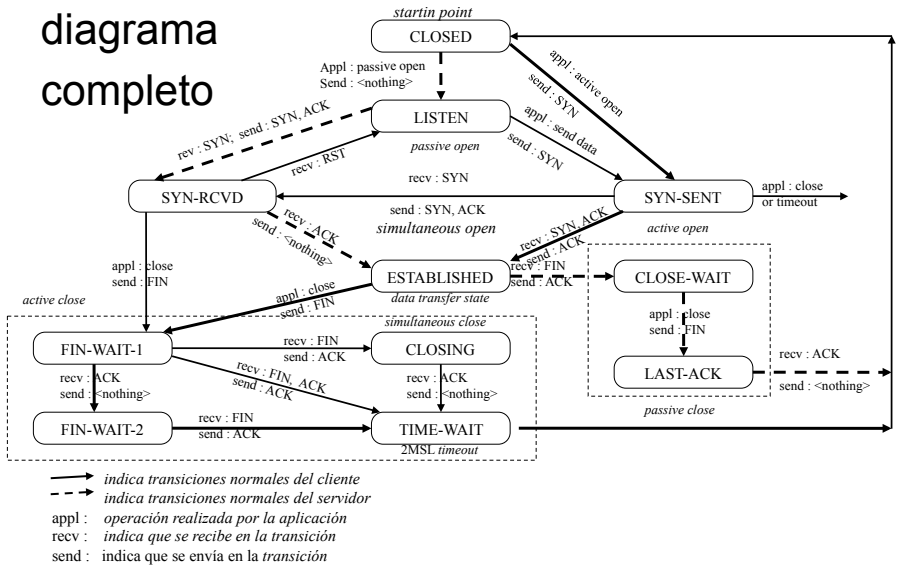
6. Grafo de estados en una conexión TCP

- ya hemos visto que los estados en cada extremo de la conexión són diferentes:



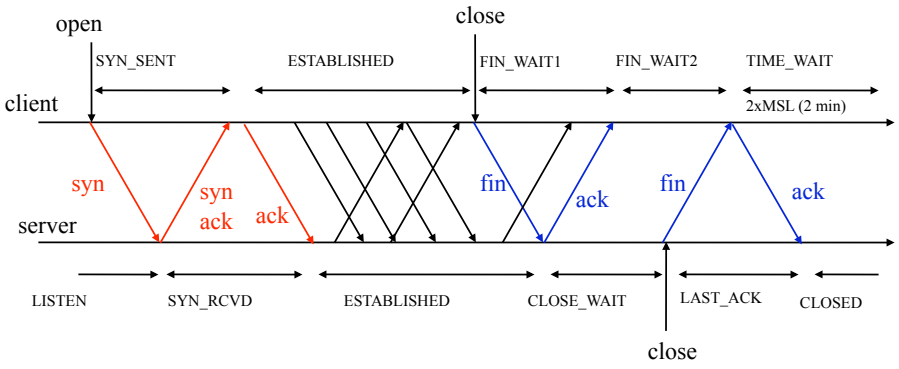
6. Grafo de estados en una conexión TCP

diagrama completo



6. Grafo de estados en una conexión TCP

- ejemplo conexión completa:



En el mismo escenario, ejecutamos el comando netstat -a en el servidor S y obtenemos la siguiente salida:

Local Adress	Remote Adress	State
*.111	*.*	LISTEN
*.21	*.*	LISTEN
147.83.30.10.3045	147.83.30.33.60	ESTABLISHED
147.83.30.10.111	147.83.30.20.3200	SYN_RCVD
127.0.0.1.111	127.0.0.1.3200	ESTABLISHED
147.83.30.10.21	147.83.30.20.2000	FIN_WAIT1

a) ¿Cuántos servidores tenemos activos en S?

c) ¿Por qué en la columna “Local Address” encontramos dos direcciones IP distintas?

a) ¿Qué secuencia de intercambio de segmentos ha ocurrido probablemente en la conexión en estado SYN_RCVD?

b) ¿Y en la conexión en estado FIN_WAIT1?

c) ¿Qué esperas que suceda en los próximos segundos en las dos anteriores onexiones?

Local Adress	Remote Adress	State
*.111	*.*	LISTEN
*.21	*.*	LISTEN
147.83.30.10.3045	147.83.30.33.60	ESTABLISHED
147.83.30.10.111	147.83.30.20.3200	SYN_RCVD
127.0.0.1.111	127.0.0.1.3200	ESTABLISHED
147.83.30.10.21	147.83.30.20.2000	FIN_WAIT1

- Recordemos que TCP proporciona fiabilidad mediante

  I. el control de flujo (**ventana advertida**)

  II. control de errores (**ARQ**)

  III. y control de la congestión (**ventana de congestión**)

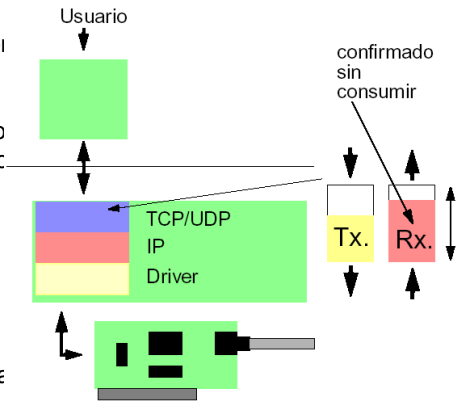
7. Control de flujo

"vigilar que el emisor no sature al receptor"

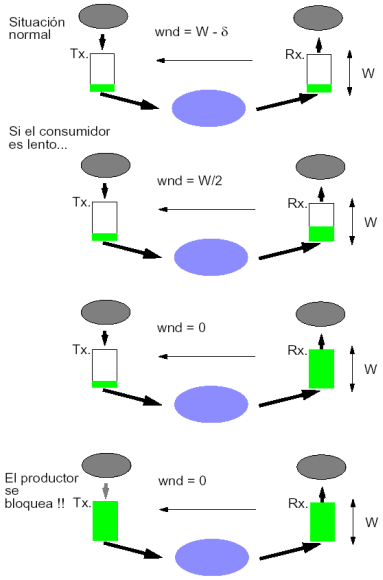
- Objetivo: el emisor no desborde el buffer del receptor por transmitir demasiado
  - el receptor tiene un buffer de recepción
  - la aplicación del receptor puede leer lento.
- Funcionamiento:
  - El receptor anuncia el espacio disponible en el buffer al emisor (con el campo **window size**)
  - El emisor limita los envios

7. Control de flujo

- El transmisor guarda en un buffer los bytes enviados pendientes de confirmación (por si se da el caso de que debe retransmitirlos) así comolos bytes que no se han transmitido (por ejemplo, por haber agotadc la ventana)
- El receptor guarda en un buffer los segmentos recibidos pendientes de ser consumidos por la aplicación. El tamaño de este buffer determina la ventana anunciada por el receptor



7. Control de flujo

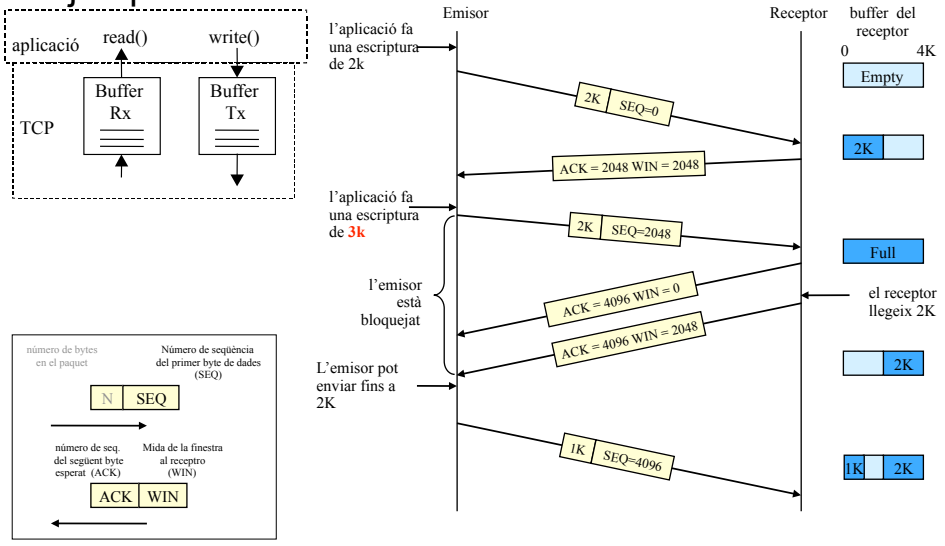


Casos que se pueden dar:

- Si el receptor es más rápido que el emisor, en este caso estara esperando contiamente datos y al ventana podría llegar a ocupar todo el buffer.
- El receptor es igual de rápido que el emisor, en este caso se van transmitiendo datos y el bufer del receptor tiene datos no consumidos y datos que se han recibido del emisor.
- El receptor es más lento que el emisor, en este caso la ventana advertida se ira reduciendo hasta que bloquee el emisor ya que no podrá enviar más datos.

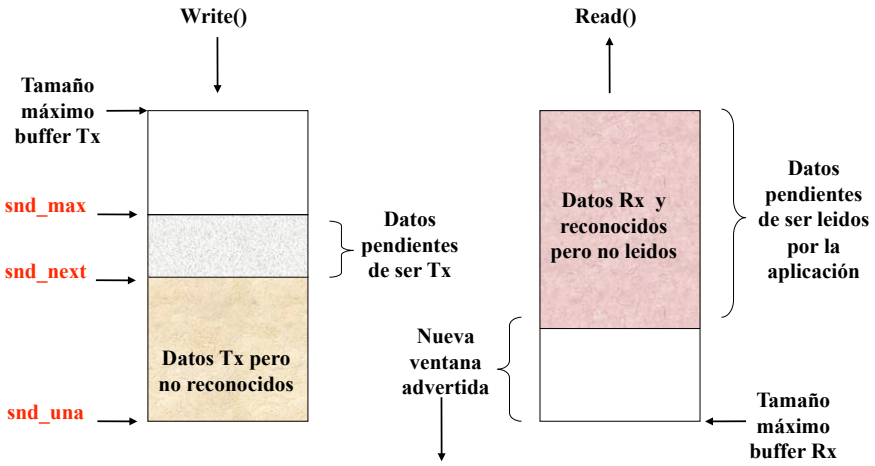
7. Control de flujo

Ejemplo:



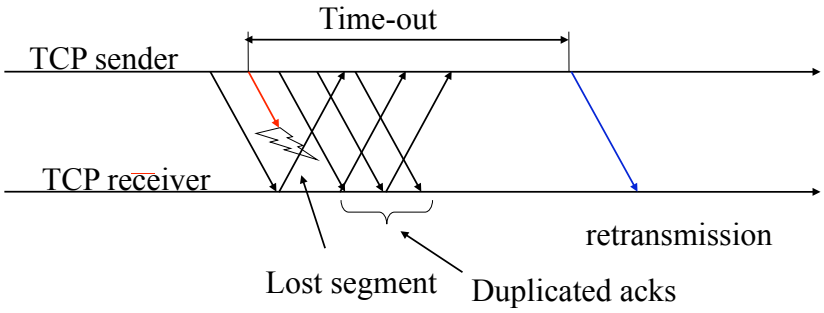
7. Control de flujo

- ¿Tamaño de la nueva ventana advertida?



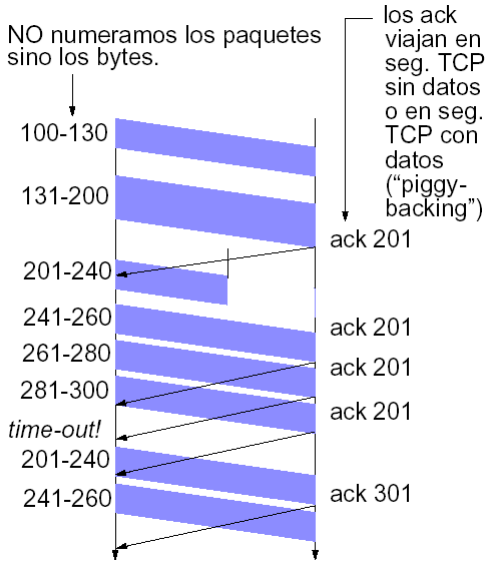
8. Control de errores

- ¿Qué pasa cuando se pierde un paquete?
- TCP consigue transmisión fiable de información mediante el uso de un protocolo de control de errores.



8. Control de errores

- ej:



8. Control de errores

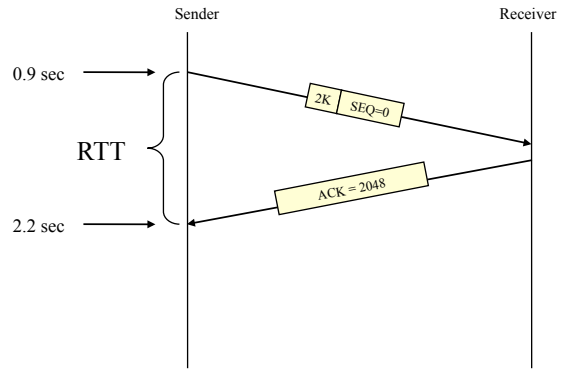
- Si un **segmento se pierde** y el emisor no recibe su ACK, los siguientes segmentos se van enviando y el receptor los reconocera con el ACK del anterior al que no se ha recibido, pero el emisor necesita el ACK del segmento perdido. Cuando tengamos el **timeout** (**en el emisor**) de dicho segmento lo retransmitira generando en el receptor un ACK del último paquete que llego.
- Si un segmento llega al receptor, pero no se recibe su ACK en el emisor, pero si el ACK del siguiente segmento, la recepcion del mismo **implica la recepción de todos los anteriores**.

8. Control de errores

- Retransmisión de los paquetes:
  - Cuando un paquete no recibe su ACK dentro de un periodo de tiempo, TCP asume que se ha perdido y vuelve a retransmitirlo.
  - TCP intenta calcular el “round trip time” (RTT) para un paquete y su ACK.
  - A partir de RTT, TCP puede suponer lo que debe esperar.
  - El cálculo del RTT no forma parte de las especificaciones de TCP!
- ¿Como fijar los valores de time-out?
  - Dependiendo de los retardos que sufren los ACKs, vamos fijando el valor del temporizador de forma adaptativa.

8. Control de errores

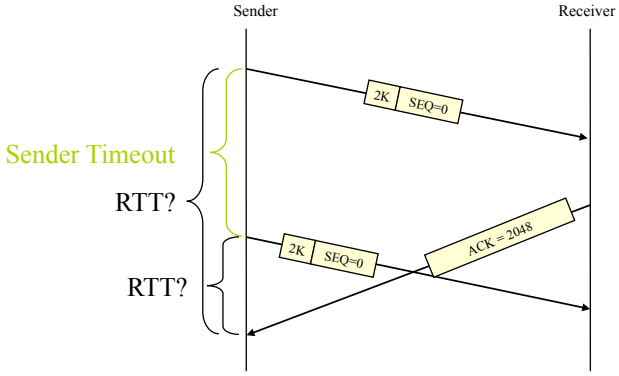
- Ejemplo de cálculo del RTT



RTT = 2.2 sec - 0.9 sec. = 1.3 sec

8. Control de errores

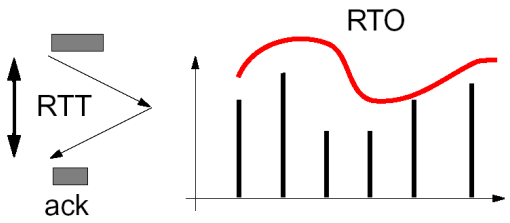
- Problemas con el cálculo del RTT



En este caso no se debe actualizar el RTT basado en información obtenida de paquetes retransmitidos (idea del Karn's Algorithm).

8. Control de errores

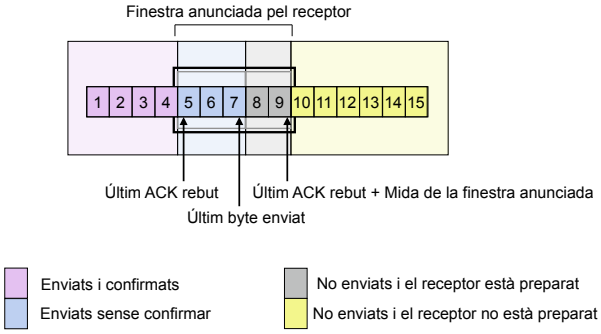
- Otro problema es que el RTT puede fluctuar mucho
  - » recordemos que es conexión extremo-extremo y que en medio puede haber cualquier tipo de red → los tiempos no son fijos.
- ¿Como fijar los valores de time-out?
  - Dependiendo de los retardos que sufren los ACKs, vamos fijando el valor del temporizador **de forma adaptativa**.



RTT: Round Trip-Time
RTO : Retransmission Time-Out

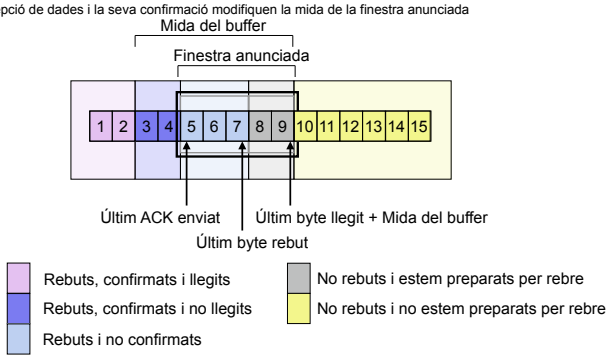
Finestra Lliscant - Emisor

- Tipus
  - Enviats i confirmats
  - Enviats, però no confirmats
  - No enviats i el receptor està preparat
  - No enviats i el receptor no està preparat
- Funcionament
  - Els ACKs (confirmacions) fan "lliscar" la finestra
  - La finestra anunciada pel receptor decideix la mida

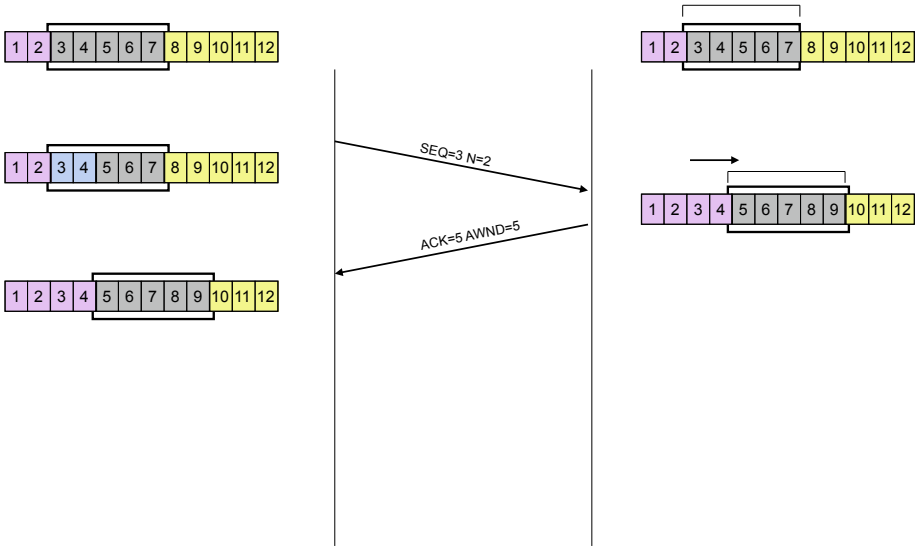


Finestra Lliscant - Receptor

- Tipus
  - Rebuts, confirmats i llegits per l'aplicació
  - Rebuts, confirmats i no llegits per l'aplicació
  - Rebuts i no confirmats
  - No rebuts i estem preparats per rebre
  - No rebuts i no estem preparats per rebre
- Funcionament
  - Les lectures per part de l'aplicació fan "lliscar" la finestra
  - La recepció de dades i la seva confirmació modifiquen la mida de la finestra anunciada

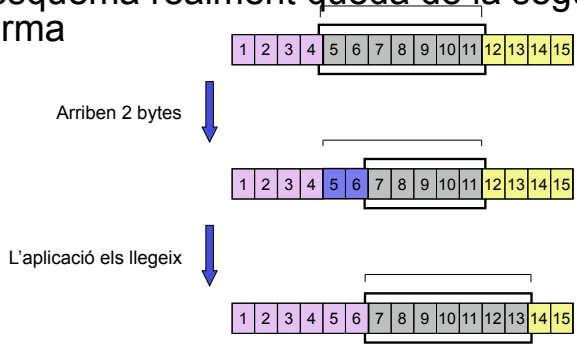


l'aplicació llegeix immediatament

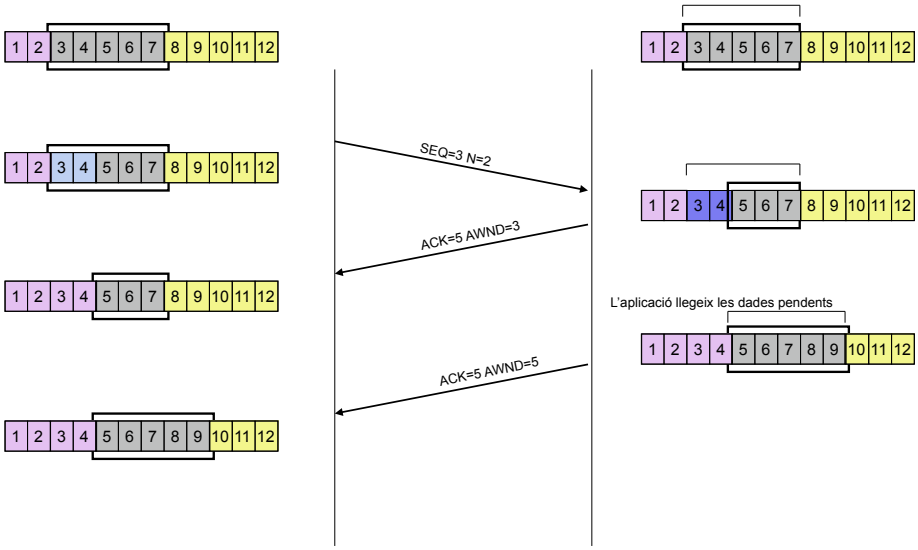


Finestra Lliscant - Receptor

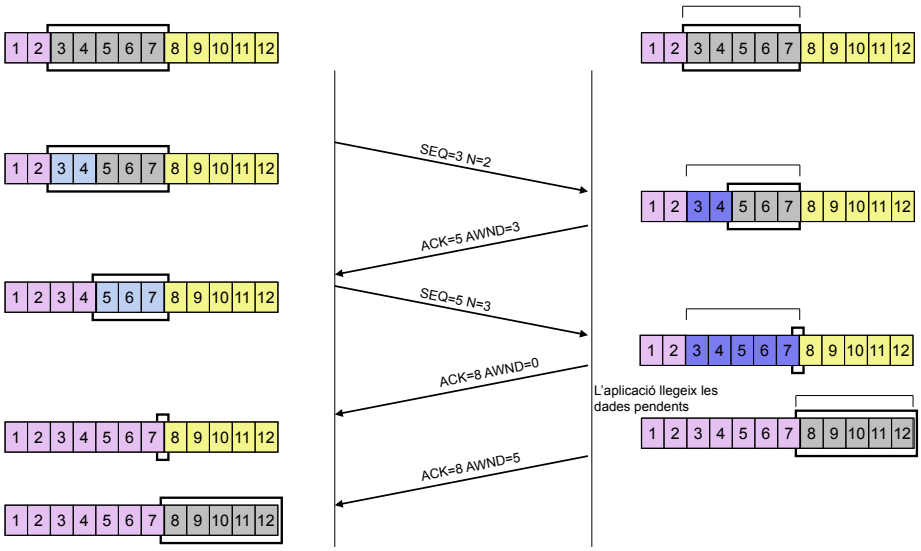
- Comentaris
  - Si l'aplicació llegeix de forma immediata l'esquema realment queda de la següent forma



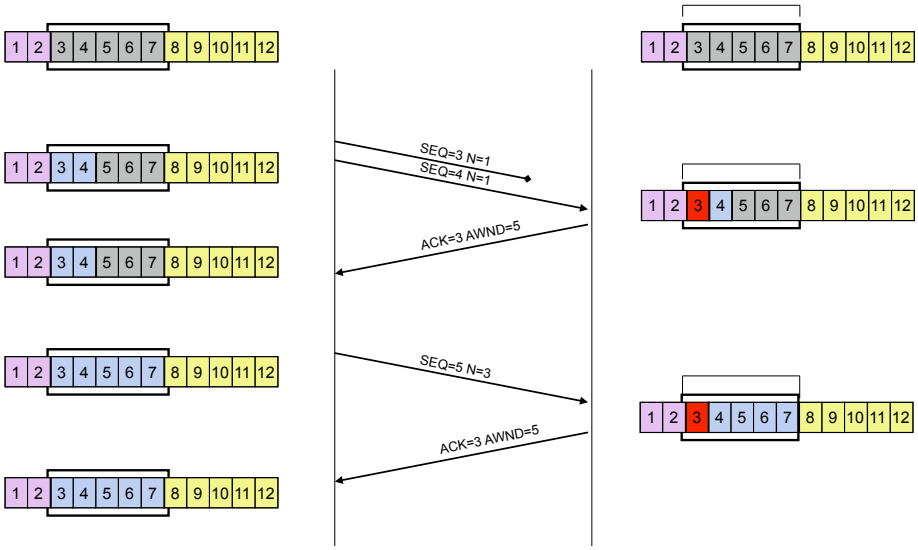
l'aplicació no llegeix - 1



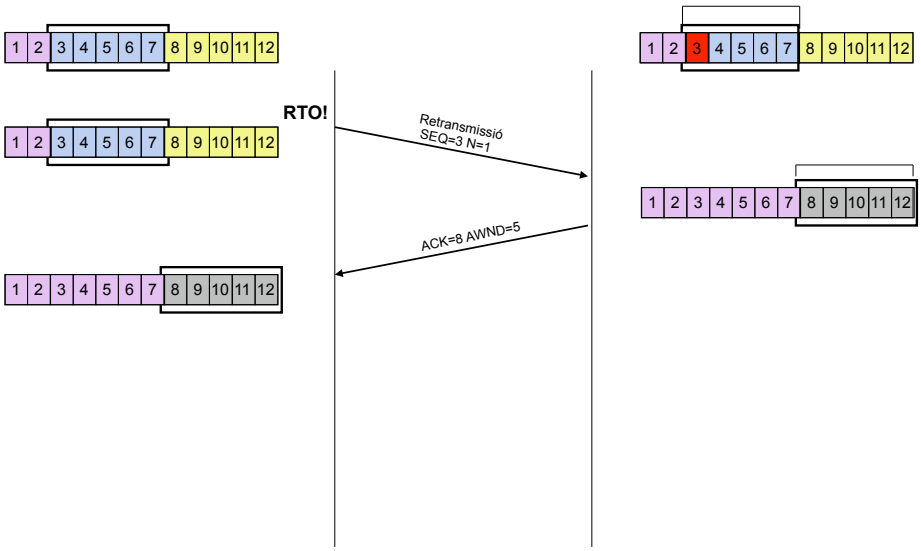
l'aplicació no llegeix - 2



es perd un segment



es perd un segment – continuació



Contenido T5

1. Introducción
2. UDP (User Datagrama Protocol)
3. TCP (Transmission Control Protocol)
4. Cabecera TCP
5. Establecimiento (3wHS) y cierre de la conexión TCP
6. Grafo de estados TCP
7. Control de flujo en TCP
8. Control de errores en TCP
9. Tipos de aplicaciones que usan TCP
10. Control de congestión en TCP
11. Sockets

9.Tipos de aplicaciones que usan TCP

- **Interactive aplications**: aquellas que envían poca cantidad de datos
  - ej: rlogin, telnet
  - Transmiten segmentos de tamaño muy pequeño (90 % de los segmentos tienen menos de 10 bytes de datos)
  - No es importante el control de la congestión
  - Importante los algoritmos de Nagle y los “Delayed ACKs”

9.Tipos de aplicaciones que usan TCP

Named for its creator, John Nagle, the **Nagle algorithm** is used to automatically concatenate a number of small buffer messages; this process (called nagling) **increases the efficiency of a network application system by decreasing the number of packets that must be sent.** Nagle's algorithm, defined in 1984 as Ford Aerospace and Communications Corporation Congestion Control in IP/TCP Internetworks (IETF RFC 896) was originally designed to relieve congestion for a private TCP/IP network operated by Ford, but has since been broadly deployed

9.Tipos de aplicaciones que usan TCP

Delayed ACKs

TCP has a rule like the following: If you send me two packets, I will send you one acknowledgement (ACK). If you send me one packet, I will wait 100 ms but not more than 200 ms before I respond with an ACK. The reason for delay is the chance a second packet might be coming in which case I should wait before I respond with an ACK.

This rule reduces the number of unnecessary ACKs. However, it causes relatively large delays in special cases, particularly for chatty applications that run across LANs. It can also be substantial for applications that run across WANs.

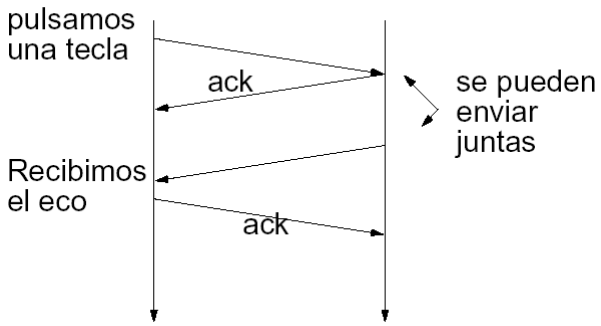
9.Tipos de aplicaciones que usan TCP

– **Bulk Transfer**: aplicaciones que envían gran cantidad de datos
  - ej: FTP
  - Los segmentos llevan mas de 512 bytes de datos
  - Importante los algoritmos de control de la congestión (Slow Start, Congestion Avoidance, Fast Retransmit y Fast Recovery)

! Que en las aplicaciones interactivas no sea importante el control de la congestión, no significa que no lo implementen, sólo que no se ven afectadas por este control de la congestión.

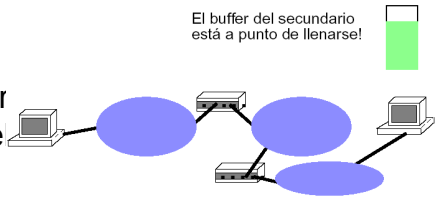
9.Tipos de aplicaciones que usan TCP

- Piggybacking

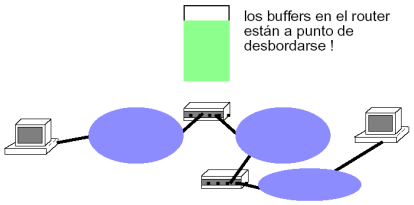


10. Control de congestión

- El algoritmo de ventana permite al receptor ejercer un control de flujo sobre e emisor



- Sin embargo los problemas podrían estar en la red (ej: router)



→ Necesitamos control de congestión

10. Control de congestión

- En las “Bulk Data Transfer”:
  - hace falta un control de la congestión debido a que enviamos muchos datos de golpe y los buffers pueden desbordarse (ya sea de los hosts o de los routers)
  - Mecanismo de control de la congestión:
    - Slow Start,
    - Congestion Avoidance,
    - Fast Retransmit,
    - Fast Recovery

10. Control de congestión

- Implementaciones TCP: muy variadas

  Todas están obligadas a implementar Slow Start y Congestion Avoidance

  - TCP Tahoe (1988): slow start, congestion avoidance, Fast Retransmit
  - TCP Reno (1990): Tahoe + Fast recovery + TCP header prediction
  - TCP Sack: Reno+ Selective ACK
  - Otras:
    - multicasting,
    - routing tables,
    - ....

10. Control de congestión

- Slow Start (SS) y Congestion Avoidance (CA):
  (Funcionan conjuntamente)

  – Queremos "arrancar" la conexión de forma que averigüemos a qué velocidad podemos transmitir los datos.

  – Introducimos una nueva ventana: **`cwnd`** (ventana de congestión,"congestion window").

  - Ventana advertida: Control de flujo ejercido por el receptor.
  - Ventana de congestión: Control de congestión ejercido por el emisor.

  → El valor de la ventana que se aplica és el mínimo de ambos valores

10. Control de congestión

- Slow Start (Funcionamiento):
  - Inyectar segmentos en la red a la velocidad a la que recibimos los ACKs desde el otro extremo
  - Ventana advertida (snd_wnd): ventana advertida por el receptor
    - *Es un control de flujo impuesto por el receptor que es el que fija el valor de la ventana advertida*
    - *S'explicita en un camp dels segments TCP que s'envien*
  - Ventana de congestión (snd_cwnd): ventana que se inicializa a 1
    - *Es un control de flujo impuesto por el transmisor*
    - *És un comptador intern que no es mostra explícitament*
  - Cada vez que se recibe un ACK → snd_cwnd++ (se incrementa en 1).

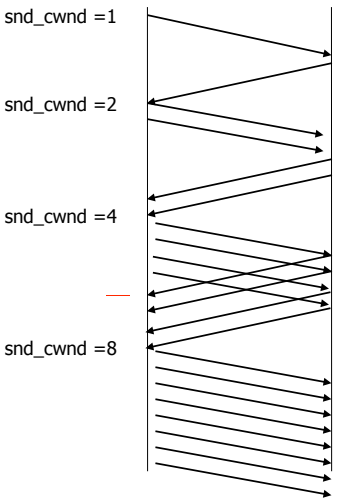
El transmisor transmite:

Ventana de TX = min (snd_wnd, snd_cwnd)

10. Control de congestión

- Slow Start (SS):

Cada vez que recibimos un ack, incrementamos `cwnd` en 1.

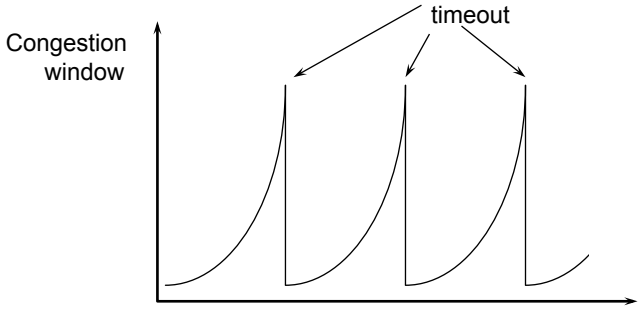


Inicialmente { snd_wnd =10; snd_cwnd =1 }

Min (snd_wnd, snd_cwnd)

10. Control de congestión

- El incremento de segmentos transmitidos es exponencial
- Perdida de paquetes indica congestión (se sabe por timers en el que envia)
- Cuando ocurre un timeout, la ventada de congestión se reduce a 1 (todo empieza de nuevo!)



10. Control de congestión

- TCP Slow Start, solo, es claramente ineficiente.

  la ventana de congestión crece exponencial, pero, cae al tamaño de un segmento

  →esto claramente conlleva un *throughput* pobre

- Solución:

  Establecer un umbral a partir del cual se incrementa linealmente en lugar de exponencialmente para aumentar la eficiencia. → Congestion Avoidance

10. Control de congestión

- Slow Start y Congestion Avoidance:
  (Funcionan conjuntamente)

  – Congestion Avoidance
    - Con SS vamos aumentando la ventana de congestión hasta que llega un momento en que llenamos el buffer de algún router y se producen pérdidas.
    - Con Congestion Avoidance (CA) intentamos mantenernos al valor límite de throughput sin tener pérdidas.

10. Control de congestión

– Congestion Avoidance (CA) (Funcionamiento):
  - Si comenzamos con Slow Start, enviamos segmentos de forma exponencial
  - Si la red se congestiona queremos que la red se recupere (congestion avoidance) sin dejar de transmitir y comenzar de nuevo en un slow start
  - Es un control de flujo impuesto por el transmisor
  - CA hace que cuando se detecta una perdida, en vez de transmitir exponencialmente segmentos, pasemos a transmitir de forma lineal hasta que se recupere la red

10. Control de congestión

- Algoritmo de “Congestion Avoidance”:

  `cwnd` = 1; `ssthresh` = Ventana máxima

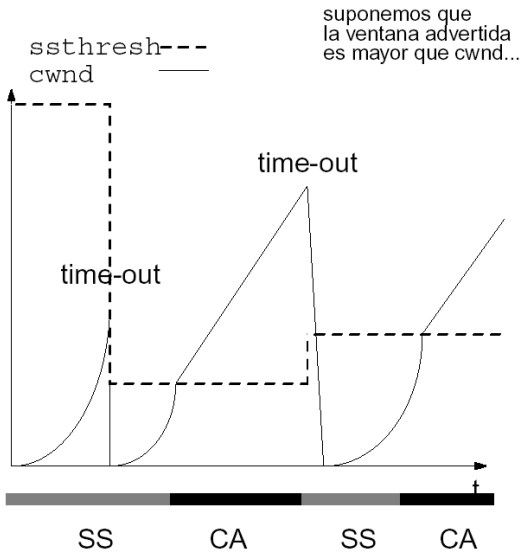
  Con SS vamos incrementando `cwnd` hasta que hay una pérdida (time-out)

  `ssthresh` <- max(2,1/2 min(`cwnd`, `awnd`))
  `cwnd` <- 1 e iniciamos SS

  Cuando `cwnd` >= `ssthresh`, abandonamos la fase de SS e iniciamos la fase de CA.

  Durante la fase de CA, cada vez que recibimos un ack, incrementamos `cwnd` <- `cwnd` + 1/`cwnd` (es decir, `cwnd` crece linealmente en el tiempo)

10. Control de congestión

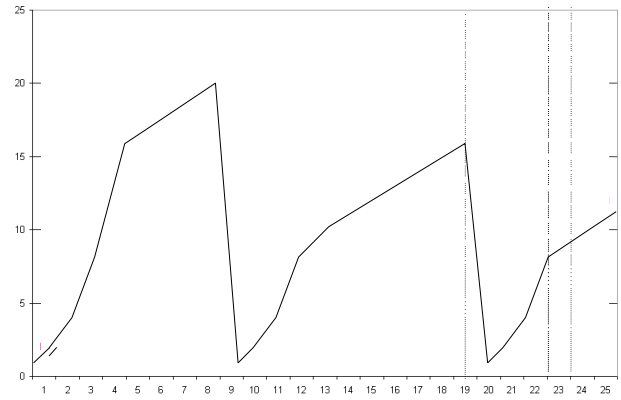


10. Control de congestión

- Congestion Avoidance (CA) (más detallado):
  - Define un nuevo parámetro:
    - Threshold (umbral) (ssthresh) que se inicializa a 65535 bytes
    - Si hay congestión:
      - Debido a una perdida (Tout) o a la recepción de 3 ACKs duplicados → el threshold pasa a valer la mitad de la ventana de transmisión pero no por debajo de 2 segmentos, es decir:
        - » ssthresh=max[2, 1/2 min(snd_wnd, snd_cwnd)]
      - Si además la congestión es por el salto del temporizador, snd_**c**wnd=1
    - Si snd_cwnd < ssthresh → estamos en Slow Start (SS)
      - Cada vez que recibamos un ACK, incrementamos la ventana de congestión de la siguiente manera
        - » Si estamos en SS → snd_cwnd ++
    - Si snd_cwnd ≥ ssthresh → estamos en Congestion Avoidance (CA)
      - Cada vez que recibamos un ACK, incrementamos la ventana de congestión de la siguiente manera
        - » Si estamos en CA → snd_cwnd = snd_cwnd + 1/ snd_cwnd

10. Control de congestió

- Suposeu la següent gràfica de la mida de la finestra de congestió d'una connexió TCP



10. Control de congestió

- Preguntes breus prèvies:
  - Què indiquen les unitats de l'eix de les X? Quina unitat s'està fent servir?
  - Quin és el valor del *Threshold* en el punt 22? i en el 23?

10. Control de congestió

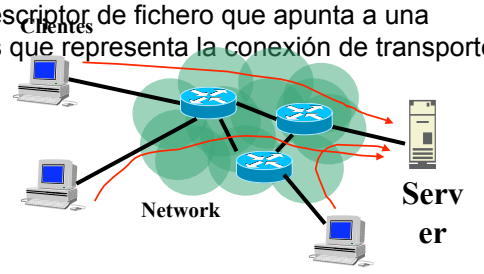
- Suposeu que:
  - La finestra advertica/anunciada és molt gran.
  - Que el receptor i emisor són molt ràpids (suposa infinit). Això vol dir que el temps de enviar tots els segments d'una finestra i rebre els ACKs corresponents, és just un RTT.
  - Suposa que el segment conté 1000 bytes de dades, que el *RTT* és 100 ms (1 ms = 10^{-3} segons).
- Quina és la velocitat (throughput mig) amb la que envia les dades el emisor entre els punts 19 i 25 (inclosos)?

10. Control de congestió

In TCP/IP, **fast retransmit and recovery** (FRR) is a congestion control algorithm that makes it possible to quickly recover lost data packets. Without FRR, the TCP uses a timer that requires a retransmission timeout if a packet is lost. No new or duplicate packets can be sent during the timeout period. With FRR, if a receiver receives a data segment that is out of order, it immediately sends a duplicate acknowledgement to the sender. If the sender receives three duplicate acknowledgements, it assumes that the data segment indicated by the acknowledgements is lost and immediately retransmits the lost segment. With FRR, time is not lost waiting for a timeout in order for retransmission to begin.

11. Sockets (repaso)

- Modelo cliente-servidor
  - Los servidores están esperando peticiones de los clientes
  - Cuando un cliente quiere un servicio (aplicación) debe establecer una conexión con el servidor. A la conexión se le llama en UNIX "socket"
  - Un socket es un descriptor de fichero que apunta a una estructura de datos que representa la conexión de transporte
    - Sockets TCP
    - Sockets UDP

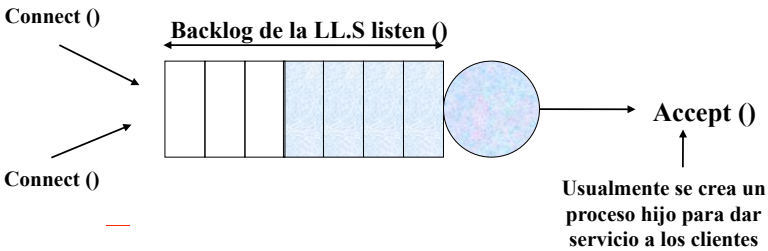


11. Sockets (repaso)

- Puertos TCP/UDP
  - Un puerto identifica la aplicación de red que deseamos utilizar
    - Puertos conocidos ("Well-known ports"): puertos que identifican la aplicación servidor (rango 1-1023), ver /etc/services. Servicios no estándares usan ports > 5000
      - 20/tcp ftp-data
      - 21/tcp ftp
      - 23/tcp telnet
      - 69/udp tftp
      - 80/tcp http
      - .....
    - Puertos efímeros ("ephimeral ports"): puertos usados por los clientes (típicamente entre 1024 y 5000)

11. Sockets

- Relación entre las LL.S y el diagrama de estados TCP

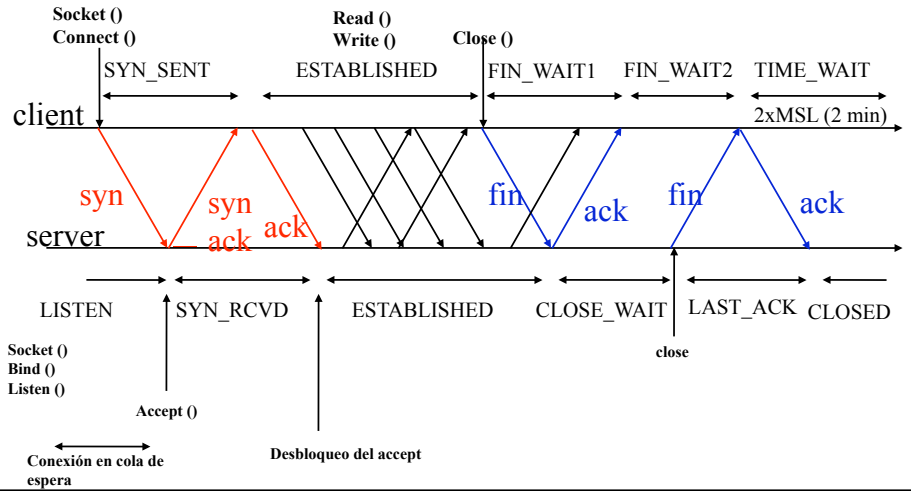


• Cola de conexiones incompletas: conexiones que no han terminado el 3WHS
• Cola de conexiones completas: conexiones que han terminado el 3WHS

Devuelve SYN-ACK, espera un ACK del cliente antes de dar servicio al proximo connect ()

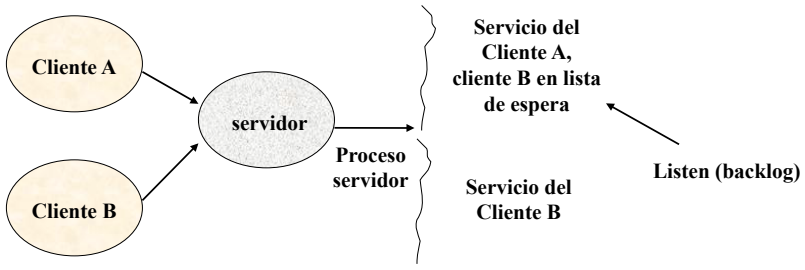
11. Sockets

- Relación entre las syscalls y el diagrama de estados TCP



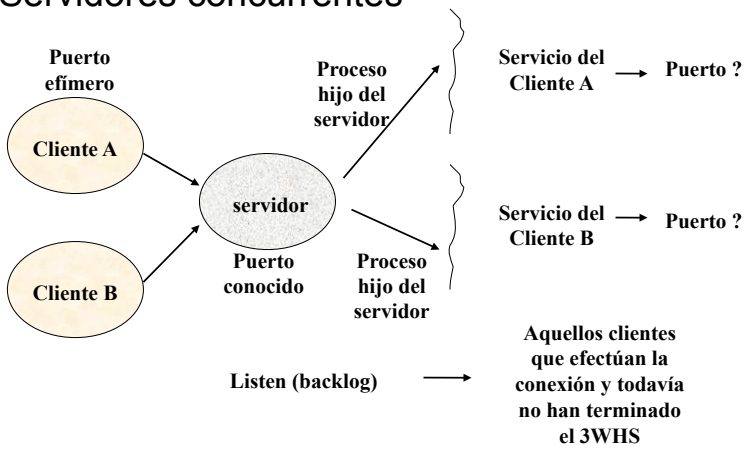
11. Sockets

- Tipos de servidores:
  - Servidores interactivos (iterativos)



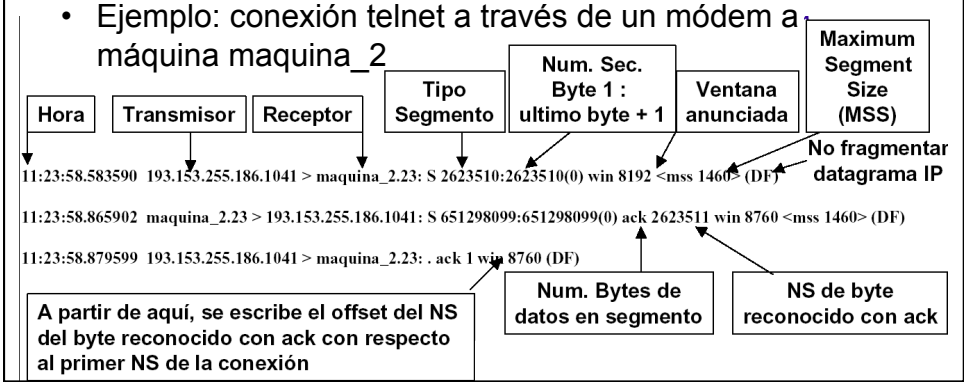
11. Sockets

- Tipos de servidores:
  - Servidores concurrentes



Annex: Using tcpdump

- Programa que coloca a la tarjeta de red en modo promíscuo: todo lo que pasa por la red es recogido y pasado a los módulos IP y TCP. Tcpdump presenta la información al usuario en forma inteligible.
- Ejemplo: conexión telnet a través de un módem a máquina maquina_2



Annex: tcpdump

- **tcpdump -w** filename.trace **-i** interface **-s 100**
  host sender **and host** receiver
- Tracea los paquetes en “interface” y los coloca en “filename.trace”
- Se pueden especificar reglas complejas para el filtrado (aquí sólo indicamos la pareja de hosts).
- **-s 100** especifica cuantos bytes por paquete.
  - 12 bytes for MAC headers
  - 20 bytes IP header
  - 20 bytes TCP header
  - extra room for any options which may appear
- Muchas más opciones disponibles!
- ^C para acabar!

T7
Nivell enllaç

Xarxes de Computadors i Aplicacions

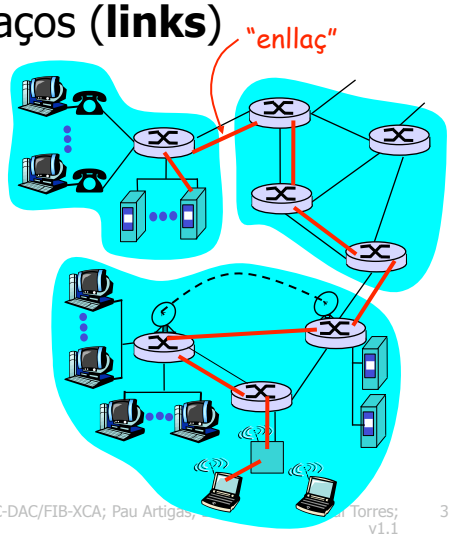
PAU ARTIGAS, DAVID CARRERA i JORDI TORRES
Departament d'Arquitectura de Computadors
UPC, setembre - 2009

Nivell enllaç

1. Introducció(tipus enllaç, ppp o broadcast)
2. Detecció i correcció d'errors
3. LANs
   - Accés a medis compartits
   - Topologies i cablejat
   - IEEE 802 (802.2, 802.3, 802.5)
   - Concentradors i commutadors
4. PPP

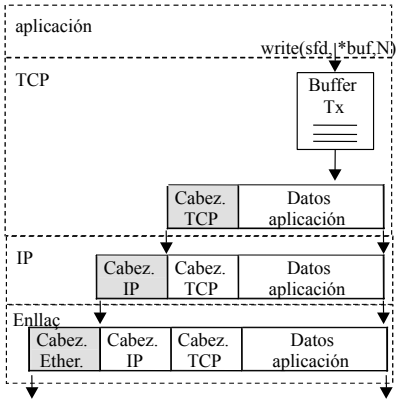
1. Introducció

- Els **nodes** (hosts o routers) estan connectats amb enllaços (**links**)
- El que es transporta s'anomena **trama**
- Tipus de canals:
  - **Broadcast**
  - **Punt a punt**



1. Introducció

- El nivell enllaç encapsula les dades del nivell xarxa
- En cada enllaç del camí d'un datagrama el nivell enllaç pot ser diferent
- Aquest nivell ofereix serveis al nivell de xarxa



1. Introducció

- Serveis que pot oferir el nivell enllaç:
  - **Entramat**: transportar les dades en una trama, amb una possible adreça origen i destí de nivell enllaç
  - **Accés al medi:** l'accés al canal de comunicació no sempre és exclusiu i pot requerir certa coordinació entre nodes
  - **Fiabilitat:** alguns medis que tendeixin a introduir errors poden afegir protocols de retransmissió per a no obligar sempre a fer-ho al nivell transport

1. Introducció

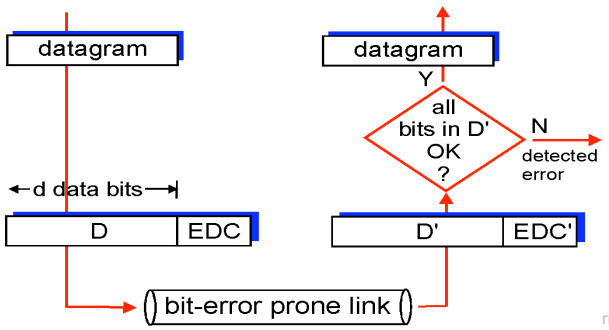
  - **Control de flux**: els nodes connectats per un enllaç no tenen buffers infinits i requereixen un control de la quantitat de dades enviades (finestra lliscant)
  - **Detecció d'errors:** a nivell físic es poden produir problemes de transmissió, que si és possible, s'han de detectar
  - **Correcció d'errors:** a part de detectar l'error es determina quin ha estat i es repara
  - **Full duplex/Half duplex:** Els dos extrems d'un enllaç poden emetre simultàniament en full duplex, però no en half duplex

1. Introducció

- El protocol usat per la interfície de xarxa es troba implementat en un adaptador (i el seu corresponent driver de sistema)
- Els adaptadors s'acostumen a anomenar Network Interface Cards, o NIC
- Quan rep dades realitza control errors, control de flux...

2. Detecció i correcció d'errors

- El nivell enllaç acostuma a realitzar tasques de control d'errors i correcció d'errors **a nivell de bit** en les trames que es reben
- No 100% fiable!!
- Funcionament basat en l'inclusió de bits extres



2. Detecció i correcció d'errors

- Tècniques habituals:
  - Control de paritat
  - Checksumming
  - CRC

2. Detecció i correcció d'errors

- Un cop detectat un error, com es resol?
  - Correcció d'errors ARQ (Automatic Repeat Request)
- Tipus de protocols:
  - Idle-RQ
    - Stop&Wait
  - Continuous-RQ
    - Selective Repeat
    - Go-back-N

2. Detecció i correcció d'errors

- Idle-RQ: el transmissor s'atura
  - **Stop&Wait**: El transmissor no envia una dada fins que no sap que el receptor ha rebut l'anterior correctament
- Continuous-RQ: el transmissor no s'atura
  - **Go-back-N**: Es retransmeten totes les dades a partir de la detecció de l'error i no es reaprofita el que ja s'havia enviat posteriorment
  - **Selective Repeat**: Es retransmeten trames individuals que contenien errors

3. LANs

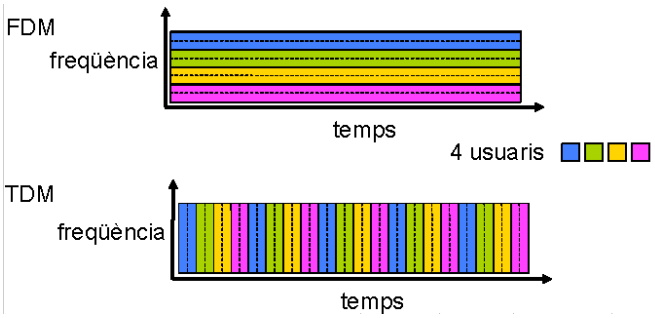
- Tipus de canals de comunicació:
  - Connexions punt a punt
  - Xarxes de difusió (broadcast)
- Les xarxes de difusió necessiten gestionar l'accés al medi compartit, per a evitar interferències:
  - Protocols **MAC**: Medium Access Control

3. LANs: Accés a medis compartits

- Tres possibles maneres de control:
  - **Particionament del canal**
    - Dividir el canal entre els "usuaris" (time slots, frequencia…) per a evitar colisions
  - **Accés aleatori**
    - Es permeten colisions, i s'implanten mecanismes de recuperació
  - **Per torns**
    - Coordinació entre els equips per a evitar colisions

3. LANs: Accés a medis compartits

- **Particionament del canal**
  - TDM (Time Division Multiplexing)
  - FDM (Frequency Division Multiplexing)



3. LANs: Accés a medis compartits

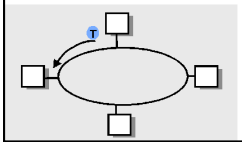
- **Accés aleatori**:
  - El protocol especifica
    - Com detectar colisions
    - Com recuperar-se de les colisions
- Exemples de protocols:
  - ALOHA
  - CSMA/CD

3. LANs: Accés a medis compartits

- CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
  - Accés múltiple amb detecció de portadora i detecció d'errors
  - El protocol primer "escolta" el medi abans de transmetre dades (si algú ja està transmetent, s'espera per a evitar colisions) (no és infalible!)
    - Si dos estacions comencen alhora en un medi on no hi ha dades, colisionaran
  - Si es detecta una colisió, l'estació transmissora avisa les altres estacions, s'espera una quantitat aleatòria de temps i torna a intentar la transmissió

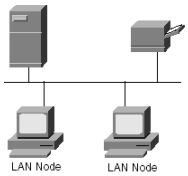
3. LANs: Accés a medis compartits

- **Per torns**
  - Dos tipus:
    - Polling
      - Una estació "master" invita a les estacions esclaves a transmetre (fa arbitratge)
      - Problema: un sol punt de fallada -> master
    - Pas de token
      - Les estacions es passen un "token" (relleu)
      - L'estació que disposa del token pot transmetre
      - Problema: un sol punt de fallada -> token

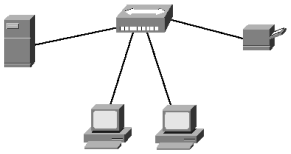


3. LANs: Topologies i cablejat

- Les LANs que usen cablejat (aquelles no inalàmbriques), es poden caracteritzar per la seva **topologia** (física o lògica poden ser diferents):
  - Bus:

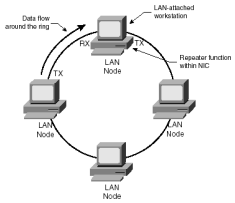


  - Estrella:

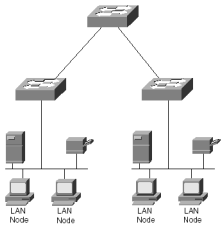


3. LANs: Topologies i cablejat

– Anell:



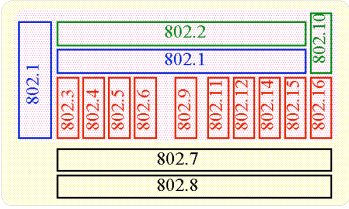
– Arbre



3. LANs: IEEE 802

- El conjunt d'estandards definits en una sèrie de RFCs anomenades conjuntament IEEE 802 defineixen un conjunt de protocols de nivell enllaç

- Aquest nivell es divideix en 2 subnivells:
  - El nivell d'enllaç lògic (LLC): 802.2
  - El nivell d'accés al medi (MAC):
    - 802.3, 802.4, 802.5 … 802.11

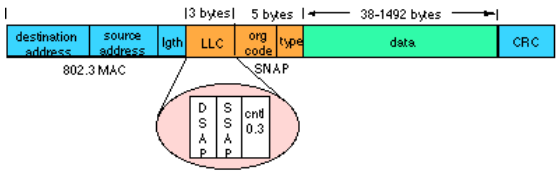


3. LANs: IEEE 802.3

- Nivell MAC 802.3 -> conegut típicament com a Ethernet
- En realitat, ethernet era un protocol anterior creat per Xerox, que després va ser ampliat per DEC i Intel i que va acabar sent la base del IEEE 802.3 (el més extès actualment)
- El protocol es basa en CMSA/CD operant a velocitats de 1 a 10 Mbps

3. LANs: IEEE 802.3

- Camps:
  - Adreces origen i destí
    - Adreces MAC (o físiques) del NIC / 48 bits per adreça
  - Longitut de la trama
    - Bytes en el camp de dades (de 0 a 1492)
  - Capçalera 802.2 (LLC)
  - Dades
  - CRC



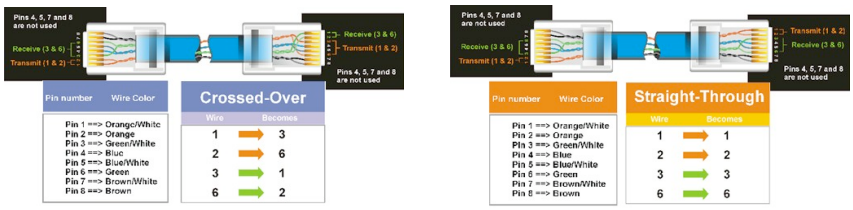
3. LANs: IEEE 802.3

Network	Unix IP SAP: 80	IBM Netbios SAP: F0	Novell IPX SAP: E0
Data Link	IEEE 802.2 Logical Link Control Layer (LLC)		
	IEEE 802.3 CSMA/CD Medium Access Control Layer		
Physical	802.3 - 10Base5	802.3a - 10Base2	802.3i - 10BaseT

- Cablejat (10 Mbps):
  - 10BaseT
    - Ethernet over **Twisted Pair** Media / Long màxima: 100 m
  - 10BaseF
    - Ethernet over **Fiber** Media / Long màxima: 2000 m
  - 10Base2
    - Ethernet over **Thin Coaxial** Media / Long màxima: 200 m
  - 10Base5
    - Ethernet over **Thick Coaxial** Media / Long màxima: 500 m

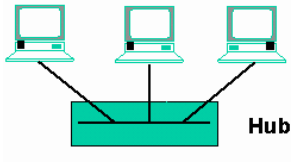
3. LANs: IEEE 802.3

- Els cables poden ser paral·lels o creuats



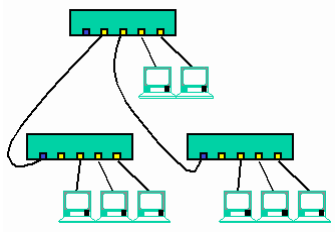
3. LANs: IEEE 802.3

- Concentradors:
  - Per tal d'unir diferents **segments** de tipus 10Base-T s'utilitza un concentrador (**hub**), que internament és un bus
  - Crea una topologia lògica de **bus**, però física d'**estrella**
  - Els equips es connecten utilitzant cables directes (no creuats). El hub fa el creuament internament
  - Creen un sol **domini de col·lisió** (el bus funciona en mode broadcast, tot s'envia a tothom)



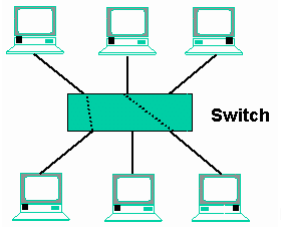
3. LANs: IEEE 802.3

  - Els hubs es poden connectar en cascada: existeix un port que s'anomena **uplink**! (no està creuat internament)



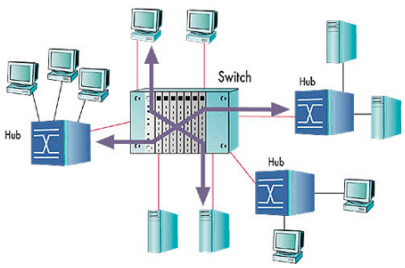
3. LANs: IEEE 802.3

- Commutadors:
  - Per tal d'evitar els problemes derivats de la creació de dominis de colisió molt grans, s'usen commutadors (**switch**)
  - Cada segment connectat a un switch és un domini de colisió independent
  - El switch no reenvia per tots els ports les dades que rep per un d'ells (no fa broadcasting)



3. LANs: IEEE 802.3

- Concetradors i commutadors es poden combinar (tots els equips interconnectats amb un hub formen un domini de colisió independent)
- Poden funcionar en mode full-duplex

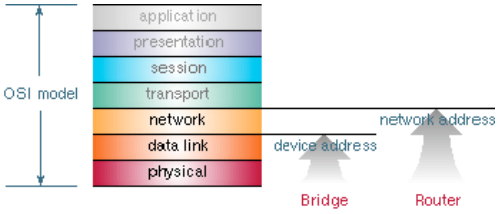


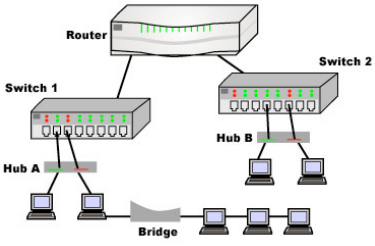
3. LANs: IEEE 802.3

- Un switch coneix els equips que té connectats
  - Emmagatzema les adreces MAC dels equips que té connectats
- Només reenvia pel port que toca

3. LANs: Enllaç entre protocols

- Es pot fer utilitzant un bridge o un router
  - Bridge: Nivell 2
  - Router: Nivell 3
- Ajuda a reduir tr





4. PPP: Point to Point Protocol

- Protocol per a transportar datagrames multi-protocol a través d'enllaços punt a punt (RFC 1661)

- Tres funcions principals:
  - Encapsulament
  - Gestió de l'enllaç
    - Link Control Protocol (LCP)
  - Cooperació amb el nivell de xarxa
    - Network Control Protocol (NCP)

4. PPP: Point to Point Protocol

- PPP Link-Control Protocol
  - Mecanismes de control de la connexió
    - establishing
    - configuring
    - maintaining
    - terminating