TEMA 1- INTRODUCCIÓ

Model OSI de ISO

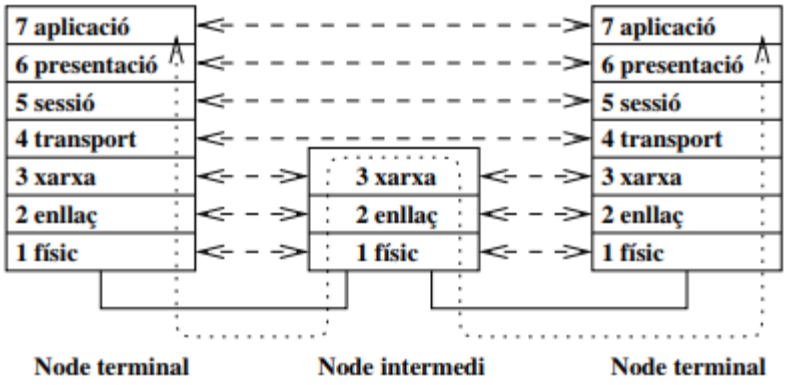
Divideix el conjunt de protocols que formen part d'una xarxa de computadors en **7 nivells**;

Els **nodes terminals** representen els dos sistemes que es comuniquen a través de la xarxa (p.e: **client** i servidor).

Els **nodes intermedis** tenen la funció d'encaminar l'informació (p.e: els **routers**).

Cada nivell utilitza una estructura de dades anomenada ***Protocol Data Unit*** **(PDU)** per a intercanviar informació amb el seu nivell parell. Que conte un **header i un payload**.

Cada nivell estableix un diàleg al corresponent amb les línies discontínues.



- Nivell 1 - **Físic**: Defineix les **característiques físiques i elèctriques** d'un dispositiu de xarxa. En aquest nivell només es parla de **bits o caràcters**.

- Nivell 2 - **Enllaç** *(Data link)*: Fa **d'interfície entre el nivell de xarxa i el físic**. Les PDUs que utilitza s'anomenen **"trames" *(frames)*.** Entre les seves funcions hi pot haver:
    - ***Framing***: Per a descobrir on comença i on acaba una trama dintre del flux de bits o caràcters que llegeix del nivell físic.
    - **Detecció d'errors**: Per a detectar si la trama rebuda té algun error.
    - **Recuperació d'errors**: per exemple, demanant la retransmissió d'una trama errònia al seu nivell parell.

- Nivell 3 - **Xarxa** *(Network)*: Les PDUs que fa servir s'anomenen **"paquets"** (**datagrames per Internet**). La seva principal funció és **l'encaminament de PDUs**: És a dir, aconseguir que les PDUs segueixin **el camí correcte entre la font i la destinació**.

- Nivell 4 - **Transport**: Les PDUs utilitzades es diuen **"*segments*"** (TCP) i ***datagrames*** en (UDP), . És un nivell extrem-a-extrem. La seva funció és establir **un canal entre les dues aplicacions que es comuniquen**. Pot oferir serveis de **segmentació o recuperació d'errors**.

- Nivell 5 - **Sessió**: La seva funció és el ***login***, permetre tornar a un punt conegut després d'un tall de connexió, etc.

- Nivell 6 - **Presentació**: Proporciona protocols de presentació de dades (**ASCI, MPEG**, etc.)

- Nivell 7 - **Aplicació**: Són els protocols de les aplicacions que fan servir la xarxa. Per exemple: **http, smtp, ftp, telnet**, etc.

Arquitectura TCP/IP

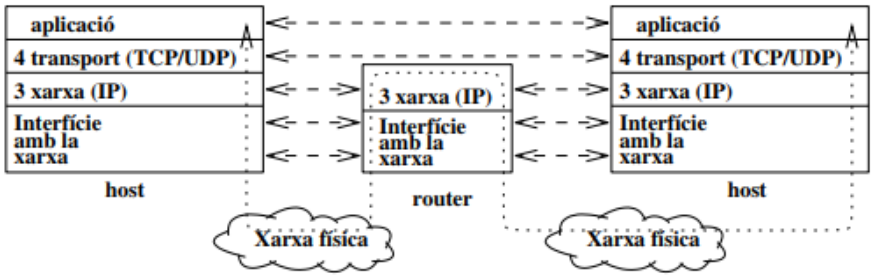


Figura 1.5: Arquitectura de TCP/IP.

Els **protocols més importants** d'Internet es corresponen amb el nivell de **xarxa: el *Internet Protocol* (IP) i** el nivell de **transport: El *Transmission Control Protocol* (TCP)**.

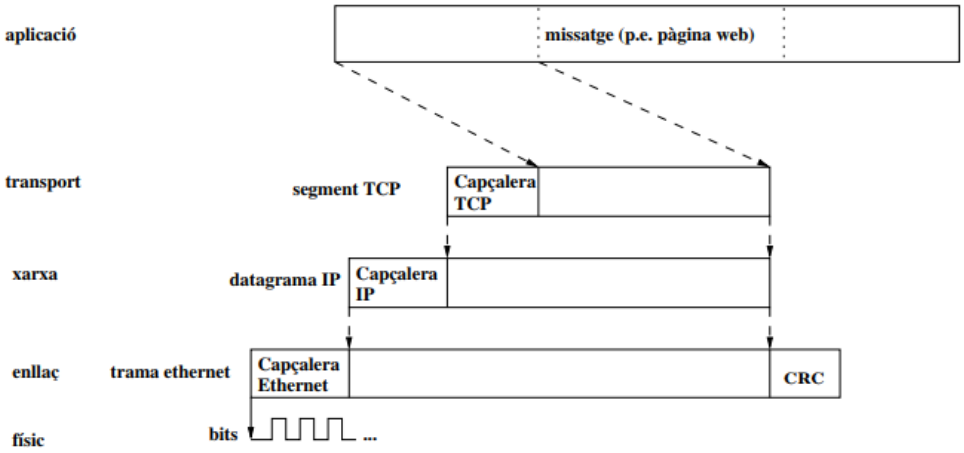
Per ser més exactes, pel que fa al **transport** hi ha **dos** protocols:

- **TCP,** que implementa una **transmissió fiable** (assegura una transmissió lliure d'errors),
- ***User Datagram Protocol* (UDP)**, que no assegura el **lliurament correcte de l'informació**.

Encapsulament de la informació

En la transmissió d'informació a través de la xarxa, **cada nivell** afegeix **una capçalera** amb **l'infromació necessària** per a comunicar-se amb **el nivell parell**.

Cada nivell **afegeix una capçalera** abans de passar **la PDU al nivell inferior** i **elimina la capçalera** abans de passar **la PDU al nivell superior**. Aquest procés s'anomena **encapsulament**:



Paradigma Client-Servidor

És el **model utilitzat per a les aplicacións a Internet per intercanviar-se informació**.

L'aplicació que **inicia l'intercanvi** d'informació s'anomena "**client**". Així doncs, el client envia el primer datagrama cap al servidor. Tal com hem vist, **el nivell de xarxa (IP)** porta els datagrames **fins a la destinació**. Un cop a la destinació**, cal lliurar l'informació a l'aplicació on van dirigits** (p.e: un servidor web). El responsable d'aquesta tasca és el nivell de transport (TCP o UDP).

Per poder **identificar les aplicacions** que es comuniquen, TCP/UDP fan servir els anomenats "**ports**": Un port identifica l'aplicació client, i l'altra aplicació servidor. Aquests ports **formen part de la capçalera de protocol TCP o UDP**.

Aquest servidor té associat un **port *well-known***. Cada port *well-known* **identifica un servei estàndard d'internet**, com ara ***ftp (port 21), telnet (port 23), web (port 80)***, etc. Els ports *well-known* tenen un valor en l'interval **[0, ..., 1023**] i estan estandarditzats per un organisme d'Internet anomenat **IANA**.

Quan el **client** inicia l'intercanvi d'informació, el SO li assigna un **port "efímer"** (que té un **valor >= 1024**). Aquest port identifica el client mentre dura l'intercanvi d'informació. Així doncs, per identificar completament una connexió en TCP/IP es fa servir la **tupla**: (port font, adreça IP font; port destinació, adreça IP destinació). Les **adreces IP** identifiquen **els hosts,** **els ports** identifiquen **les aplicacions dels hosts**.

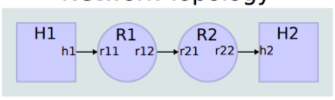
Example

- Discussion: Just Ethernet header contents are modified over links (hops)
- Description
  - Host 1 is connected to Router 1, Router 1 to Router 2, and Router 2 to Host2
  - Host 1 sends a test packet (ICMP Echo Request) to Host 2 over the network
  - Host 2 replies with 1 packet (ICMP Echo Reply)
  - Physical network interfaces identifiers (Ethernet addresses)
    - Host 1: h1
    - Router 1: r11 and r12
    - Router 2: r21 and r22
    - Host 2: hi
  - Network identifiers (IP addresses)
    - Host 1: H1
    - Host 2: H2
- Question
  - Describe the evolution of the Ethernet headers and IP headers of both packets

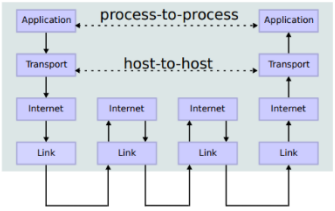
Example (cont.)

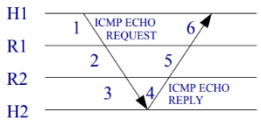
- Solution

Network Topology



Data Flow





Link	Packet	Header			
		Ethernet		IP	
		src	dst	src	dst
H1-R1	1	h1	r11	H1	H2
R1-R2	2	r12	r21	H1	H2
R2-H2	3	r22	h2	H1	H2
H2-R2	4	h2	r22	H2	H1
R2-R1	5	r21	r12	H2	H1
R1-H1	6	r11	h1	H2	H1

Exercici resolt: 2020t-c1-sol.pdf

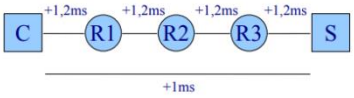
Duració: 1h 30 minuts. El test es recollirà en 25 minuts. ⟶ ~8 preguntes

Test (3,5 punts). Les preguntes valen la mitat si hi ha un error i 0 si hi ha més d'un error a la resposta.

1. El temps de transmissió d'un paquet de 1500 octets a 10 Mbps és 1,2 ms. En un enllaç determinat, el temps de propagació extrem a extrem entre un client i un servidor és d'1 ms. En aquest cas, el retard total extrem a extrem quan no hi ha cap node intermedi és de 2,2 ms.
Si afegim tres routers entre el client i el servidor:

- ☐ El retard mínim extrem a extrem serà 2,2 ms.
- ☐ El retard extrem a extrem serà com a màxim 6,6 ms.
- ☐ El retard mínim extrem a extrem serà 5,8 ms.
- ☐ El retard mínim extrem a extrem serà 4,6 ms.

$$temps\ transmissió = 1500\ octets \frac{8\,b}{1\,octet} \frac{1\,Mb}{1000000\,b} \frac{1}{10\frac{Mb}{s}} = 1{,}2\,ms$$



$$retard\ mínim\ extrem\ a\ extrem = 4 * 1{,}2\,ms + 1\,ms = 5{,}8\,ms$$

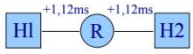
Exercici resolt: 2021t-c1-sol.pdf

2. Dos dispositius estan connectats a través d'un router. Suposem que el temps de propagació extrem a extrem és zero, que el router no afegeix retard a les cues i que la velocitat de transmissió dels enllaços és 10 Mbps.

- ☐ Si el paquet té 1400 octets (bytes) el temps de transmissió del paquet és 0'14ms
- ☐ Si el paquet té 1400 octets (bytes) el temps de transmissió del paquet és 1'12ms.
- ☐ Si el paquet té 1400 octets (bytes) el temps total fins que ha arribat a l'altre extrem és 2'24ms.
- ☐ Si es transmeten dos paquets de 700 octets (bytes) el temps total fins que el segon paquet arriba a l'altre extrem és 1'68ms.

1 pk de 1400 octets

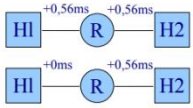
$$temps\ transmissió = 1400\ octets \frac{8\,b}{1\,octet} \frac{1\,Mb}{1000000\,b} \frac{1}{10\frac{Mb}{s}} = 1{,}12\,ms$$



$$temps\ total\ extrem\ a\ extrem = 2 * 1{,}12\,ms = 2{,}24\,ms$$

2 pk de 700 octets

$$temps\ transmissió = 700\ octets \frac{8\,b}{1\,octet} \frac{1\,Mb}{1000000\,b} \frac{1}{10\frac{Mb}{s}} = 0{,}56\,ms$$



⟶ 0ms pel primer hop del segon paquet perquè es fa en paral·lel al segon hop del primer paquet

$$temps\ total\ extrem\ a\ extrem = 3 * 0{,}56\,ms = 1{,}68\,ms$$

3. El model de referència ISO defineix 7 nivells: físic, enllaç de dades, xarxa, transport, sessió, presentació i aplicació.

- ☐ Tots els dispositius d'usuari i els routers de la xarxa gestionen (implementen) els 7 nivells.
- ☐ El model de referència TCP/IP agrupa els nivells de sessió, presentació i aplicació en un únic nivell d'aplicació.
- ☐ Tots els routers gestionen els nivells físic, enllaç de dades, xarxa i transport.
- ☐ El nivell de transport només el gestionen els dispositius d'usuari ("*hosts*").

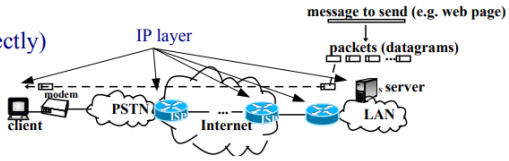
6. Sobre el model de comunicació client-servidor.

- ☐ Un host pot actuar a la vegada com a client i com a servidor.
- ☐ Els paquets d'una comunicació entre processos client i servidor s'identifiquen amb les adreces IP origen i destinació, els ports de client i de servidor, i el protocol.
- ☐ Un dispositiu pot establir moltes comunicacions com a client amb el mateix servidor i protocol.
- ☐ Un dispositiu amb una única adreça IP pot mantenir simultàniament moltes comunicacions client-servidor amb molts servidors diferents.

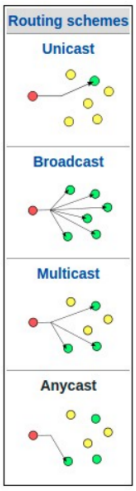
TEMA 2 - Xarxes IP

IP Layer Service

- Internet Protocol (IP)
  - Goal: routing packets from host to host
  - Design principle: interconnect hosts attached to LANs/WANs networks of different technologies
  - Characteristics
    - Connectionless (send/receive directly)
    - Stateless (no memory)
    - Best effort (no guarantee)



  - Design implications
    - Packets can be delivered out-of-order
    - Each packet can take a different path to the destination
    - No error detection (=> no correction) in payload
      - Left to higher layers, if any
    - No congestion control (beyond "drop")
  - Routing schemes: destinations
    - Unicast: one-to-one (multi-hop)
      - Ex. application layer: classic client-server
    - Broadcast: one-to-all (broadcast domain – data link layer)
      - Ex. link+IP layer: ARP request (upcoming slides)
    - Multicast: one-to-many
      - IP layer: Requires multicast routing: not supported by Internet
    - Anycast:
      - Application layer: DNS root nameservers clusters

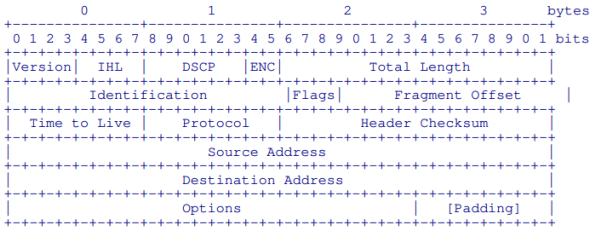


El procediment de **rebre els datagrames, processar-los i emmagatzemar-los fins que es transmeten **es coneix com a ***store and forward***. Cal destacar que les **taules d'encaminament** del router han d'estar **inicialitzades**. Si el router rep un datagrama dirigit a una **direcció desconeguda, el descarta**.

IPv4 Header (RFC 791 + updates)

Offsets	Octet	0							1							2								3						
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length														
4	32	Identification																Flags			Fragment Offset										
8	64	Time To Live								Protocol								Header Checksum														
12	96	Source IP Address																														
16	128	Destination IP Address																														
20	160	Options (if IHL > 5)																														
⋮	⋮																															
56	448																															

https://en.wikipedia.org/wiki/IPv4#Head



- 14 fields, 13 required
- `Version` – [4b] Version of the header: IPv4 vs. IPv6
  - `0100` for IPv4
  - `0110` for IPv6
- `IHL` (Internet Header Length) – [4b] Number of 32-bit words in the header
  - Min 5 (4*5 = 20 bytes); max 15 (4*15 = 60 bytes)
  - Length = [20, 24, 28, …, 60] bytes
- `DSCP` & `ECN` (previously `ToS`) – [6b] & [2b]
  - Differentiated Services Code Point – related to quality of service (QoS) management (e.g. voice over IP, VoIP)
  - Explicit Congestion Notification – related to end-to-end notification of network congestion
- `Total Length` – [16b] Packet size in bytes (header + data)
  - Min 20 bytes (header without data)
  - Max 65,535 (2^{16})
- `Identification` – [16b] Used for identifying the group of fragments of a single (fragmented) IP datagram
- `Flags` – [3b] Used to control fragmentation or identify fragments
  - bit 0: Reserved; always zero
  - bit 1: Don't Fragment (DF)
  - bit 2: More Fragments (MF)
- `Fragment offset` – [13b] Offset of a particular fragment relative to the beginning of the original unfragmented IP datagram in units of eight-byte blocks
- **`Time to Live`** – [8b] Each hop is decreased by one. IF zero, the router discards the packet and typically sends an ICMP time exceeded message to the sender.
- **`Protocol`** – [8b]
  - `0x01` (1) for ICMP
  - `0x04` (4) for IP-in-IP
  - `0x06` (6) for TCP
  - `0x11` (17) for UDP
- `Header checksum` – [16b] Used for error-checking of the header. Each router calculates the checksum of the header and compares it to the checksum field. If the values do not match, the router discards the packet.
- **`Source IP address`** – [32b]
- **`Destination IP address`** – [32b]
- Options – [max 40B] Options in 32-byte words

Fragmentació

IP **pot fragmentar** un datagrama quan la ***Maximum Transfer Unit* (MTU)** del nivell d'enllaç on ha d'encaminar-se té una **mida menor que la del datagrama.** Això sol passar en els següents casos:

- Quan un router ha d'encaminar un datagrama per una xarxa amb MTU menor que la d'on ha arribat.
- Quan es fa servir **UDP** i l'aplicació fa una **escriptura major que l'MTU de la xarxa**.

Quan es produeix la fragmentació, el reassemblatge es fa en la destinació dels datagrames. Per poder fer el reassemblatge es fan servir els camps:

- ***Identification*** *(16 bits)*: El nivell IP del *host* que genera els datagrames hi posa el valor d'un comptador que incrementa cada cop que es genera un nou datagrama. **Aquest número permet identificar fragments d'un mateix datagrama.**
- ***Flags*** *(3 bits)*: Són **0DM**. El primer bit no s'utilitza, els altres són:
  - D: **Flag de *Don't fragment***. Quan està a **'1' el datagrama no es pot fragmentar**. Si un router necessita fer-ho **el descartarà** i enviarà un missatge ICMP d'error. Es fa servir en el mecanisme MTU *path discovery*.
  - M: Flag de ***More fragments***. Si està a 1 vol dir que **hi ha més fragments**. Tots els datagrames el porten a '1' **excepte l'últim**.
- ***Offset*** *(13 bits)*: Posició del **primer byte** del fragment **en el datagrama original** (el primer val '0'). Es compta en unitats de 8 bytes.

MTU Path Discovery

El nivell de transport **TCP agrupa els bytes que escriu l'aplicació** fins a tenir un segment de **mida òptima** i després **l'envia**. La fragmentació no és desitjable perquè:

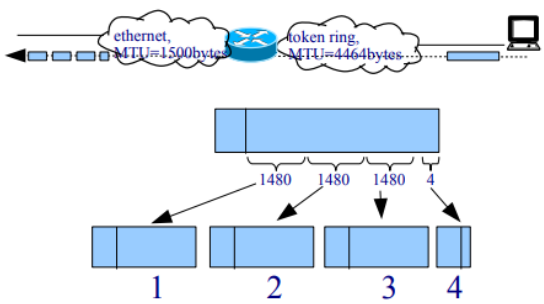
1. **Relantitza els routers**.
2. Pot provocar que hi hagi **fragments de mida petita** que redueixen **l'eficiència de la xarxa.**
3. Si es perd **un sol fragent**, s'han de **descartar tots els altres quan arriben a la destinació**.

MTU Path Discovery, consisteix en:

Primer proven d'**adaptar la mida dels segments a l'MTU de la xarxa on està connectat el *host***. Els datagrames s'envien amb el bit de ***Don't fragment*** **activat**. Si un router intermedi ha d'enviar-los per una **xarxa d'MTU menor, descartaran el datagrama** i enviaran un missatge ICMP d'**error *fragmentation needed but DF set***.. Quan el *host* rep el missatge d'error, **TCP redueix la mida dels segments** per adaptar-los en aquesta MTU.

IP Fragmentation - Example

- Original datagram = 4464 bytes (4Mbps Token Ring): 20 header + 4444 payload.
- Fragment size = $\left\lfloor \frac{1500-20}{8} \right\rfloor = 185$ 8-byte-words (1480 bytes) — Reminder: fragments sizes in bytes are multiple of 8
  - 1st fragment: offset = 0 , M = 1. 0~1479 payload bytes.
  - 2nd fragment: offset = 185, M = 1. 1480~2959 payload bytes.
  - 3rd fragment: offset = 370, M = 1 . 2960~4439 payload bytes.
  - 4th fragment: offset = 555, M = 0 . 4440~4443 payload bytes.



Adreces IP

Tenen **32 bits** (4 bytes). La següent figura mostra el format d'una adreça IP.

***Netid* identifica la xarxa i el *hostid* identifica un *host* dintre la xarxa**. El límit entre el netid i el hostid és variable. La notació que es fa servir es coneix com a *notació amb punts* i consisteix a expressar els **4 bytes de l'adreça en decimal separats per punts**, per exemple 147.83.34.25.

Hi ha diferents classes de ip; la classe **D és per adreces *multicast*** (per exemple la 224.0.0.1 identifica "*All systems on this Subnet*") i la **classe E són adreces reservades**. Tipus d'adreces:

Class	Netid [bytes]	Hostid [bytes]	Number of subnets	Netmask (slide 13)	Leading bits	Range
A	1	3	$2^7 = 128$	/8	0	0.0.0.0 – 127.255.255.255
B	2	2	$2^{14} = 16,384$	/16	10	128.0.0.0 – 191.255.255.255
C	3	1	$2^{21} = 2,097,152$	/24	110	192.0.0.0 – 223.255.255.255
D	-	-	-	-	1110	224.0.0.0 – 239.255.255.255
E	-	-	-	-	1111	240.0.0.0 – 255.255.255.255

L'assignació d'aquestes adreces es fa tenint en compte que:

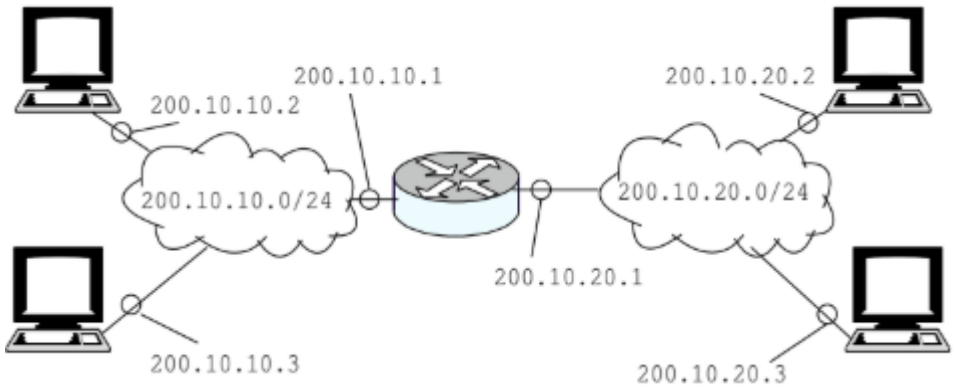
- **Una adreça identifica una interfície.**
- Totes les adreces d'una **mateixa xarxa IP** tenen el **mateix netid.**
- Totes les **adreces** assignades han de ser **diferents entre elles**

No totes les adreces es poden fer servir per numerar les interfícies. A continuació veiem les **adreces especials**:

netid	hostid	Significat
xxx	tot '0'	Identifica una xarxa. Es fa servir en les taules d'encaminament.
xxx	tot '1'	*Broadcast* en la xarxa xxx.
tot '0'	tot '0'	Identifica "aquest *host*" en "aquesta xarxa". Es fa servir com adreça origen en protocols de configuració (DHCP, vegeu la secció 2.7).
tot '1'	tot '1'	*Broadcast* en "aquesta xarxa". Es fa servir com adreça destinació en protocols d'autoconfiguració (DHCP, vegeu la secció 2.7).
127	xxx	*Loopback*: Comunicació entre processos en el mateix *host* amb TCP/IP.

A continuació veiem un exemple d'assignació d'adreces, destaquem que:

- Totes **les interfícies d'una mateixa xarxa tenen el mateix prefix** (netid).
- **No es poden fer servir adreces especials per a numerar interfícies**. Cada xarxa en té **dues**: la de la ***xarxa*** (amb el hostid tot a '0', que és la primera adreça disponible en el rang d'adreces de la xarxa) i la del ***broadcast*** en la xarxa (amb el hostid tot a '1', que és la ultima disponible en el rang d'adreces de la xarxa).
- El router ha de tenir assignada **una adreça en cada interfície**. Cada adreça ha de tenir **el netid de la xarxa on està connectada la interfície**.



Les **adreces han de ser úniques en tot internet**. Per això, l'organisme **IANA** assigna blocs d'adreces als ***Regional Internet Registers (RIR)***: ***RIEPE*** (EUR), ***ARIN*** (USA), ***APNIC*** (ASIA), ***LACNIC*** (LATAM).

A la vegada els RIR assignen blocs d'adreces als ***ISP*** i aquests als seus abonats..

Adreces Privades - RFC 1918

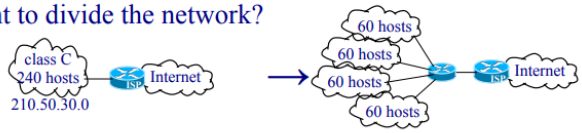
- **Private addresses** has been reserved for devices not using public addresses. These addresses are not assigned to any RIR (are not unique). There are addresses in each class:
  - 1 class A network: 10.0.0.0
  - 16 class B networks: 172.16.0.0 ~ 172.31.0.0
  - 256 class C networks: 192.168.0.0 ~ 192.168.255.0

Note: 169.254.0.0 – 169.254.255.255 is also a private range (used for IP autoconfiguration)

Subnetting

Per a expressar quina és la mida del netid es fa servir una màscara. La màscara és un número de 32 bits amb els bits més significatius que es corresponen amb el netid a '1' i els altres a '0'.

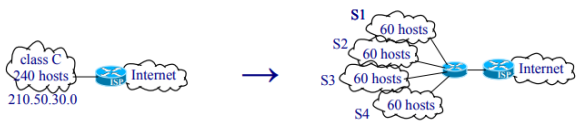
- Initially the netid was given by the address class: A with 2^{24} addresses, B with 2^{16} addresses and C with 2^{8} addresses.
- What if we want to divide the network?



- Subnetting allows adding bits from the hostid to the netid (called subnetid bits).
- Example: For the ISP the network prefix is 24 bits. For the internal router the network prefix is 26 bits. The 2 extra bits allows 4 "subnetworks".
- A **mask** (*netmask*) is used to identify the size of the netid+subnetid prefix.
- Mask notations:
  - dot notation, as 255.255.255.192
  - slash notation, giving the mask length (number of bits) as 210.50.30.0/26 (=> mask length: 26 bits)

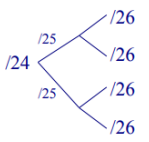
IP Addresses – Subnetting Example

- We want to subnet the address 210.50.30.0/24 in 4 subnets



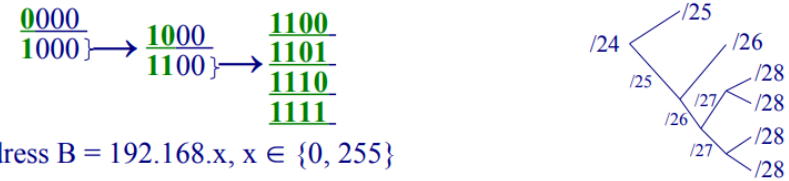
Base address B = 210.50.30

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



IP Addresses – Variable Length Subnet Mask (VLSM)

- Subnetworks of different sizes.
- Example, subnetting a class C address:
  - We have 1 byte for subnetid + hostid.
  - subnetid is green, chosen subnets addresses are underlined.



Base address B = 192.168.x, x ∈ {0, 255}

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

- In order to reduce routing tables size, CIDR proposed a "rational" geographical-based distribution of IP addresses to be able to "aggegate routes", and use masks instead of classes.
- Aggregation example:

200.1.10.0/24
200.1.11.0/24 ⟶ 200.1.10.0/23

Example

- We have been assigned the 192.169.1.0/24 (public) address block
- We want to split it into two equal subnetworks
- Secondly, we want to split in the same manner the two subnetworks
- Challenge: for each of the three cases find the network address, the network mask, the total number of IPs available for hosts, the lowest and the highest host IP and the broadcast IP

1) Whole block (/24)

```
user@dac:~$ ipcalc 192.169.1.0/24
Address:   192.169.1.0          11000000.10101001.00000001. 00000000
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000. 11111111
=>
Network:   192.169.1.0/24       11000000.10101001.00000001. 00000000
HostMin:   192.169.1.1          11000000.10101001.00000001. 00000001
HostMax:   192.169.1.254        11000000.10101001.00000001. 11111110
Broadcast: 192.169.1.255        11000000.10101001.00000001. 11111111
Hosts/Net: 254                   Class C
```

2) 2 subnets (two /25)

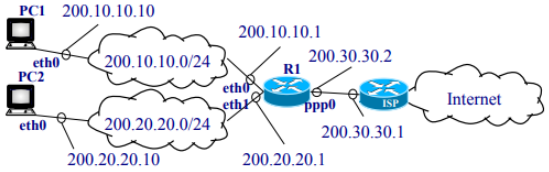
```
user@dac:~$ ipcalc 192.169.1.0/25
Address:   192.169.1.0          11000000.10101001.00000001.0 0000000
Netmask:   255.255.255.128 = 25 11111111.11111111.11111111.1 0000000
Wildcard:  0.0.0.127            00000000.00000000.00000000.0 1111111
=>
Network:   192.169.1.0/25       11000000.10101001.00000001.0 0000000
HostMin:   192.169.1.1          11000000.10101001.00000001.0 0000001
HostMax:   192.169.1.126        11000000.10101001.00000001.0 1111110
Broadcast: 192.169.1.127        11000000.10101001.00000001.0 1111111
Hosts/Net: 126                   Class C

user@dac:~$ ipcalc 192.169.1.128/25
Address:   192.169.1.128        11000000.10101001.00000001.1 0000000
Netmask:   255.255.255.128 = 25 11111111.11111111.11111111.1 0000000
Wildcard:  0.0.0.127            00000000.00000000.00000000.0 1111111
=>
Network:   192.169.1.128/25     11000000.10101001.00000001.1 0000000
HostMin:   192.169.1.129        11000000.10101001.00000001.1 0000001
HostMax:   192.169.1.254        11000000.10101001.00000001.1 1111110
Broadcast: 192.169.1.255        11000000.10101001.00000001.1 1111111
Hosts/Net: 126                   Class C
```

(cont.) 3) 4 subnets (four /26)

```
user@dac:~$ ipcalc 192.169.1.0/26
Address:   192.169.1.0          11000000.10101001.00000001.00 000000
Netmask:   255.255.255.192 = 26 11111111.11111111.11111111.11 000000
Wildcard:  0.0.0.63             00000000.00000000.00000000.00 111111
=>
Network:   192.169.1.0/25       11000000.10101001.00000001.00 000000
HostMin:   192.169.1.1          11000000.10101001.00000001.00 000001
HostMax:   192.169.1.62         11000000.10101001.00000001.00 111110
Broadcast: 192.169.1.63         11000000.10101001.00000001.00 111111
Hosts/Net: 62                    Class C

user@dac:~$ ipcalc 192.169.1.64/26
Address:   192.169.1.64         11000000.10101001.00000001.01 000000
Netmask:   255.255.255.192 = 26 11111111.11111111.11111111.11 000000
Wildcard:  0.0.0.63             00000000.00000000.00000000.00 111111
=>
Network:   192.169.1.64/25      11000000.10101001.00000001.01 000000
HostMin:   192.169.1.65         11000000.10101001.00000001.01 000001
HostMax:   192.169.1.126        11000000.10101001.00000001.01 111110
Broadcast: 192.169.1.127        11000000.10101001.00000001.01 111111
Hosts/Net: 62                    Class C

user@dac:~$ ipcalc 192.169.1.128/26
Address:   192.169.1.128        11000000.10101001.00000001.10 000000
Netmask:   255.255.255.192 = 26 11111111.11111111.11111111.11 000000
Wildcard:  0.0.0.63             00000000.00000000.00000000.10 111111
=>
Network:   192.169.1.128/25     11000000.10101001.00000001.10 000000
HostMin:   192.169.1.129        11000000.10101001.00000001.10 000001
HostMax:   192.169.1.190        11000000.10101001.00000001.10 111110
Broadcast: 192.169.1.191        11000000.10101001.00000001.10 111111
Hosts/Net: 62                    Class C

user@dac:~$ ipcalc 192.169.1.192/26
Address:   192.169.1.192        11000000.10101001.00000001.11 000000
Netmask:   255.255.255.192 = 26 11111111.11111111.11111111.11 000000
Wildcard:  0.0.0.63             00000000.00000000.00000000.00 111111
=>
Network:   192.169.192/25       11000000.10101001.00000001.11 000000
HostMin:   192.169.1.193        11000000.10101001.00000001.11 000001
HostMax:   192.169.1.254        11000000.10101001.00000001.11 111110
Broadcast: 192.169.1.255        11000000.10101001.00000001.11 111111
Hosts/Net: 62                    Class C
```

Routing Table

- ip_output() kernel function queries the routing table for each datagram.
- Routing can be:
  - Direct: The destination is directly connected to an interface.
  - Indirect: Otherwise. In this case, the datagram is sent to a router.
- Default route: Is an entry where to send all datagrams with a destination address to a network not present in the routing table. The default route address is 0.0.0.0/0.
- Hosts routing tables usually have two entries: The network where they are connected and a default route.



Genmask == Netmask

PC1 routing table:

Destination	Genmask	Gateway	Iface
200.10.10.0	255.255.255.0	0.0.0.0	eth0
0.0.0.0	0.0.0.0	200.10.10.1	eth0

PC2 routing table:

Destination	Genmask	Gateway	Iface
200.20.20.0	255.255.255.0	0.0.0.0	eth0
0.0.0.0	0.0.0.0	200.20.20.1	eth0

R1 routing table:

Destination	Genmask	Gateway	Iface
200.10.10.0	255.255.255.0	0.0.0.0	eth0
200.20.20.0	255.255.255.0	0.0.0.0	eth1
200.30.30.1	255.255.255.255	0.0.0.0	ppp0
0.0.0.0	0.0.0.0	200.30.30.1	ppp0

known destinations: network & mask

how to reach the destinations: (local) interface & gateway (next hop IP)

Routing Table – Another Unix Example

- Basic topology: Laptop connected to an access point (AP)

```
user@dac:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    600    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 eht0
```

- Laptop – AP with 2 other connections

U – Route up
G – Gateway
H – Host

```
user@dac:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    600    0        0 eth0
10.0.0.0        192.168.1.2     255.0.0.0       UG    0      0        0 eth0
10.0.0.2        192.168.1.3     255.255.255.255 UGH   0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 eth0
```

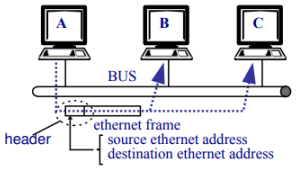
Default route
Network route
Host route
Local network

Warning: In GNU/Linux `route` is deprecated; `ip route show` (from the iproute2 package) must be used instead

Warning: routes with a longer prefix always take priority so in this case lower routes have priority (e.g. 10.0.0.2/32 over 10.0.0.0/8); however, routes shown by `route` may no respect this criterion; `ip route show` respects it.

Address Resolution Protocol, ARP (RFC 826)

- To send the datagram, IP layer may have to pass a "physical address" to the NIC driver. Physical addresses are also called MAC or hardware addresses.
- ARP translate IP addresses to "physical addresses" (used by the physical network).
- If needed, IP calls ARP module to obtain the "physical addresses" before the NIC driver call.
- Ethernet example:



Ethernet bus deployments are not used any more.

WiFi is a more up-to-date example: in a WiFi network all nodes receive all packages because they share physical layer => the MAC addresses are very meaningful. .

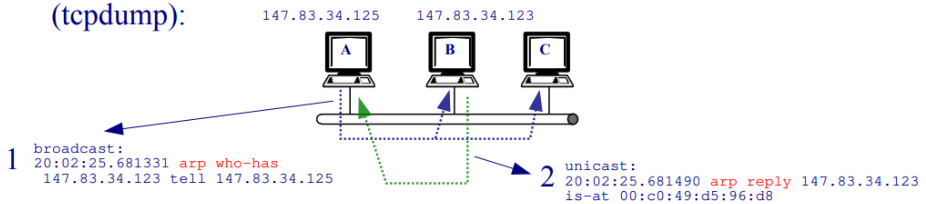
Note: In IPv6, the protocol that links IP to MAC addresses is the Neighbor Discovery (ND) Protocol.

Address Resolution Protocol, messages

- When IP calls ARP:
  - If ARP table has the requested address, it is returned,
  - otherwise:
    - IP stores the datagram in a temporal buffer, and a resolution protocol is triggered.
    - IP initiates a timeout and starts forwarding the next datagram in the transmission queue.
    - If the timeout triggers before resolution, the datagram is removed.
    - If ARP returns the requested address, IP calls the driver with it.
- ARP resolution in an ethernet network (broadcast network):
  - A broadcast "ARP Request" message is sent indicating the IP address.
  - The station having the requested IP address sends a unicast "ARP Reply", and stores the requesting address in the ARP table.
  - Upon receiving the "ARP Reply", the requesting station return the IP call with it.
  - ARP entries have a timeout refreshed each time a match occurs.

Address Resolution Protocol, messages - Example

- ARP messages (tcpdump):



- ARP tables:

```
A> /sbin/arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
147.83.34.123            ether   00:c0:49:d5:96:d8   C                     eth0

B> /sbin/arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
147.83.34.125            ether   00:14:F1:CC:59:00   C                     eth0
```

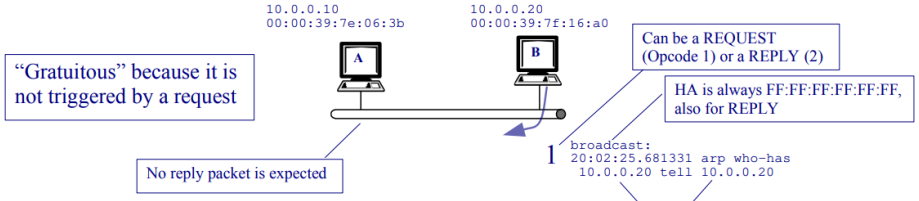
"Completed" flag

Warning: In GNU/Linux `arp` is deprecated; `ip link show` (from the iproute2 package) must be used instead (use 'ip l -s' to get statistics)

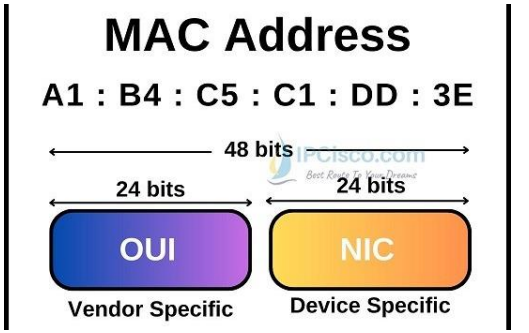
- Another example:

Time	No.	Source	Destination	Protocol	Length	Info	Pk len	Hardware src	Hardware dst
0.000000000	1	02:42:ac:1c:00:03	ff:ff:ff:ff:ff:ff	ARP	42	Who has 172.28.0.2? Tell 172.28.0.3	42	02:42:ac:1c:00:03	ff:ff:ff:ff:ff:ff
0.000148889	2	02:42:ac:1c:00:02	02:42:ac:1c:00:03	ARP	42	172.28.0.2 is at 02:42:ac:1c:00:02	42	02:42:ac:1c:00:02	02:42:ac:1c:00:03
0.000203120	3	172.28.0.3	172.28.0.2	ICMP	98	Echo (ping) request id=0x0014, seq=1/256, ttl=64 (reply in 4)	98	02:42:ac:1c:00:03	02:42:ac:1c:00:02
0.000304023	4	172.28.0.2	172.28.0.3	ICMP	98	Echo (ping) reply id=0x0014, seq=1/256, ttl=64 (request in 3)	98	02:42:ac:1c:00:02	02:42:ac:1c:00:03
5.159069240	5	02:42:ac:1c:00:02	02:42:ac:1c:00:03	ARP	42	Who has 172.28.0.3? Tell 172.28.0.2	42	02:42:ac:1c:00:02	02:42:ac:1c:00:03
5.159182181	6	02:42:ac:1c:00:03	02:42:ac:1c:00:02	ARP	42	172.28.0.3 is at 02:42:ac:1c:00:03	42	02:42:ac:1c:00:03	02:42:ac:1c:00:02
2315.53197…	7	fe80::42:66ff:fe2a:…	ff02::fb	MDNS	210	Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _p…	210	02:42:66:2a:02:ba	33:33:00:00:00:fb

Address Resolution Protocol – Gratuitous ARP



- Goals:
  - Detect duplicated IP addresses.
  - Update MAC addresses in ARP tables after an IP or NIC change.



Address Resolution Protocol – Message format (ethernet) (cont.)

- HTYPE (hardware type) – [2B] Specifies the network link protocol type
  - `0x0001` for Ethernet
- PTYPE (protocol type) – [2B] Specifies the internetwork protocol for which the ARP request is intended. Same numbering as EtherType (see Ethernet II header, Unit 3)
  - `0x0800` for IPv4
- HLEN (hardware length) - [1B] Specifies the hardware address length in octets
  - 6 for Ethernet (MAC address)
- PLEN (protocol length) - [1B] Specifies the internetwork address length in octets
  - 4 for IPv4
- OPER (operation) - [2B] Specifies the operation that the sender is performing
  - 1 for request
  - 2 for reply
- SHA (sender hardware address) [6B for Ethernet] Media address of the sender
  - In a request indicates the address of the host sending the request
  - In a reply indicates the address of the host that the request was looking for
- SPA (sender protocol address) [4B for IPv4] Internetwork address of the sender
- THA (Target hardware address) Media address of the intended receiver
  - In a request is ignored
  - In a reply indicates the address of the host that originated the ARP request
- TPA (Target protocol) [4B for IPv4] Internetwork address of the intended receiver

Internet Protocol (IPv4) over Ethernet ARP packet

Octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

https://en.wikipedia.org/wiki/IPv4#Header

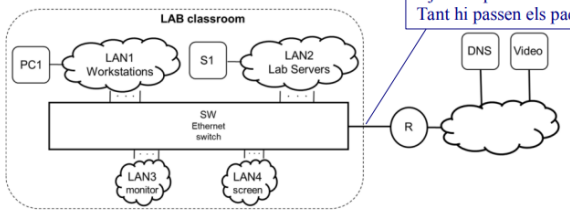
Exercici: 2021t-ef

Unit 2: IP Networks

Problema 1 (3,5 punts)

La figura mostra la configuració d'una aula de laboratori on hi ha llocs de treball (LAN1), servidors per donar suport als treballs dels laboratoris (LAN2), un PC de monitorització pel professor (LAN3) i una pantalla IP per a vídeo (LAN4). Cada laboratori disposa d'un commutador Ethernet (SW) on es configuren les 4 xarxes locals virtuals (VLAN) i l'adreçament proposat per a cada aula és 192.168.aula.0/24. El router R dona servei a més de 40 laboratoris amb la mateixa configuració.

Ajuda: aquest és l'enllaç que hem d'analitzar. Tant hi passen els paquets PC1-R com R-S1

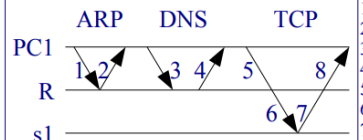


Ajuda: una connexió TCP comença amb el client enviat un SYN, al qual el servidor respon amb un SYN/ACK ...

c) (0,5 punts) La configuració que obté PC1 és: 192.168.1.2; router per defecte (gw): 192.168.1.1; DNS: 147.83.3.3. El PC1 inicia una connexió TCP amb el servidor s1-aula.fib.upc.edu. Completa la seqüència de trames i datagrames que passen per l'enllaç entre el commutador Ethernet i el router fins que PC1 rep el SYN/ACK. El router R ja té la informació a la taula ARP de tots els servidors.

Notació: majúscules per les adreces IP i minúscules per les adreces Ethernet (MAC). Exemple: PC1, pc1.

Ajuda: el cronograma d'intercanvi de paquets ens ajuda a visualitzar millor aquest intercanvi, I a comptar la quantitat de paquets



	Ethernet		ARP		IP			
	Origen	Destinació	Comanda	Missatge	Origen	Destinació	Protocol	Contingut
1	pc1	FF...FF	REQ	R?				
2	r	pc1	RESP	R -> r				
3	pc1	r			PC1	DNS	UDP	s1-aula.fib.upc.edu A?
4	r	pc1			DNS	PC1	UDP	DNS A=S1
5	pc1	r			PC1	S1	TCP	SYN
6	r	s1			PC1	S1	TCP	SYN
7	s1	r			S1	PC1	TCP	SYN/ACK
8	r	pc1			S1	PC1	TCP	SYN/ACK

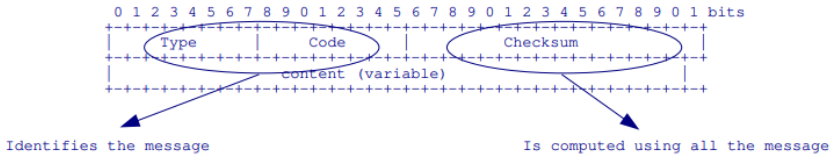
Advertència: no sempre cal omplir totes les cel·les de les taules dels anunciats

Internet Control Message Protocol, ICMP (RFC 792)

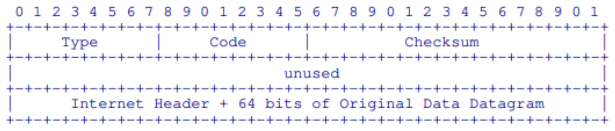
- Used for attention and error messages.
- Can be generated by
  - IP (e.g. TTL expiration)
  - ARP (e.g. resolution not possible)
  - Applications (e.g. ping)
- Are encapsulated into an IP datagram
  - (no UDP/TCP!)
  - It is an internet layer protocol
- Can be
  - i) query
  - ii) error
- Error messages are sent to the source IP of the datagram that generated the error condition
- An ICMP error message cannot generate another ICMP error message (to avoid loops)

Error messages only! Ping is a query message => it can trigger an ICMP error message (e.g. "network unreachable", "time exceed")

ICMP general format message (RFC 792)



- Query type messages have an identifier field, for request-reply correspondence.
- Error messages have a field where the first 8 bytes of the datagram payload causing the error are copied. These bytes capture the TCP/UDP ports. E.g. Destination Unreachable Message:



Code	Message	Use
1	DHCPDISCOVER	Client broadcast to locate available servers.
2	DHCPOFFER	Server to client in response to DHCPDISCOVER with offer of configuration parameters.
3	DHCPREQUEST	Client message to servers either (a) requesting offered parameters from one server and implicitly declining offers from all others, (b) confirming correctness of previously allocated address after, e.g., system reboot, or (c) extending the lease on a particular network address.
5	DHCPACK	Server to client with configuration parameters, including committed network address.

Common ICMP messages

Type	Code	query/error	Name	Description
0	0	query	echo reply	Reply an echo request
3	0	error	network unreachable	Network not in the RT.
	1	error	host unreachable	ARP cannot solve the address.
	2	error	protocol unreachable	IP cannot deliver the payload
	3	error	port unreachable	TCP/UDP cannot deliver the payload
	4	error	fragmentation needed and DF set	MTU path discovery
4	0	error	source quench	Sent by a congested router.
5	0	error	redirect for network	When the router send a datagram by the same interface it was received.
8	0	query	echo request	Request for reply
11	0	error	time exceeded, also known as TTL=0 during transit	Sent by a router when `--TTL=0`

Dynamic Host Configuration Protocol, DHCP (RFC 2131)

- Improves and can interoperate with previous BOOTP protocol.
- Used for automatic network configuration for assigning:
  - IP address and mask,
  - Default route,
  - Hostname,
  - DNS domain,
  - DNS servers,
  - etc.
- IP address configuration can be:
  - Dynamic: During a leasing time.
  - Automatic: Unlimited leasing time.
  - Manual: IP addresses are assigned to specific MAC addresses.
- It is an application layer protocol (server-client paradigm)
  - Uses UDP as transport protocol

DHCP – Message Fields (RFC 2131)

(informative slide, don't learn the message fields by heart!)

Only these 2 exist (not to be confused with `options`).
BOOTREQUEST: client to server
BOOTREPLY: server to client

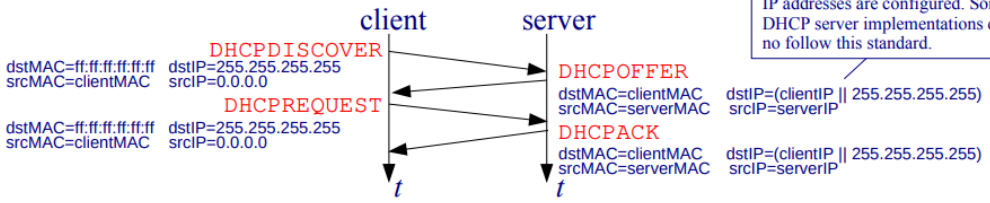
FIELD	OCTETS	DESCRIPTION
op	1	Message op code / message type. 1 = BOOTREQUEST, 2 = BOOTREPLY.

DHCP – Protocol Messages (RFC 2131)

Unit 2: IP Networks

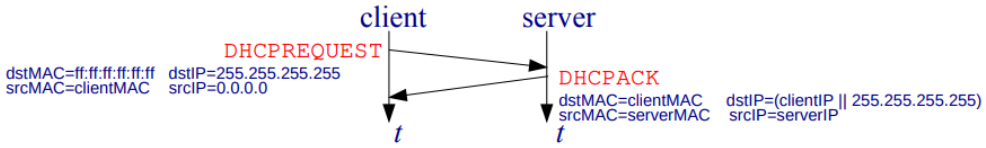
DHCP – Client-server interaction (RFC 2131)

- UDP, server port = 67, client port = 68.



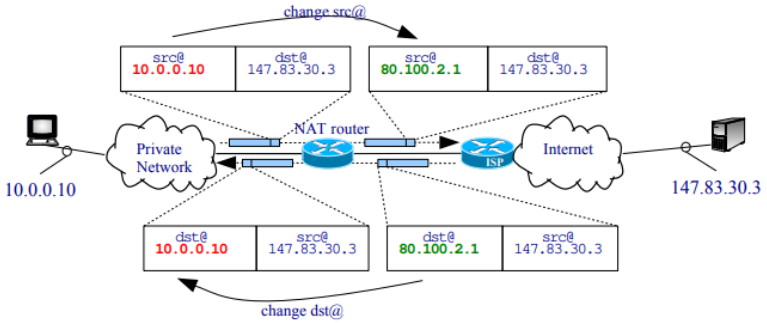
The client specifies whether in the reply (BOOTREPLY) the destination IP should be unicast or broadcast (i.e. clinetIP or 255.255.255.255). Unicast is the standard. Broadcast must be used only by those (old) hosts which cannot accept unicast packets before IP addresses are configured. Some DHCP server implementations do no follow this standard.

- The client can directly send `DHCPREQUEST`:
  - After rebooting if it remembers and wishes to reuse a previously allocated network address.
  - Extending the lease on a particular network address.



Network Address Translation, NAT (RFCs 1631, 2663 3022)

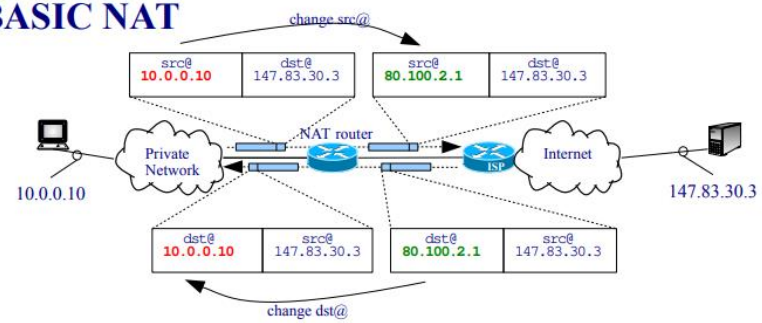
- Typical scenario: Private addresses (internal addresses) are translated to public addresses (external addresses).
- A NAT table is used for address mapping.
- Advantages:
  - Save public addresses.
  - Security.
  - Administration, e.g. changing ISP does not imply changing private network addressing.



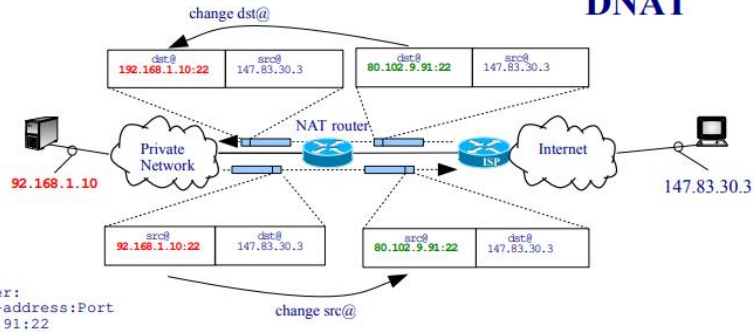
NAT – Types of translations

- Basic NAT (one-to-one):
  - A different external address is used for each internal address:
    - A different public IP address is needed for each hosts accessing Internet.
  - Each NAT table entry has the tuple: (internal address, external address).
  - Each host requires one NAT table entry.
  - Manually configured
- Port and Address Translation, PAT (one-to-many):
  - The same external address can be used for each internal address.
    - A unique public IP address can be used for all (internal) hosts accessing Internet.
  - Each NAT table entry has the tuple: (int. addr., int. port, ext. addr., ext. port)
  - Each connection requires one NAT table entry.
  - Entries are automatically added when an internal connection is initiated.
- Destination NAT, DNAT can be:
  - Enables external connections to internal servers.
  - The address translation is exactly the same as NAT, but, the connection is initiated from an external client.
  - Typically, some static configuration is needed to configure the server IP/port.

BASIC NAT



DNAT




```
Static entry in the NAT router:
Inside-address:Port  Outside-address:Port
192.168.1.10:22      80.102.9.91:22
```

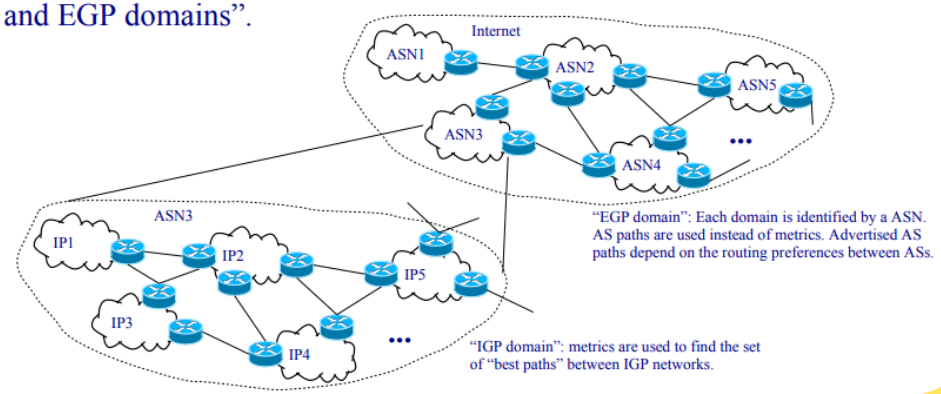
Routing algorithms

- Objective: add/update/remove entries to routing tables. Can be:
  - Static: Manual, scripts, DHCP.
  - Dynamic: Automatically update table entries, e.g. when a topology change occurs. This is done by a routing algorithm (also routing protocol).

- Internet is organized in Autonomous Systems (AS). In terms of ASs, routing algorithms are classified as:
  - Interior Gateway Protocols (IGPs): Inside the same AS. Examples:
    - RFC standards: RIP, OSPF.
    - Proprietary: CISCO IGRP.
  - Exterior Gateway Protocols (EGPs): Between different Ass.
    - Currently BGPv4.

Routing algorithms - Autonomous Systems (AS)

- AS definition (RFC 1930): "An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy".
- Each AS is identified by a 16 or 32 bits AS Number (ASN) assigned by IANA.
- ASs facilitate Internet routing by introducing a two-level hierarchy: "IGP and EGP domains".

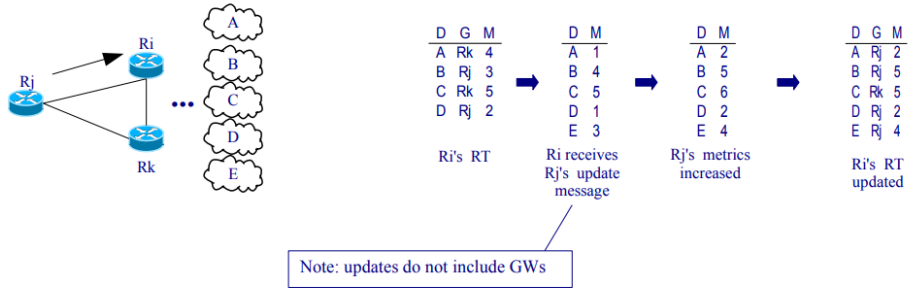


Routing Information Protocol, RIP (RFC 2453)

- The metric (distance) to a destination is the number of hops (i.e. transmissions) to reach the destination: 1 if the destination is attached to a directly connected network, 2 if 1 additional router is needed, etc.
- Routers send RIP updates every 30 seconds to the neighbors.
- RIP updates use UDP, with the same source and destination port: 520, broadcast dst. IP addr. (Version 1).
- RIP updates include destinations and metrics tuples.
- A neighbor is considered down if no RIP messages are seen during 180 seconds.
- Infinite metric is 16.
- Two versions of RIP: Version 2 i) allows variable masks (=> masks are added to the messages) and ii) uses the multicast dst. address 224.0.0.9 (all RIPv2 routers).
- This type of routing algorithms, where it is not known the whole topology but just the distance to each destination, are known as "distance-vector" and use a distributed variant of the "Bellman-Ford" algorithm for selecting the shortest path.

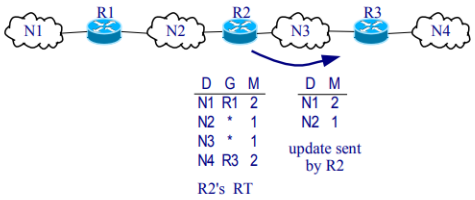
RIP – Routing Table (RT) Update Example

- Upon receiving an update
  1) Increase the message metrics (+1 to all metrics received)
  2) change the routing table if:
     - There is a better path (lower metric) towards a destination
     - The gateway being used changes the metric
     - There is a new route
- Example: When Ri receives an update message from Rj:



RIP – Count to Infinity: Solutions

- Split horizon: When the router sends the update, removes the entries having a gateway in the interface where the update is sent:



- Split horizon with Poisoned Reverse: the same as split horizon except that the entries with M=16 are not removed. The poison reverse rule overwrites split horizon rule: *poisoned routes* are also sent ("back") to the neighbor from which were learnt.
- Triggered updates: Consists of sending the update before the 30 seconds timer expires when a metric change in the routing table.
- Hold down timer (CISCO): When a route becomes unreachable (metric = 16), the entry is placed in *holddown* during 180 seconds. During this time, the entry is not updated.

To allocate time for poisoned routes to propagate.

Security in IP

- Goals:
  - Confidentiality: Who can access.
  - Integrity: Who can modify the data.
  - Availability: Access guarantee.
- Vulnerabilities:
  - Technological: Protocols (e.g. ftp and telnet send messages in "clear text") and networking devices (routers...)
  - Configuration: Servers, passwords, ...
  - Missing security policies: Secure servers, encryption, firewalls, ...

Security in IP – Attacks

- Reconnaissance: Previous to an attack.
  - Available IP addresses.
  - Available servers and ports.
  - Types of OSs, versions, devices...
  - Eavesdropping
- Access: Unauthorized access to an account or service.
- Denial of Service: Disables or corrupts networks, systems, or services.
- Viruses, worms , trojan horses...: Malicious software that replicate itself.

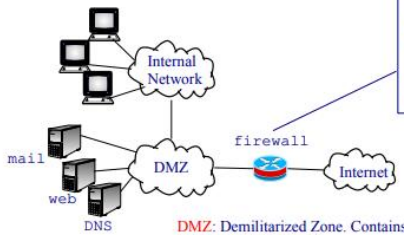
Security in IP – Basic Solutions

- Firewalls.
- Virtual Private Networks (VPN) with encrypted payload.

Security in IP – Firewalls

- Firewall: System or group of systems that enforces an access control policy to a network.
- There are many firewall types:
  - From simple packet filtering based on IP/TCP/UDP header rules,
  - to state-full connection tracking
  - and application-based filtering (packet inspection)

Middlebox is a generic term to refer devices installed between routers (frequently without knowledge of one of both communicating parties). They include: firewalls, traffic shapers, proxies, etc.



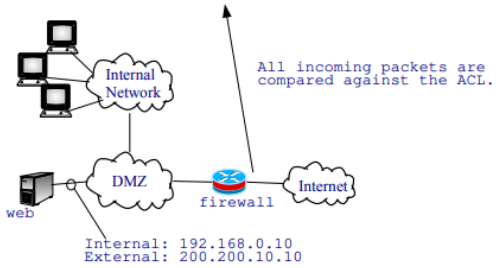
DMZ: Demilitarized Zone. Contains servers exposed to the Internet.

Security in IP – Basic Firewall Configuration

- NAT
- Access Control List, ACL

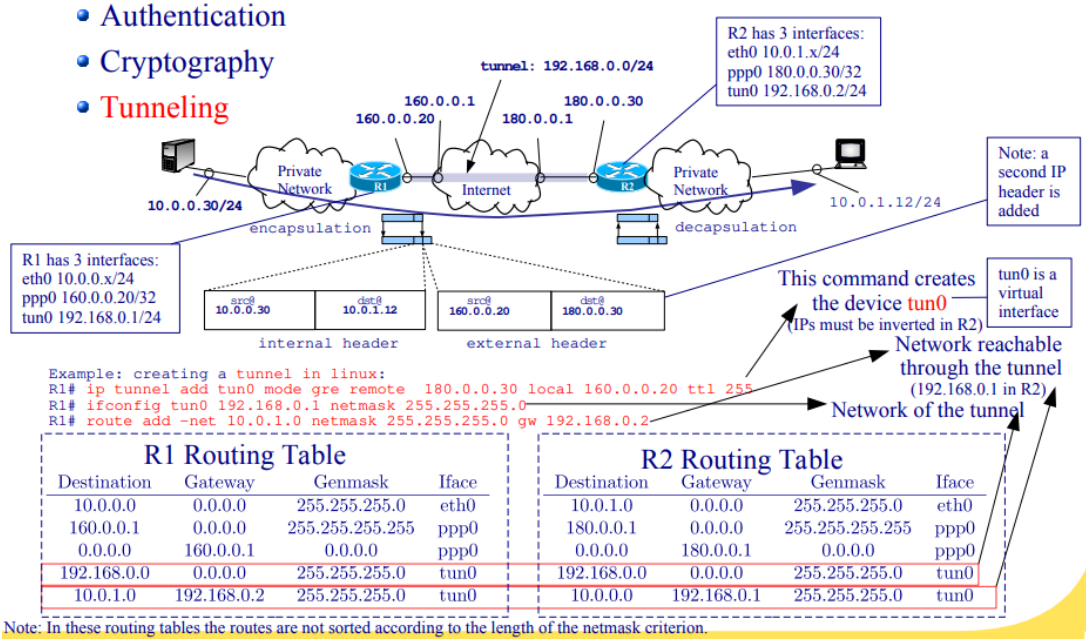
The order in which the entries appear in the list is crucial: once a packet matches a rule the corresponding action action is taken and no further entries are processed

Protocol	IP-src	IP-dst	Port-src	Port-dst	Action
TCP	*any*	200.200.10.10/32	*any*	80	*accept*
TCP	*any*	*any*	< 1024	≥ 1024	*accept*
ICMP	*any*	*any*	–	–	*accept*
IP	*any*	*any*	–	–	*deny*



Security in IP – VPN Security

- Authentication
- Cryptography
- Tunneling



R1 Routing Table

Destination	Gateway	Genmask	Iface
10.0.0.0	0.0.0.0	255.255.255.0	eth0
160.0.0.1	0.0.0.0	255.255.255.255	ppp0
0.0.0.0	160.0.0.1	0.0.0.0	ppp0
192.168.0.0	0.0.0.0	255.255.255.0	tun0
10.0.1.0	192.168.0.2	255.255.255.0	tun0

R2 Routing Table

Destination	Gateway	Genmask	Iface
10.0.1.0	0.0.0.0	255.255.255.0	eth0
180.0.0.1	0.0.0.0	255.255.255.255	ppp0
0.0.0.0	180.0.0.1	0.0.0.0	ppp0
192.168.0.0	0.0.0.0	255.255.255.0	tun0
10.0.0.0	192.168.0.1	255.255.255.0	tun0

Note: In these routing tables the routes are not sorted according to the length of the netmask criterion.

Security in IP – VPN Tunneling Problems

- Fragmentation inside the tunnel will use the external header, thus, the exit router of the tunnel may reassemble fragmented datagrams.
- ICMP messages sent inside the tunnel are addressed to the tunnel entry.
- MTU path discovery may fail.
- Solution: the router entry maintains a "tunnel state", e.g. the tunnel MTU, and generate ICMP messages that would be generated inside the tunnel. Furthermore, the tunnel entry router typically fragment the datagrams, if needed, before encapsulation, to avoid the exit router

IEEE LAN Architecture – IEEE 802 standards (some)

Types of tunnels:

802.3 Ethernet

/01): There is an additional

802.11 WiFi: Wireless LANs.

ɔcols (not only IP).

- Point-to-Point Tunneling Protocol, PPTP (RFC 2637): Add the ppp functionalities.
- IPsec (RFC 2401): Standards to introduce authentication and encryption and tunneling to IP layer.

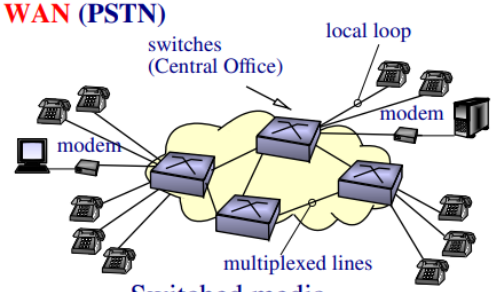
Unit 3. Local Area Networks, LANs

Introduction – WAN and LAN differences

- WANs:
  - Main goal: scalability.
  - Switched network with mesh topology.
- LANs:
  - Multy-access network **with shared media**.
  - A Medium Access Control (MAC) protocol is needed.

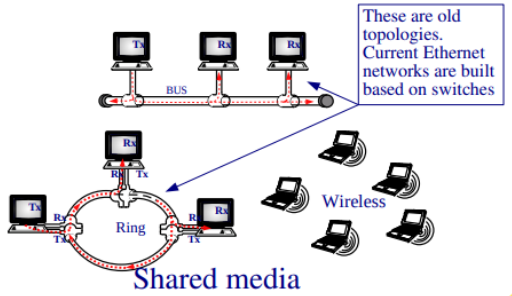
To govern the access to the transmission medium

WAN (PSTN)



(PSTN – Public switched telephony network)

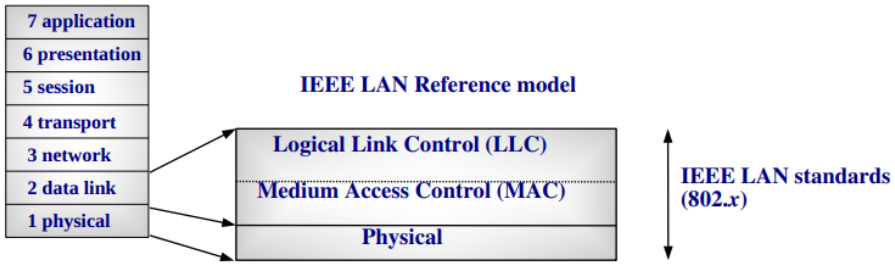
LANs



Unit 3. Local Area Networks, LANs

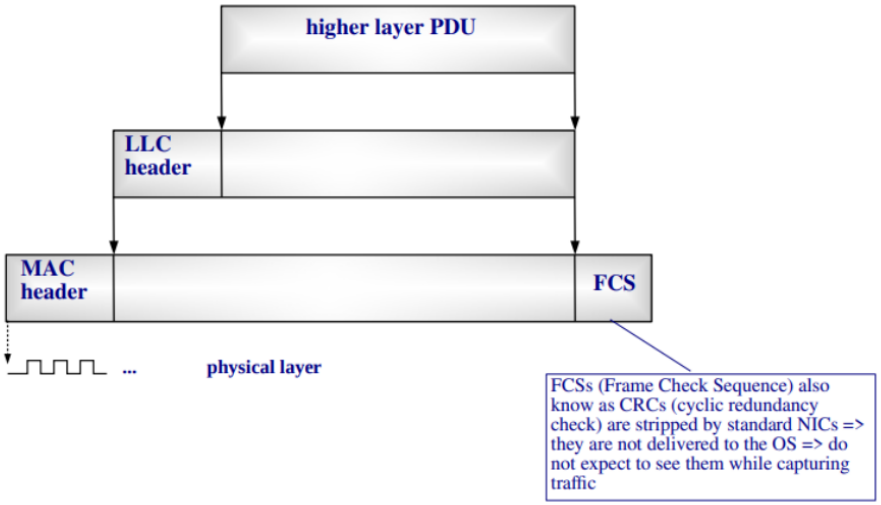
IEEE LAN Architecture

OSI Reference model:



- LLC sublayer (802.2):
  - Common to all 802.x MAC standards.
  - Define the interface with the upper layer and specifies several services (operational modes):
    - (i) unacknowledged connectionless, (ii) connection oriented, (iii) acknowledged connectionless.
- MAC sublayer:
  - Define the medium access protocol. It is different for each LAN technology.

IEEE LAN Architecture – LAN encapsulation



Types of MACs

- Token Passing:
  - Only the station having the token can transmit. After transmission the token is passed to another station.

    Examples: FDDI and Token-Ring (obsolete)
- Random:
  - There is no token. Instead, there is a non null collision probability. In case of collision, the frame is retransmitted after a random *backoff* time.
  - Examples: Ethernet, WiFi

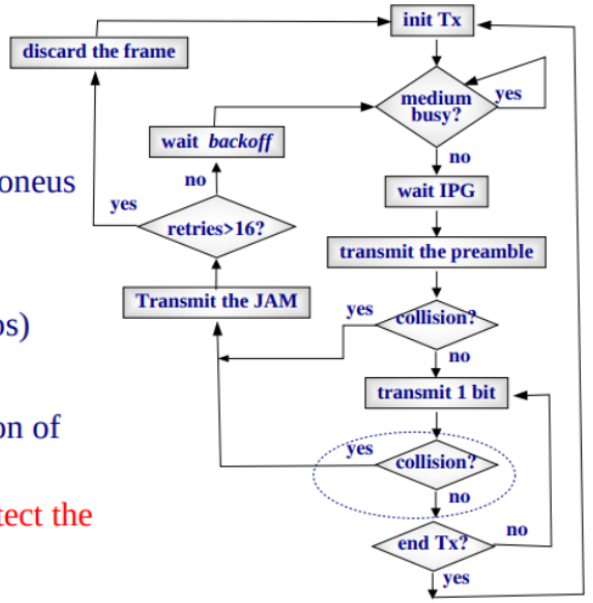
Carrier Sense Multiple Access/Collision Detection (CSMA/CD)

- Is a random MAC where the stations "listen" the medium (*carrier sense*) before transmission.
- When the medium is becomes free the frame is transmitted immediately, and the medium is listened to detect collisions.
- In case of collision, the frame is retransmitted after a random *backoff* time.

Ethernet – CSMA/CD Ethernet protocol (simplified)

Legend:

- InterPacket Gap (IPG): 96 bits.
- JAM: 32 bits that produce an erroneus CRC.
- *backoff* = $n\ T_{512}$
- T_{512}:*SlotTime* (51,2 μs at 10 Mbps)
- n = random{0, $2^{\min\{N,\ 10\}}$-1},
  - N: number of retransmission of the same frame (1, 2…)
- The transmitting station must detect the collision (no ack is sent).



Ethernet – Half Duplex and full-duplex

- Half Duplex: Using CSMA/CD only one NIC can be simultaneously transmitting into the medium.
- Full Duplex: When 2 Ethernet NICs are connected point-to-point, some Ethernet standards allow a full-duplex Tx, .
- Ethernet NICs have an auto-negotiation mechanism to detect the full-duplex availability.
- In full-duplex mode Ethernet NICs deactivate CSMA/CD (no collisions can occur).

Ethernet – Frames

- Ethernet II (DIX) Frame:

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

- Min. length: 64B => Padding with 0 if needed (e.g. ARP). Wireshark keep in mind:
  - FCS also known as CRC (cyclic redundancy check) are stripped by NICs => -4B
  - Padding done by NICs => Not show in outgoing frames
- Preamble: Give time to detect, synchronize and start reception.
- Frame type: Identifies the upper layer protocol (IP, ARP, etc. RFC 1700, Assigned numbers)
  - `0x0800` for IPv4
  - `0x0806` for ARP
  - `0x86DD` for IPv6
  - (`0x8808` for Ethernet flow control (Control opcode: 0x0001))
  - 0x8100 for VLAN tagging (IEEE-802.1Q)

Ethernet – Different Ethernet Standards: 10BaseT

1990. Cable UTP-cat 3.

- Hub: Is a multi-port repeater (layer 1).
- The signal received in 1 port is retransmitted by all the others.
- Always half duplex

Otherwise it could receive data from more than one port simultaneously, which is incompatible with "The signal received in 1 port is retransmitted by all the others"

10BaseT segments

Ethernet – Different Ethernet Standards: after 10BaseT

All standards use UTP o OF (except 10GBaseCX4):

- Fast Ethernet (1995). 100BaseTX: UTP-cat. 5
- Gigabit Ethernet (1998). 1000BaseT: UTP-cat 5e
- 10Gigabit Ethernet (2002). Uses optical fiber. The only copper standard is Infiniband with segment size ≤ 15m.

Ethernet – Different Ethernet Standards

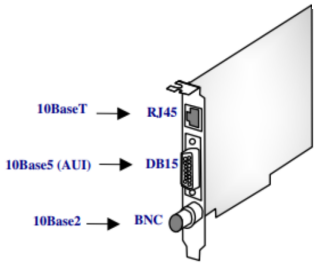
Denomination:

*x*Basey

Line bitrate:
- 10: 10 Mbps
- 100: 100 Mbps
- 1000: 1000 Mbps (1 Gbps)
- 10G: 10 Gbps

Base band signal.
Broad: translated band signal.

Various meanings:
- Number: Maximum segment distant in hundreds of m.
- Reference to the medium type:
  - T: UTP
  - F: Optical Fiber
  - Other:
  - T4: Uses 4 UTP pairs.
  - TX: Full Duplex
  - ...

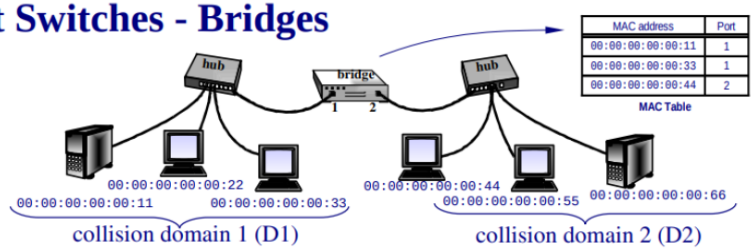


NIC "combo":
Supports 10Base5, 10Base2, 10BaseT

Ethernet Switches - Introduction

- Hub problem: If many stations are connected, may be inefficient due to collisions.
- Solution: bridges and switches.
- Ethernet bridge:
  - "plug and play" layer 2 device.
  - In each port there is a NIC in "promiscuous" mode: Capturing all frames.
  - The source address is used to "learn" which MAC is present in each port (MAC table). Each entry has the MAC and the port numbers.
  - The destination MAC is used to decide whether the frame needs to be retransmitted by another port.
  - Segments the "collision domain".

Ethernet Switches - Bridges



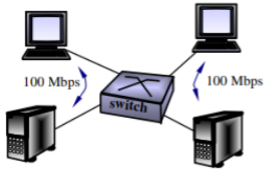
How the bridge works:

- If a frame is received with a source address not in the MAC table, it is added (*learning bridge*).
- If a frame from D1 is received with a destination address that: (i) is in D2, (ii) it is not in the table, (iii) it is broadcast: It is sent into D2 (*flooding*).
- If it is received a frame from D1 addressed to another station from D1, it is discarded (*filtering*).
- The entries have an *aging timer*. Each time an entry is used, it is refreshed. If the aging timer expires, the entry is removed.

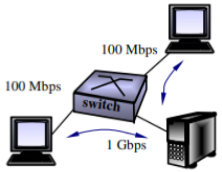
Advantages:

- Segments the collision domain (less collisions).
- Clients in D1 and D2 can simultaneously send frames to their servers.

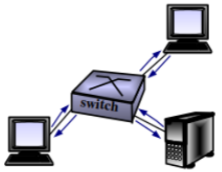
Ethernet Switches - Switch Capabilities



Simultaneous Transmissions



Ports with Different bitrates



Full Duplex Ports

- Each port is different a collision domain (less collisions).
- Different ports can be simultaneously Tx/Rx.
- Ports can have different bitrates.
- Ports may be full-duplex (usable if only one host is connected).
- There can be ports simultaneouly in half or full duplex mode.
- Link aggregation: Bitrate can be increased by aggegating several links, which behave as a slingle one (*etherchannel* in CISCO).
- Security: Stations can only capture the traffic of their collision domain.

Ethernet Switches - Switch Architecture

How the switch works:

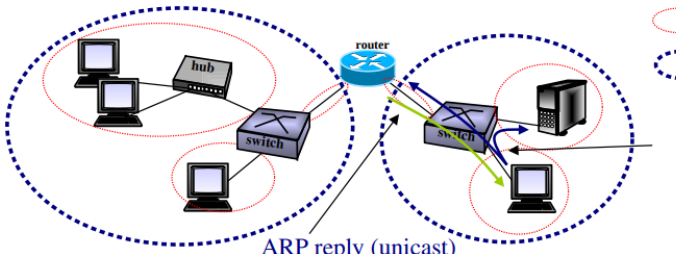
- It is equivalent to a "multiport bridge".
- When a frame is received with a source address not in the table, it is added.
- If a frame is received with a destination address: (i) not in the table, (ii) broadcast or multicast: copy the frame in all transmission buffer of the other ports (flooding).
- If a frame is received with the address from another port: It is switched as fast as possible the the transmission buffer of that port.
- If receives a frame addressed to another station from the same port, it is discarded (filtering).

```
Switch#show mac-address-table
Address          Dest Interface
---------------------------------
00D0.5868.F583    FastEthernet 2
00E0.1E74.6ADA    FastEthernet 1
00E0.1E74.6AC0    FastEthernet 1
0060.47D5.2770    FastEthernet 3
00D0.5868.F580    FastEthernet 5
```

MAC Table in a CISCO Switch

Ethernet Switches - Broadcast and Collision Domains

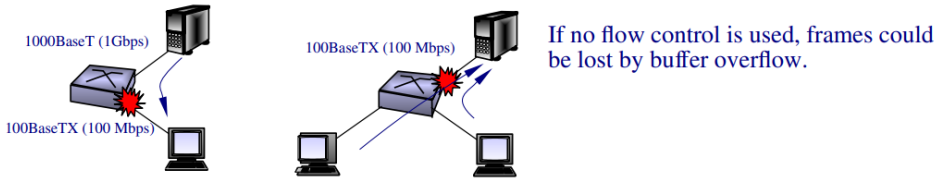
- Broadcast Domain: Set of stations that will received a broadcast frame sent by any of them.
- Unless Virtual LANs are used, a switch does not segment the broadcast domain.
- A router segment the broadcast domain.
- The broadcast reachability is important because allows reaching stations having one hop connectivity (with ARP).



ARP cannot solve an @IP out of the broadcast domain. To leave the broadcast domain a router is required.

Ethernet Switches – Flow Control

- Switch Flow Control: Consists of adapting the rate at which the switch receives the frames, and the rate at which the switch can send them.
- Examples:

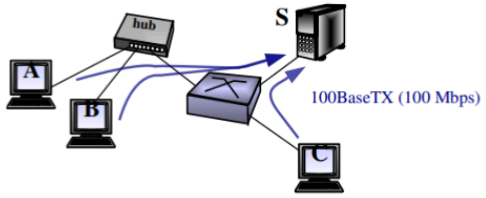


If no flow control is used, frames could be lost by buffer overflow.

- Flow control techniques (back pressure):
  - Jabber signal (half duplex): The switch sends a signal into the port which need to be throttled down, such that CSMA see the medium busy.
  - Pause frames (full duplex): The switch send special *pause frames*. These frames have an integer (2 bytes) indicating the number of slot-times (512 bits) that the NICs receiving the frame must be silent.

Ethernet Switches – Line bitrate sharing

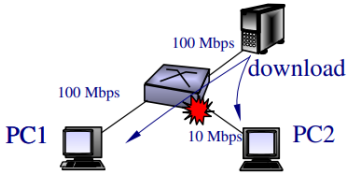
- Hub: If the hub is the bottleneck for all the active ports, the capacity is equally shared between all ports where frames are transmitted.
- Switch: If one congested port is the bottleneck for all ports sending traffic to it, the port bit rate is equally shared between all ports sending traffic to it.
- Example:



  - If A, B and C simultaneously transmit to S:
    throughput C ≈ 100 Mbps / 2 = 50 Mbps
    throughput A = throughput B ≈ (100 Mbps / 2) / 2 = 25 Mbps

Ethernet Switches – Problems of Flow Control

- Flow Control can introduce inefficiencies (*head of line blocking*):

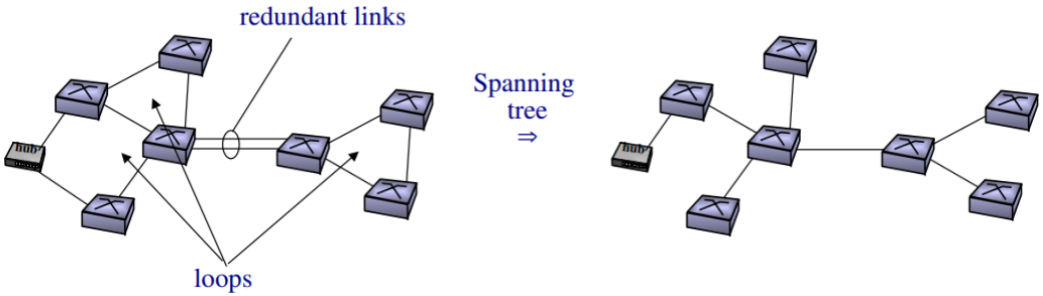


The slow link may trigger the flow control and send pause frames towards the server, causing under-utilization of the switch-server link.

- We would expect a download of approximately 90 Mbps for PC1 and 10 Mbps for PC2. However, the flow control can make the PC1 throughput to be significantly lower. Switches allow disabling the flow control in a link (In CISCO: Switch(config-if)# flowcontrol send off). If flow control is disabled, traffic is assumed to be controlled by TCP.
- If not otherwise stated, we shall assume an *ideal* flow control in the problems, which allow achieving the maximum throughput.

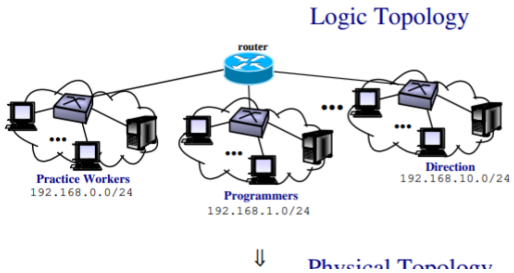
Ethernet Switches – Spanning Tree Protocol (STP)

- STP goal: Build a loop free topology (STP-tree) with optimal paths. The ports that do not belong to the STP tree are blocked.
- The switches send 802.1D messages to their neighbors to build up the STP-tree. If the topology changes (e.g. due to a link failure), a new STP-tree is setup.



Ethernet Switches – Virtual LANs, VLANs

- Motivation:
- Grouping related servers and hosts in different broadcast domains.
- How VLANs work:
- Each switch port belongs to a VLAN.



Ethernet Switches – VLAN Trunking

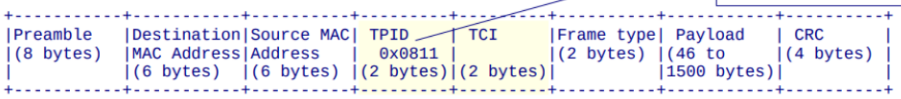
- Trunking:
- The port configured as trunk belongs to several VLANs (maybe all).
- The traffic sent in one VLAN is also sent to the trunk the VLAN belongs to.
- A tagging mechanism is used in the trunk to discriminate the traffic from different VLANs.

Wireless LANs (WLANs) – 802.11 MAC

- Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA): In contrast to CSMA/CD, always wait a random backoff before Tx. Acks are needed to detect whether a transmitted frame collided.
- CSMA/CD is not used because collisions can hardly be detected in wireless (because in the antenna the Tx power is orders of magnitude higher than Rx power).

- Trunking Protocols:
- Inter-Switch Link (ISL). CISCO propietary protocol.
- IEEE-802.1Q.

This is the position "Frame type" of a standard Ethernet II header. The value 0x0811 adds 4 extra bytes to the header: the first 2 for the "TCI" and the last 2 for the the "Frame type" of the next header (IPv4, IPv6, ARP, etc.)



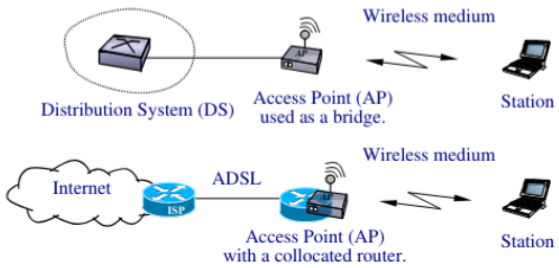
IEEE-802.3 frame with the 802.1Q *tag*.

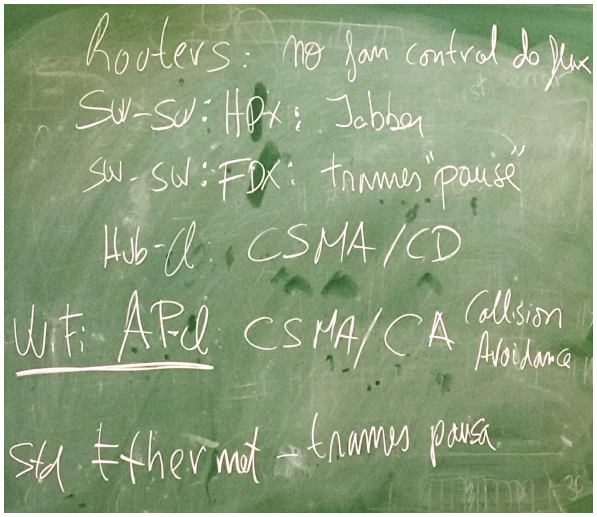
Legend:

- *Tag Protocol Identifier* (TPID): Field with the hex. value 0x8100 for an Ethernet frame.
- *Tag Control Information* (TCI): Contains several fields. The most important is the VLAN ID (12 bits), which identify the VLAN.

Wireless LANs (WLANs) – 802.11 Components

- Distribution System (DS):
  - Used by APs to exchange frames with one another and with wired networks. (e.g. an ethernet switch).
- Access Point (AP)
  - Simplify communication between stations.
  - All transmissions go through the AP.
  - APs are bridges and may have a collocated router.





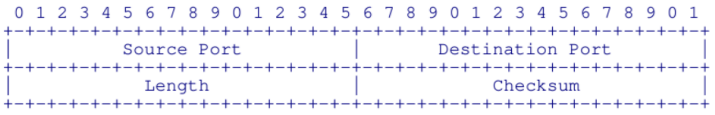
UDP

- És un protocol de nivell de transport **"orientat al datagrama"**.
- El servei UPD és **connectionless**, d'extrem a extrem i **no fiable** (si el datagrama UDP es perd, UDP no el retransmet).
- Una altra característica importat és que no te **TX Buffer** cada **operació d' escriptura del nivell d'aplicació** genera un **datagrama UDP**, ex DHCP, DNS, RIP. **Les aplicacions en temps real fan servir UDP**.

La capçalera UDP

Té mida fixa de **8 bytes** i hi ha **4 camps**:

1. **Port font i destinació** per identificar els processos que es comuniquen. El **port font pot ser 0** si no espera resposta.
2. **Length** amb la mida total del datagrama UDP (**payload UDP +8**)
3. CHECKSUM: protegeix **la capçalera i el camp de dades**. Es calcula aplicant l'algorisme de checksum conjuntament amb una pseudo capçalera, la capçalera UDP i el camp de dades. La "pseudocapçalera" té alguns camps de la capçalera IP per fer una doble comprovació que el datagrama ha arribat a la destinació correcta. En **UDP el cheksum és opcional** i en cas de no fer-se servir és igual a **0**. Si es fa servir **NAT** cal **recaluclar el checksum** de la capçalera.



UDP datagram header

Algorismes ARQ (Automatic Repeat reQuest)

L'objectiu és aconseguir un canal de comunicació **fiable** (sense errors ni dublicacions i en el mateix ordre en què s'envia). A més de proporcionar funcionalitats com **la deteccio y recuperacio d'errors** i **control de fluxos**.

Fet servir en **TCP i en Wireless**. Y TCP fa servir una variant de **Go back N**.

Números de seqüència: es fan servir per relacionar els missatges d'informació i les seves corresponents confirmacions.

Protocol orientat a la **connexió**: protocol que hi és quan hi han fases de senyalització per saber **en quina part estàs i reservar recursos**.

1. STOP AND WAIT
2. GO BACK N
3. **RETRASMISSIO SELECTIVA**

ARQ Ingredients

- Connection oriented
- Tx/Rx buffers
- Acknowledgments (ack)
- Acks can be *piggybacked* in information PDUs sent in the opposite direction
- Retransmission Timeout (RTO)
- Sequence Numbers

Recall: connection oriented protocol vs. connectionless protocol

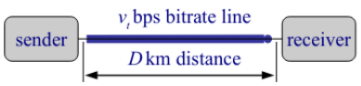
Connection oriented:
Phase 1: connection establishment
Phase 2: data stream
Phase 3: connection termination

NACK (negative ACK) to indicate some kind of error

Send ACK in data packages sent in the opposite direction

ARQ Protocols - Assumptions

- We shall focus on the the transmission in one direction.
- We shall assume a saturated source: There is always information ready to send.
- We shall assume full duplex links.
- Protocol over a line of D m distance and v_t [bps] bitrate.
- Propagation speed of v_p [m/s], thus, propagation delay of D/v_p [s].
- We shall refer to a generic layer, where the sender sends Information PDUs (I_k) and the receiver sends ack PDUs (A_k).
- Frames carrying I_k and A_k, are Tx using L_I and L_A bits, thus the Tx times are respectively: $t_t = L_I/v_t$ and $t_a = L_A/v_t$ s.



STOP AND WAIT

`Transmetre una PDU d'informació i esperar que es confirmi abans de transmetre'n una de nova.`

Tenim un **primari i un secundari** connectats per un **cable de D km de distància** i **velocitat de transmissió** de Vt (bps).

La **velocitat de propagació del senyal elèctric** en el cable és de Vp (m/s).

.

- El **nivell superior** escriu la informació que s'ha de transmetre, en un **temps de procés que suposarem 0**, el **primari** ensambla la **PDU d'informació Ik de Lt bits** amb aquesta informació, la guarda en el **buffer de transmissió** i la passa al **nivell inferior** per la seva transmissió.
- La transmissió comença **immediatament** i dura un temps Tt (segons)= L(bits)/Vt(bps)
- El **temps de propagació de cada bit** que es transmet dura Tp (segons)= D(m)/Vp(m/s). Si és el buit és la velocitat de la llum **3·10^8**.
- Quan arriba **l'últim bit de Ik al secundari**, el nivell superior **llegeix la informació rebuda en un temps de procés que suposarem 0** i el **secundari envia la confirmació Ak**. Si la **mida de la confirmació** és de La bits, el temps de transmissió de la confirmació serà Ta(segons) = La(bits) / Vt(bps).
- Els bits de la confirmació tarden un temps **tp** a arribar al **primari**. Quan arriba **l'últim bit de la confirmació**, el **primari esborra Ik** del **buffer de transmissió** i **repeteix el procés** per a una **nova PDU (Ik+1)**

.

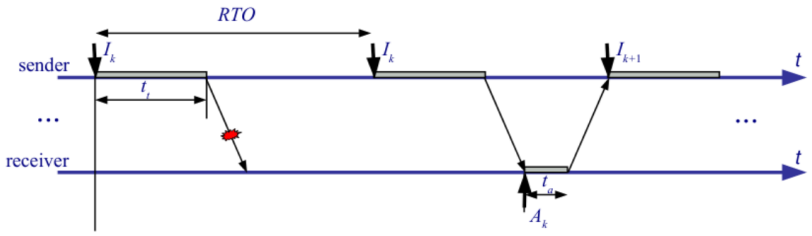
Cada vegada que el **primari envia una PDU**, activa un **temporitzador time-out To**; si el temporitzador **salta sense haver rebut la confirmació** llavors el **primari retransmet la PDU**, així si hi ha errors i el secundari descarta la PDU o no arriba es retransmet.

$$E = \frac{Tt}{Tc} = \frac{Tt}{Tt+Ta+2Tp} \approx \frac{Tt}{Tt+2Tp} = \frac{1}{1+2a}$$

on $a = \frac{Tp}{Tt}$

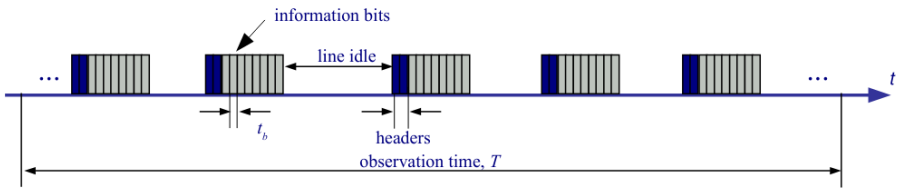
$$E = \frac{1}{1+2a}$$

Si a és molt petit (**Tp és molt més petit que Tt**) l'eficiència és pròxima al **100%**.



ARQ Protocols – Notes on computing the efficiency (channel utilization)

- Line bitrate (*velocitat de transmissió de la línia*): $v_t = 1/t_b$, [bps] — The average transmission rate of the user data in bps
- Throughput (*velocitat efectiva*) v_{ef} = number of inf. bits / obs. time, [bps]
- Efficiency or channel utilization $E = v_{ef} / v_t$ (times 100, in percentage)



$$E = \frac{v_{ef}}{v_t} = \frac{\#\text{info bits}/T}{1/t_b} = \begin{cases} \frac{\#\text{info bits} \times t_b}{T} = \frac{\text{time Tx information}}{T} \\ \frac{\#\text{info bits}}{T/t_b} = \frac{\#\text{info bits}}{\#\text{bits at line bitrate}} \end{cases}$$

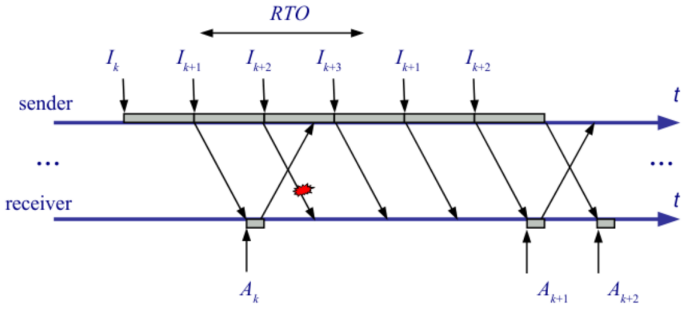
Protocols de transmissió contínua

Els protocols de transmissió contínua resolen el problema de que si la tp no és molt més petita que tc en el Stop and Wait és molt ineficient. Per fer-ho **deixen que s'enviï més d'una PDU sense confirmar**. Cada vegada que es transmet una PDU d'informació es **guarda en el buffer de transmissió** (per si s'ha de retransmetre). Quan es rep la corresponent **confirmació** s'esborra el buffer.

GO BACK N

En cas d'error el primari torna enrera fins la PDU que falta al secundari i comenci a transmetre novament a partit d'aquest punt.

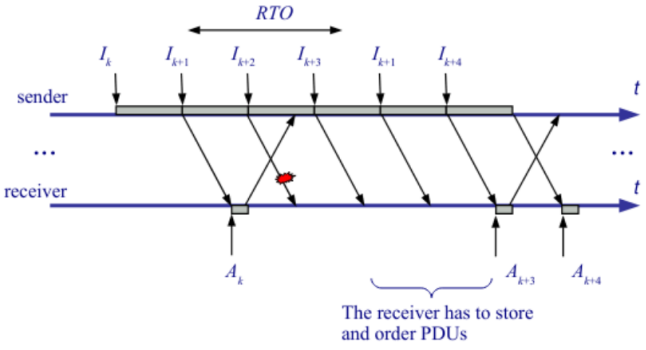
1. Les confirmacions són **acumulatives**, és a dir la confirmació Ak confirma les PDU d'informació amb **números de seqüència <= k** .
2. Si el **secundari** rep una PDU d'informació (Ik) amb **errors o fora de seqüència**:
    i. Deixa d'enviar confirmacions fins que **rep correctament la PDU** que falta
    ii. **Descarta totes les PDU** que rep amb **número de seqüència diferent a k** .
3. Quan **salta el temporitzador** de transmissió d'una PDU, el **primari retransmet la PDU Ik** i **continua** amb la tramissió **Ik+1, Ik+2...**



RETRANSMISSIÓ SELECTIVA

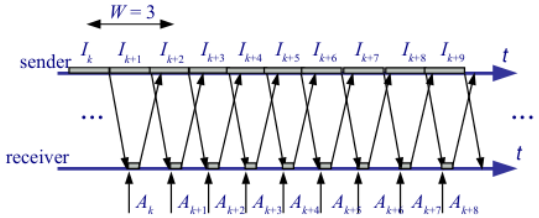
IDEA GENERAL : Que el **secundari** no descarti mai les **PDU que arriben correctament** encara que arribin **fora de seqüència**. S'ha **d'emmagatzemar i ordena les PDUs** que arriben fora de seqüència, **no poden ser llegides immediatament** del nivell superior.

1. Les confirmacions són **acumulatives**, és a dir la confirmació Ak confirma les PDU d'informació amb **números de seqüència <= k** .
2. Si el **secundari** rep una PDU d'informació (Ik) amb **errors o fora de seqüència**:
    i. Deixa d'enviar confirmacions fins que **rep correctament la PDU** que falta
    ii. **Guarda totes les PDUs** que rep amb números de seqüpencia **diferent a k** .
3. Quan **salta el TimeOut** d'una PDU, el **primari retransmet la PDU Ik**, però **no retransmet altres PDUs que ja havia enviat abans**. És a dir **només retransmet les PDUs** per les quals **salta el time-out**. .
4. Quan el **secundari** rep una retransmissió, **envia una confirmació acumulada** que **confirma** fins a **l'última PDU en seqüencia rebuda correctament**.



ARQ Protocols – Optimal Tx window

- Optimal window: Minimum window that allows the maximum throughput.
- Optimal window example:

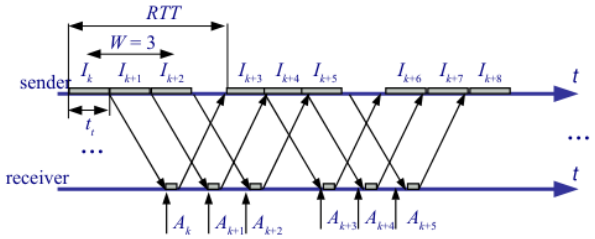


- Formula:

$$W_{opt}[PDU] = \left\lceil \frac{RTT}{t_t} \right\rceil$$

[ ] stands for ceiling function

- Non optimal window example:



ARQ Protocols – Optimal Tx window

W_{opt} is referred to as the **bandwidth delay product**:

Recall: W_{opt} is the min. win. that allows max thtroughput, i.e. v_{ef}

$$W_{opt}[\text{PDU}] = \left\lceil \frac{\text{RTT}}{t_t} \right\rceil = \left\lceil v_{ef}^{max}[\text{PDU/s}] \times \text{RTT[s]} \right\rceil$$

In **bytes**:

$$W_{opt}[\text{bytes}] \approx v_{ef}^{max}[\text{bytes/s}] \times \text{RTT[s]} = \frac{v_{ef}^{max}[\text{bps}]}{8\ [\text{bits/byte}]} \times \text{RTT[s]}$$

Example:
for $v_{ef} = 4$ Mbps and RTT $= 200$ ms we need

$$W_{opt} = v_{ef} \times \text{RTT} = \frac{4 \times 10^6\ \text{bps}}{8\ [\text{bits/byte}]} \times 200 \times 10^{-3}\ \text{s} = 100\ \text{kbyte}$$

Thus, we need a window ≥ 100 kbyte to reach 4Mbps with an RTT=200ms.

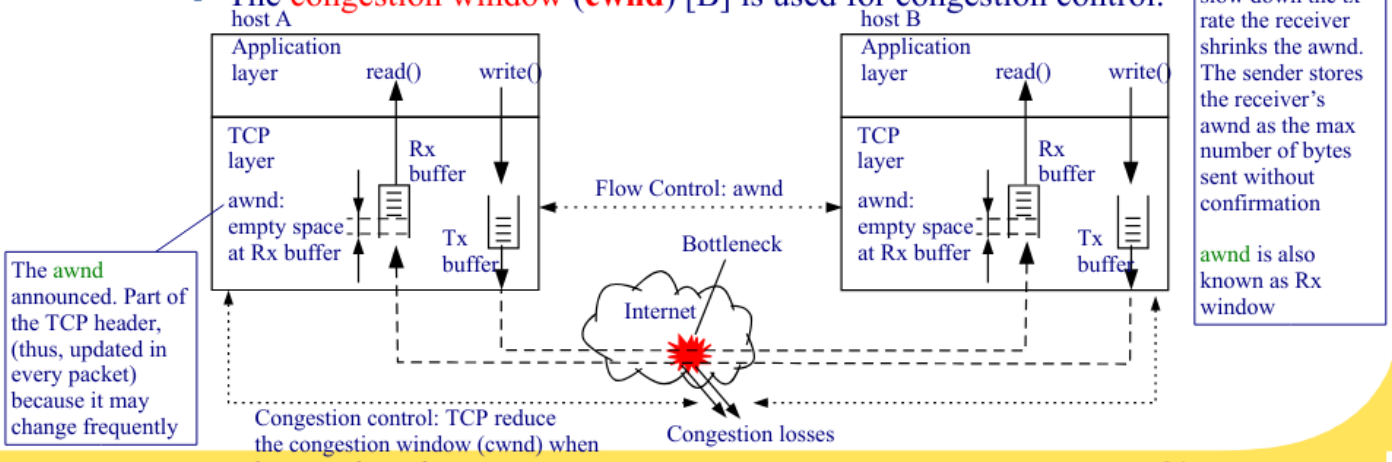
TCP

Protocol de nivell de transport de **transmissió fiable** d'informació. TCP és un protocol **d'extrem a extrem**, orientat a **la connexió** i **ARQ amb windows variable**.

- RECUPERACIO D'ERRORS, per tenir transmissió fiable. .
- CONTROL DE FLUX perquè el **primari** no enviï els segments a **més velocitat de la que pot processar-los el secundari**. Altrament hi hauria segments que es podrien perdre per vessament del buffer de recepció del secundari. .
- CONTROL DE CONGESTIÓ, perquè el **primari** no enviï els segments a **més velocitat de la que pot processar-los la xarxa**. Altrament hi hauria segments que es podrien perdre en el coll d'ampolla de la xarxa. El **coll d'ampolla** és **l'enllaç** en què els segments que envia el primari veuen una **velocitat mitjana més petita**. Les pèrdues per congestió no són un mal funcionament de TCP, a diferència del que serien les pèrdues degudes a un control de flux, que es poden evitar fàcilment. .
- SEGMENTS DE MIDA ÒPTIMA. A diferència d'UDP, TCP va a**gafant bytes de l'aplicació** per generar segments de **mida òptima**. TCP manté una variable anomenada **Maximum Segment Size(MSS)** que és la mida del que considera òptima pel **payload (camp de dades)**. Típicament la mida òptima és la major possible(per minimitzar l'overhead de les capçaleres) però que **no produeixi fragmentació a nivell IP**.
  - MSS adjusted using MTU path discovery: send datagrams with Don't Fragment bit set, and reduce MSS upon receiving ICMP error messages.

TCP Protocol – Basic operation

- TCP sender immediately sends the segments allowed by the window
- Upon segment arrival TCP receiver immediately sends sends an ack (unless delayed ack is used) without waiting for the upper layer to read the data.
  - Ack are cumulative
- ARQ window protocol, with **variable window**: [B] **wnd = min(awnd, cwnd)**
  - The advertised window (**awnd**) [B] is used for flow control.
  - The congestion window (**cwnd**) [B] is used for congestion control.



CONTROL DE FLUX

Per evitar el **vessament del buffer de Rx** (en cas que el primari enviï a una velocitat major de la que el secundari processa les dades), TCP fa servir la **advertised window**. Quan TCP rep un segment **guarda el valor de la finestra advertida en la variable awnd**. El **primari** no pot enviar mai **més bytes sense confirmar dels que diu awnd**. Si el buffer d'RX **s'omple**, el **secundari** enviarà una finestra advertida **igual a 0** i el **primari es quedarà bloquejat** fins que el secundari torni a enviar una **finestra advertida major de 0**.

```
Resum: awnd depen de lo que hi ha al buffer de Rx per a que no perdem merdes
```

TCP Protocol – Delayed acks

- TCP connections can be classified as:
  - Bulk: (e.g. web, ftp) There are always bytes to send. TCP send MSS bytes.
  - Interactive: (eg. telnet, ssh) The user interacts with the remote host.
- In bulk connections sending an ack every data segment can unnecessarily send too many small segments. Solution: Delayed acks.

Delayed ack. It is used to reduce the amount of acks. Consists of sending 1 ack each 2 MSS segments whenever possible under the time limit of 200 ms. Acks are always sent in case of receiving out of order segments.

Remark: After SYN tcpdump extracts the initial sequence number from the sequence numbers

tcpdump example (bulk transfer):

It's a download

```
...
11:27:13.798849 147.83.32.14.ftp > 147.83.35.18.3020: P 9641:11089(1448) ack 1 win 10136 (DF)
11:27:13.800174 147.83.32.14.ftp > 147.83.35.18.3020: P 11089:12537(1448) ack 1 win 10136 (DF)
11:27:13.800191 147.83.35.18.3020 > 147.83.32.14.ftp: . 1:1(0) ack 12537 win 31856 (DF)
11:27:13.801405 147.83.32.14.ftp > 147.83.35.18.3020: P 12537:13985(1448) ack 1 win 10136 (DF)
11:27:13.802771 147.83.32.14.ftp > 147.83.35.18.3020: P 13985:15433(1448) ack 1 win 10136 (DF)
11:27:13.802788 147.83.35.18.3020 > 147.83.32.14.ftp: . 1:1(0) ack 15433 win 31856 (DF)
...
```

Lenght (11089 - 9641 = 1448)

TCP flags

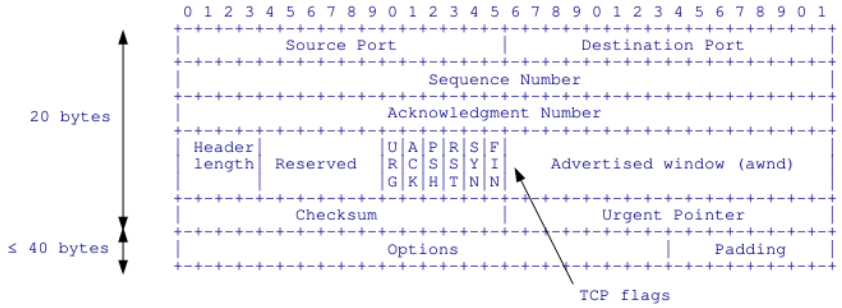
DF flag in IP header set.

timestamp | src IP addr/port | dst IP addr/port | seq. num:next seq num (bytes) | ack | awnd

```
11:27:13.798849 147.83.32.14.ftp > 147.83.35.18.3020: P 9641:11089(1448) ack 1 win 10136 (DF)
```

Unit 4. TCP

TCP Protocol – TCP Header

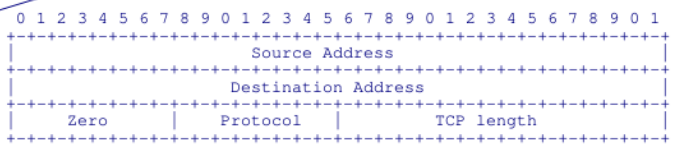


Note: The length of the data section is not specified because it is not required at implementation level (the TCP stack just does a sizeof(buf) to find out how much data a segment has).
However, it can be calculated by subtracting the combined length of the segment header and IP header from the total IP packet length specified in the IP header.

Conversely, the UDP includes the length because this data is needed at implementation level (obtaining it from the IP header is not acceptable at implementation level because it implies to mix data from different layers and because it cannot be assumed that it runs over IP).

- Variable size: Fixed fields of 20 bytes + options (15x4 = 60 bytes max.).
- The header length is in 32-bit words. Between 5 and 15).
- Reserved for future protocol extensions (currently 3bits already in use).
- Like in UDP, the checksum is computed using an IP pseudo-header:

In TCP it is mandatory



TCP Protocol – TCP Flags

- URG (Urgent): The Urgent Pointer is used. It points to the first urgent byte. Rarely used. Example: ^C in a telnet session.
- ACK: The ack field is used. Always set except for the first segment sent by the client.
- PSH (Push): The sender indicates to "push" all buffered data to the receiving application. Most BSD derived TCPs set the PSH flag when the send buffer is emptied.
- RST (Reset): Abort the connection.
- SYN: Used in the connection setup (*three-way-handshaking, TWH*). Only the first packet sent from each end should have this flag set (bcs it synchronizes sequence numbers). Some other flags and fields change meaning based on this flag.
- FIN: Used in the connection termination. Only set in the last packet from the sender.

If set (1), this is the initial sequence number, thus, the receiver has to acknowledge this sequence number plus one.
Otherwise (0), this is the accumulated seq. num. of the first data byte of this segment for the current session.

TCP Protocol – TCP Options

"suggest" to the other end

- Maximum Segment Size (MSS): Used in the TWH to initialize the MSS. Only with SYN set.
  - Common values in IPv4:
    - **MSS 1460**: 1500-40 (MTU [1500B] - (IPv4 header [20B] + TCP header without options [20B]))
    - **MSS 1448**: 1500-52 (MTU [1500B] - (IPv4 header [20B] + TCP [20B] with timestamp options [10B] and 2 nops [1B])). Currently the most common value.
- Window Scale factor: Used in the TWH. The awnd is multiplied by $2^{\text{Window Scale}}$ (i.e. the window scale indicates the number of bits to left-shift awnd). It allows using awnd larger than 2^{16} bytes. Only with SYN set.
- Timestamp: Used to compute the Round Trip Time (RTT). Is a 10 bytes option, with the timestamp clock of the TCP sender, and an echo of the timestamp of the TCP segment being ack. Used to set RTO.
- SACK: In case of errors, indicate blocks of consecutive correctly received segments for Selective Retransmission. Only with SYN set.
- NOP: Used for padding (1B).
- Etc. (~35 in total; may be extended in the future)

Remark: In TCP the options are **frequently** used (contrary to IP)

ESTABLIMENT I TERMINACIÓ D'UNA CONNEXIO TCP

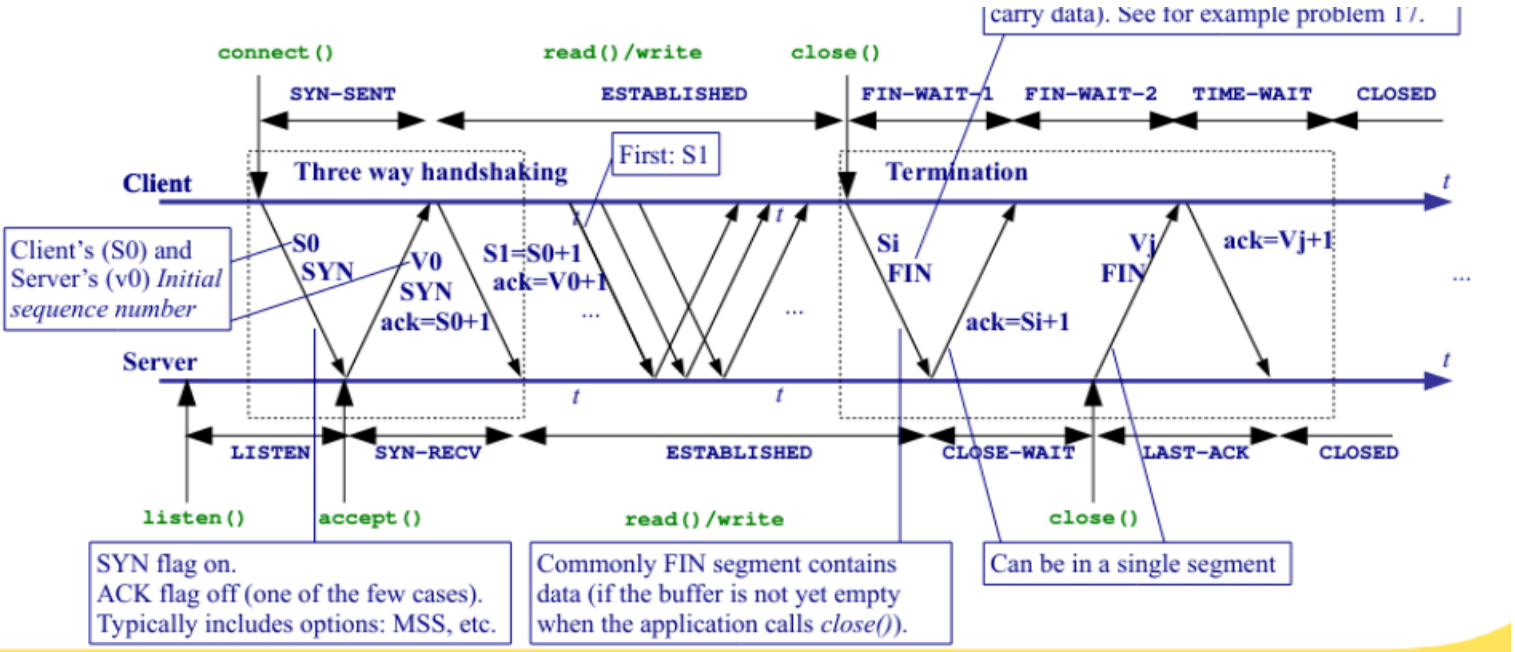
L'estabilment d'una connexió TCP es coneix amb el nom de **three-way-handshaking** i consisteix sempre en **l'intercanvi de tres segments que no porten dades** (només la capçalera TCP): **SYN/SYN+ack/ack**

- El primer segment (SYN) sempre l'envia **el client**. Aquest segment té típicament un **port well known com a destinació** i un **port efímer com a port origen**. La característica més important és que té el **flag de SYN activat**. Aquest segment és un dels pocs casos en què, per motius obvis **no es confirma res** (el flag d'ACK ha d'estar desactivat). Aquest segment típicament porta opcions com ara **l'MSS o les opcions timestamp, window scale factor o SACK**. Aquest segment també porta **l'initial sequence number** que és el número de seqüència inicial que es farà servir per **identificar els bytes de dades enviats pel client**. Aquest número és un **número aleatori de 32 bits**. .
- El segon segment (SYN + ack) l'envia **el servidor**. També té el **flag de SYN activat** i porta **l'initial sequence number** per identificar els bytes de dades **enviats pel servidor**. Els **segments de SYN**, tot i no portat cap byte de dades, consumeixen un número de seqüència. **L'ack d'aquest segment**, per tant, té un valor igual a l'initial sequence number del client més 1. .
- Finalment **el client confirma la recepció de SYN+ack** enviant la confirmació amb **l'initial sequence number del servidor més 1**.

La terminació de la connexió es produeix amb **l'intercanvi de 3 o 4 segments**: FIN/ack en un sentit i FIN/ack en el sentit contrari. .

El segment de **FIN** s'envia quan **l'aplicació fa la crida close()**. Igual que el SYN, el FIN consumeix un número de seqüència. Una diferència important és que el **segment de FIN pot portar dades**. De fet, és normal que això passi si encara hi ha bytes de dades per enviar en el buffer de transmissió de TCP quan l'aplicació fa la crida close(). Si a l'altre extrem ja ha executat la crida close(), aleshores es diu que **la connexió està half closed**, i només es poden enviar dades en unsentit(el que encara no ha enviar el segment de FIN).

A diferència del three-way-handshaking, on el client envia sempre el primer segment de SYN, el **primer segment de FIN pot enviar-lo el client o el servidor**.



NÚMEROS DE SEQUÈNCIA EN TCP

En TCP els números de seqüència i les finestres es mesuren en **bytes**. És a dir, si un segment porta **S bytes**, el número de seqüència del pròxim segment **s'incrementarà en S**. El número de seqüència que porta la capçalera identifica **el primer byte de dades del segment**. Inicialment **la finestra val MSS** (només permet enviar **un segment**) i s'incrementa amb **MSS** cada vegada que arriba **un ack que confirma noves dades**.

Les confirmacions porten un **valor igual al número de sequência** que porta el segment que **confirmen més el nombre de bytes de dades del segment**. Per exemple, si el segment amb número de seqüència **Si porta MSS bytes**(segment de mida màxima), aleshores la confirmació porta el valor: **ack = Si + MSS**. Quan el **primari** rep la **confirmació** interpreta que tots els bytes identificats amb **números de seqüència inferiors al de l'ack**, **han arribat correctament al secundari** i els **esborra del buffer de transmissió**.

TCP Protocol – tcpdump example (web page download)

win is recommended to be a multiple of mss for optimisation reasons, but it is just a recommendation.

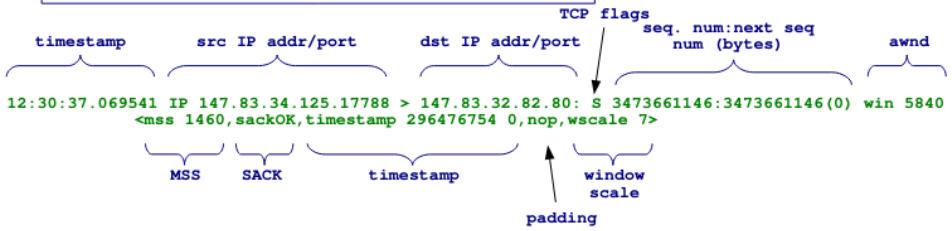
```
1  12:30:37.069541 IP 147.83.34.125.17788 > 147.83.32.82.80: S 3473661146:3473661146(0) win 5840 <mss 1460,sackOK,timestamp
                   296476754 0,nop,wscale 7>
2  12:30:37.070021 IP 147.83.32.82.80 > 147.83.34.125.17788: S 544373216:544373216(0) ack 3473661147 win 5792 <mss
                   1460,sackOK,timestamp 1824770623 296476754,nop,wscale 2>
3  12:30:37.070038 IP 147.83.34.125.17788 > 147.83.32.82.80: . ack 1 win 46 <nop,nop,timestamp 296476754 1824770623>
4  12:30:37.072763 IP 147.83.34.125.17788 > 147.83.32.82.80: P 1:602(601) ack 1 win 46 <nop,nop,timestamp 296476754 1824770623>
5  12:30:37.073546 IP 147.83.32.82.80 > 147.83.34.125.17788: . ack 602 win 1749 <nop,nop,timestamp 1824770627 296476754>
6  12:30:37.075932 IP 147.83.32.82.80 > 147.83.34.125.17788: P 1:526(525) ack 602 win 1749 <nop,nop,timestamp 1824770629 296476754>
7  12:30:37.075948 IP 147.83.34.125.17788 > 147.83.32.82.80: . ack 526 win 54 <nop,nop,timestamp 296476755 1824770629>
8  12:30:53.880704 IP 147.83.32.82.80 > 147.83.34.125.17788: F 526:526(0) ack 602 win 1749 <nop,nop,timestamp 1824787435 296476755>
9  12:30:53.920354 IP 147.83.34.125.17788 > 147.83.32.82.80: . ack 527 win 54 <nop,nop,timestamp 29680966 1824787435>
10 12:30:56.070200 IP 147.83.34.125.17788 > 147.83.32.82.80: F 602:602(0) ack 527 win 54 <nop,nop,timestamp 296481504 1824787435>
11 12:30:56.070486 IP 147.83.32.82.80 > 147.83.34.125.17788: . ack 603 win 1749 <nop,nop,timestamp 1824789625 296481504>
```

Data TWH stream

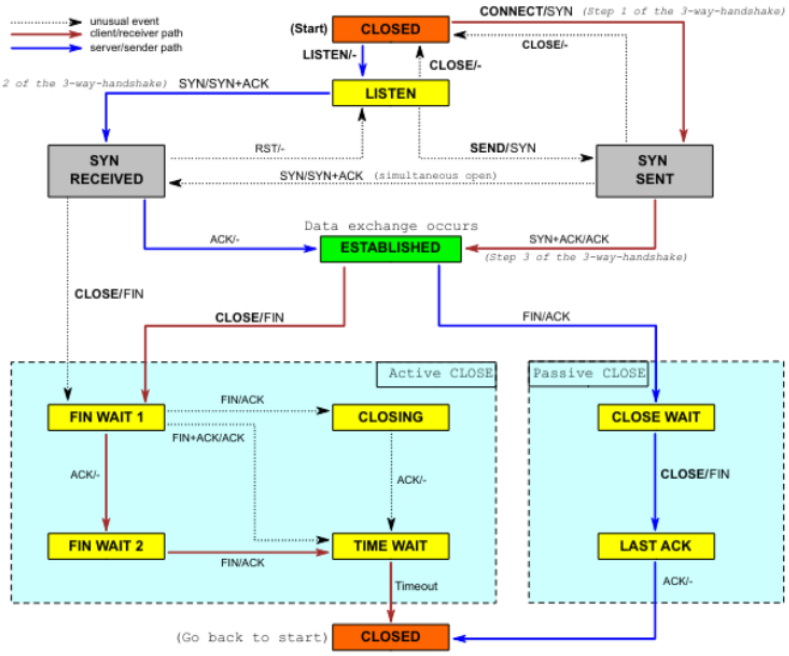
Termination

1, 2, 3: Three way-handshake
4: The client (147.83.34.125) requests a web page (e.g. index.html)
5: The server (147.83.32.82) ACKs the client's request
6: The server sends the web page in a single segment
7: The client ACKs the segment containing the web page
8: The server starts the connection termination
9: The client ACKs server's termination
10: The client starts the connection termination
11: The server ACKs client's termination

Recall: Flags
S SYN
· ACK only
P Push
F FIN



TCP Protocol – State diagram (simplified)

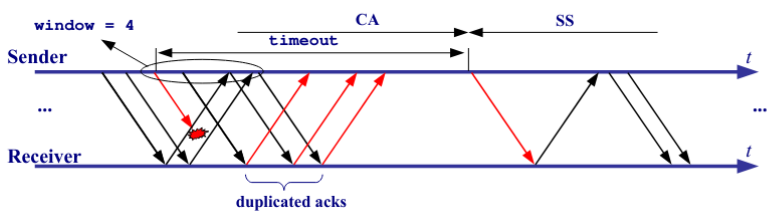


TCP Protocol – Slow Start / Congestion Avoidance (SS/CA)

- Variables:
  - snd_una: First non ack segment (head of the TCP transmission queue).
  - ssthresh: Threshold between SS and CA.

```
Initialization:
        cwnd = MSS ; NOTE: RFC 2581 allows an initial window of 2 segments.
        ssthresh = infinity ;

Each time an ack confirming new data is received:
        if(cwnd < ssthresh) {
                cwnd += MSS ;                   /* Slow Start */
        } else {
                cwnd += MSS * MSS / cwnd ; /* Congestion Avoidance */
        }

When there is a time-out:
        Retransmit snd_una ;
        ssthresh = max(min(awnd, cwnd) / 2, 2 MSS) ;
        cwnd = MSS ;
```

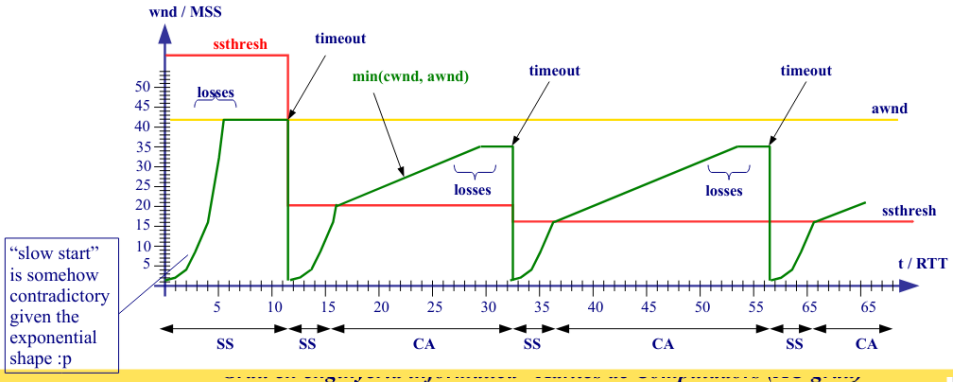
Time-out Example:



TCP Protocol – Slow Start / Congestion Avoidance (SS/CA)

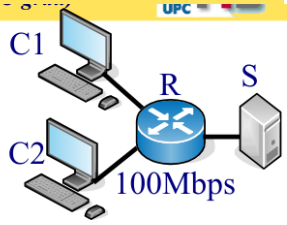
- During SS cwnd is rapidly increased to the "operational point".
- During CA cwnd is slowly increased looking for more available "bandwidth".

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

Each time an ack confirming new data is received:
        if(cwnd < ssthresh) {
                cwnd += MSS ;                   /* SS */
        } else {
                cwnd += MSS * MSS / cwnd ; /* CA */
        }

When there is a time-out:
        Retransmit snd_una ;
        ssthresh = max(min(awnd, cwnd) / 2, 2 MSS) ;
        cwnd = MSS ;
```



"slow start" is somehow contradictory given the exponential shape :p

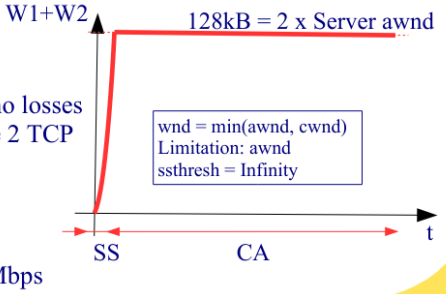
Unit 4. TCP



TCP Protocol – Evaluation without losses

- Preliminaries:
  - TCP sends the entire window, W (in several segments)
  - The segments accumulate in the queues of the interfaces where there are bottlenecks
  - Steady state: the TCP connection started time ago
  - In general, we can assume that, on the average, is fulfilled vef = W / RTT
  - If there are no losses, W will be awnd, otherwise W follows a "saw tooth"

- Example without losses: C1 and C2 send to S, each with a TCP connection. Server awnd = 64kB. Router queues ≧ 128kB.
  - The bottleneck is the link R-S
  - For each connection vef = 100/2 = 50 Mbps
  - If the propagation delays in the links are negligible and no losses occur in the queue of the router there will be 128 kB (the 2 TCP windows)
    - The RTT is the time in the queue of the router:
      - RTT=128 kB/100 Mbps = 10,24 ms
    - Check that vef=W/RTT = 64 kB/10,24 ms = 50 Mbps

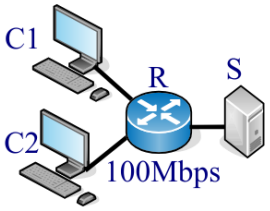


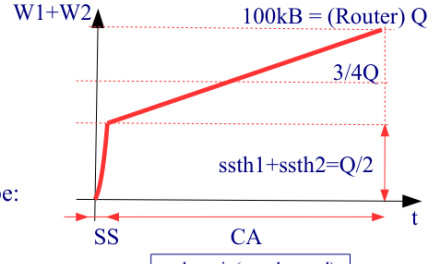
TCP Protocol – Evaluation with losses

- Example with losses: C1 and C2 send to S, each with a TCP connection. Server awnd = 64kB. Assume now that the interface queue of the router is limited to Q = 100 kB
  - The bottleneck is the link R-S
  - For each connection vef = 100/2 = 50 Mbps
  - There will be losses, because when both TCP windows add to 100kB, there will be no space left in the router queue.
  - The figure shows a possible evolution of the queue in the router, which stores the window of both connections: W1+W2. When the queue is full, both connections have losses and reduce the ssth to the half. Therefore, the average queue size in the router will be, approximately:

    (Q/2+Q)/2=3/4Q=75 kB
  - Thus, the average RTT will be:
    - $\overline{RTT}$=75 kB/100 Mbps = 6 ms
  - Note that the average window of each connection will be:

    $\overline{W1}$=$\overline{W2}$=75 kB/2=37,5 kB
  - Check that vef=$\overline{W}$/$\overline{RTT}$ = 37,5 kB/6 ms = 50 Mbps





Tests

En un protocol de finestra si la finestra de transmissió val 1 es comporta igual que Stop-and-Wait(mirar què és Stop-and-Wait).

Augmentant la mida de la finestra més enllà de la finestra òptima no es guanya eficiència.

Sempre cal un temporitzador de retransmissió (RTO).

Les capçaleres UDP i TCP tenen un camp amb el port font i el port destinació i tenen un camp checksum.

Respecte TCP, el temporitzador de retransmissió, RTO, s'actualitza a partir del càlcul que es fa del round trip time RTT. Hi ha algnes opcions que només es fan servir durant l'establiment de la connexió (three-way-handshake). El slow start treshold (mirar què és) no pot tenir un valor inferior a 2 segments (2 MSS bytes). És possible enviar una finestra anunciada (advertized window) igual a 0 bytes.

En un switch ethernet on hi ha configurades 2 VLANs i un port en mode trunk és possible que una trama que arriba per el trunk es reenviï per més d'un port, és possible que una trama que arriba per el trunk es reenvïi per tots els ports d'una mateixa VLAN.

Respecte Ethernet en un switch hi pot haver ports full duplex i half duplex simultàniament, les trames Ethernet tenen un camp amb l'adreça destinació i un camp amb l'adreça font.

Se puede abortar una conexión TCP enviandon un segmento con el flag R activo.

DNS (Domain Name System)

`Objectiu : permetre que els usuaris d'Internet puguin fer servir noms en comptes d'adreces IP`

1. Segueix el paradigma client/servidor amb nivell de transport TCP/UDP amb **port well-known 53**.
2. Hi ha una base de dades amb els noms i les adreces per poder fer la resolució.
3. El sistema de noms està organitzat en una jerarquia que permet distribuir la base de dades arreu d'Internet.
4. És un protocol d'aplicació.

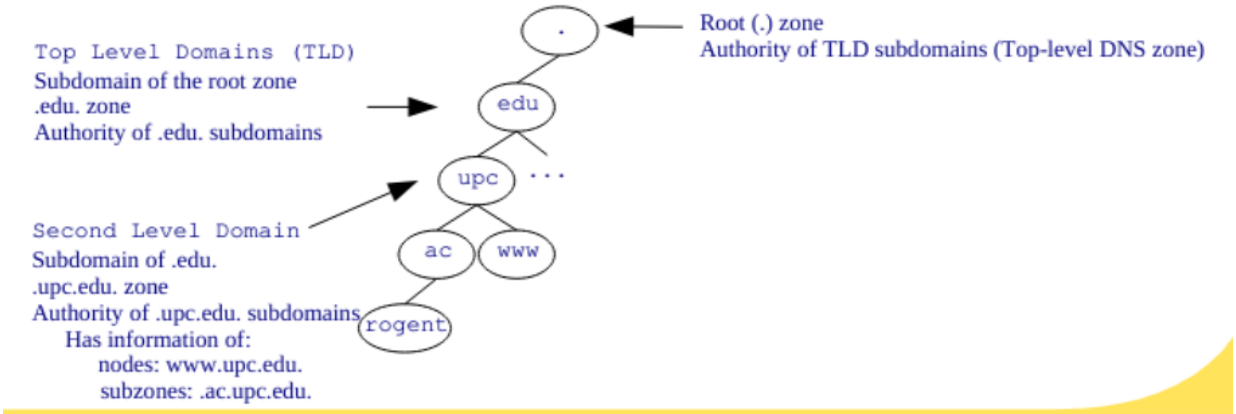
Jerarquia

La jerarquia del sistema de noms està organitzada en **"dominis" o "zones"**. El **domini arrel no té nom** i d'ell pengen els **top level domains (TLD)**. Cada un dels TLD té **un administrador** que delega part del domini en **"sub-dominis"**.

El nom s'escriu començant pel **host** i separant **els dominis per punts fins el TLD**. El nom es pot fer **acabar en un punt** per indicar que **s'especifiquen tots els dominis** (fully qualified domain name).

Cada administrador d'un domini (o subdomini) ha de **mantenir part de la base de dades de DNS en un "servidor primari"** i **un o més "servidors secundaris" (de backup)**. En l'argot DNS, aquests servidors també es coneixen com a **"autoritat" (authority) del domini**. En aquests servidors hi ha d'haver **el nom i adreça dels hosts** que pengen del seu domini i **el nom i adreça dels servidors primaris i secundaris** de les autoritats dels subdominis que hagi delegat.

Actualment hi ha 13 servidors que tenen les adreces dels TLD anomenats root-servers i estan distribuïts arreu del món.

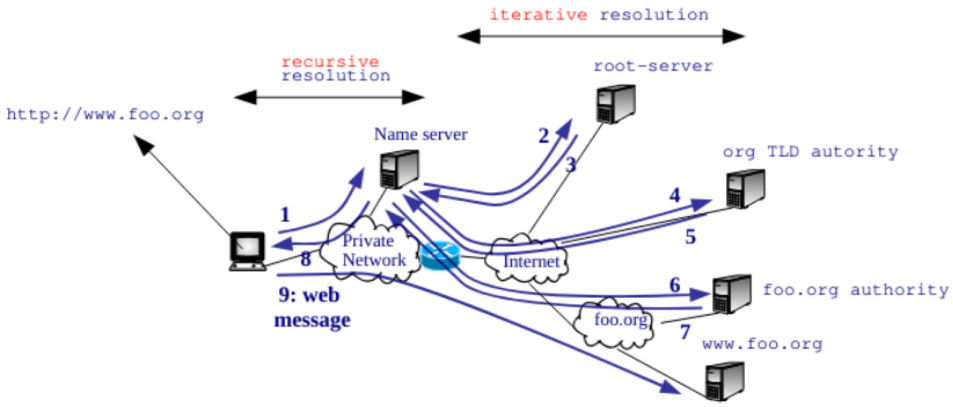


Accés a la base de dades de DNS

Exemple: accés a mauriciabad.com.

1. El **host envia el nom que es vol resoldre** al seu servidor de DNS. .
2. El **servidor envia la petició a un root-server**, que li retorna **l'adreça del servidor de noms del TLD**, domini com. .
3. Després el **servidor s'adreça al servidor del domini com**, que retorna **l'adreça del servidor de noms del second level domain**, domini mauriciabad.com. .
4. Després el **servidor s'adreça al servidor del domini mauriciabad.com**, que li retorna **l'adreça buscada**. .
5. Finalment el **servidor retorna l'adreça buscada al host** que ho havia sol·licitat. .

La **resolució que fa el host** s'anomena RECURSIVA. La **resolució que fa el servidor de noms del host** s'anomena ITERATIVA perquè consulta iterativament els servidors dels dominis fins que resol l'adreça buscada. Una característica molt imporant és la CACHING. Consisteix en que el servidor de noms del host guardarà l'adreça sol·licitada.



- All DNS messages have the same format:
  - Header: type of message.
  - Question: What is to be resolved.
  - Answer: Answer to question.
  - Authority: Domain authority names.
  - Additional: Typically, the authority name's addresses.

```
+-----------------------------------------------+
|            Header (12 bytes)                  |
+-----------------------------------------------+
/            Question (variable)                /
+-----------------------------------------------+
/            Answer (variable)                  /
+-----------------------------------------------+
/            Authority (variable)               /
+-----------------------------------------------+
/            Additional (variable)              /
+-----------------------------------------------+
```

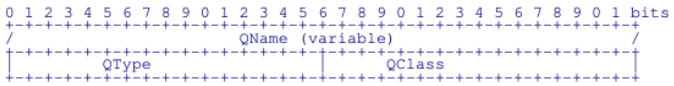
Header

En el header hi ha els següents camps:

- Identification, que permet relacionar els missatges de query (pregunta) i reply (resposta) amb un **número random de 16 bits**.
- **Flags**, on els més importants són:
  - Query-response (flag QR): indica si el paquet és de **query (0)** o **resposta (1)**.
  - Authoritative Answer (flag AA): indica si ha respost **l'autoritat del domini**, o si la resposta estava en l**a cache del servidor** on s'ha fet la pregunta.
  - Rescursion-Desired (flag RD): indica si es desitja que la resolució sigui **recursiva**.

DNS – Messages: Question

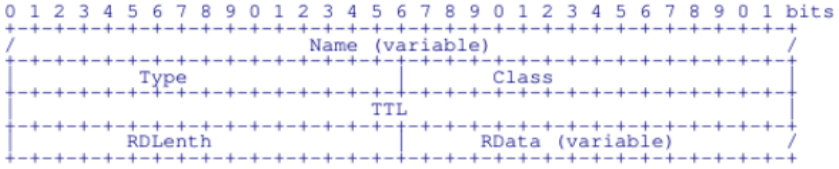
- QName: Indicates the name to be resolved.
- QType: Indicates the question type:
  - Address, A.
  - Name Server, NS.
  - Pointer, PTR: For an inverse resolution.
  - Mail Exchange, MX: Domain Mail Server address.
- Qclass: For Internet addresses is 1.



QName codification example of `rogent.ac.upc.edu`

DNS – Messages: Resource Records (RRs)

- The fields Answer, Authority and Additional are composed of RRs:
  - Name, Type, Class: The same as in the Question field.
  - TTL (Time To Live): Number of seconds the RR can be cached.
  - RDLenth: RR size in bytes.
  - Rdata: E.g. An IP address if the Type is 'A', or a name if the Type is 'NS', 'MX' or 'CNAME'.



DNS – Messages: Example

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

Query message:

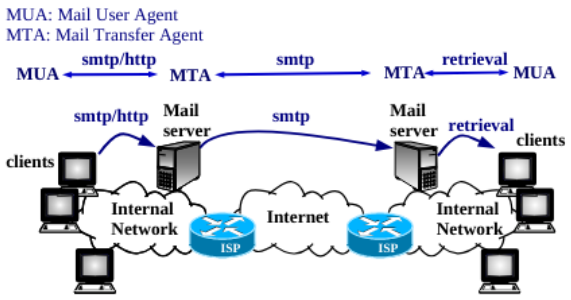
- **36388: Identifier.**
- **+: Recursion-Desired is set.**
- **A?: Qtype = A.**
- **ns.uu.net.: Name to resolve.**

Response message:

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

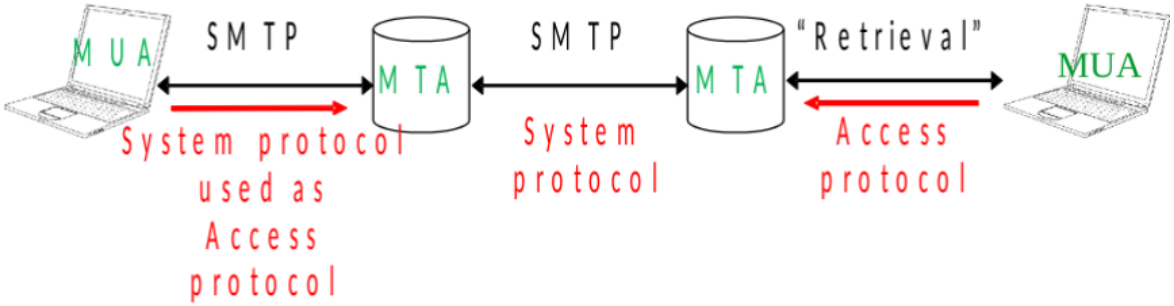
Email

- Electronic mail (email): One of the first applications used in the Internet to electronic messaging.
- Components:
  - Transport layer: TCP, well-known port: 25.
  - Application layer protocol: Simple Mail Transfer Protocol (SMTP). First defined by RFC-821 and last updated by RFC-5321.
  - Retrieval protocols (IMAP, POP, HTTP).



- **MUA: Mail User Agent**
- **MTA: Mail Transfer Agent**

Email – Protocols



- **"Retrieval" protocols (mailbox access):**
  - Post Office Protol (POP3)
  - Internet Message Access Protocol (IMAP)
- **Simple Mail Transfer Protocol (SMPT)**

Email - retrieval protocols

- Post Office Protocol (POP), RFC-1939:
  - POP server listens on well-known port 110
  - User normally deletes messages upon retrieval.

- Internet Message Access Protocol (IMAP) RFC-3501:
  - IMAP server listens on well-known port 143
  - Messages remain on the server until the user explicitly deletes them.
  - Provide commands to create folders, move messages, download only parts of the messages (e.g. only the headers)

- Web based Email (HTTP)
  - A web server handles users mailboxes. User agent is a web browser, thus, using HTTP to send and retrieve email messages.

MIME: transfer encoding

Ways to encode content: (to “get through” a 7 bit transport)

- Quoted-Printable:
  - The majority of text is 7 bits, transform some characters € → =E4
  - The result “almost" legible without decoding. Depends on table (charset)
- Base64:
  - 3 bytes (24 bits) <=> 4 ASCII (32 bits)
  - A-Za-z0-9+/=
  - '=' as padding, other are ignored (\r, \n, …)
- Binary: No encoding: any character and lines of any length
- 7Bit: No character encoding (all 7 bits) and lines of appropriate length
- 8Bit: No character encoding (8 bits) and lines of appropriate length
- In the heading:

```
MIME-Version: 1.0
Subject: =?iso-8859-1?Q?acentuaci=F3n=20t=EDpica?=
```

Web – links

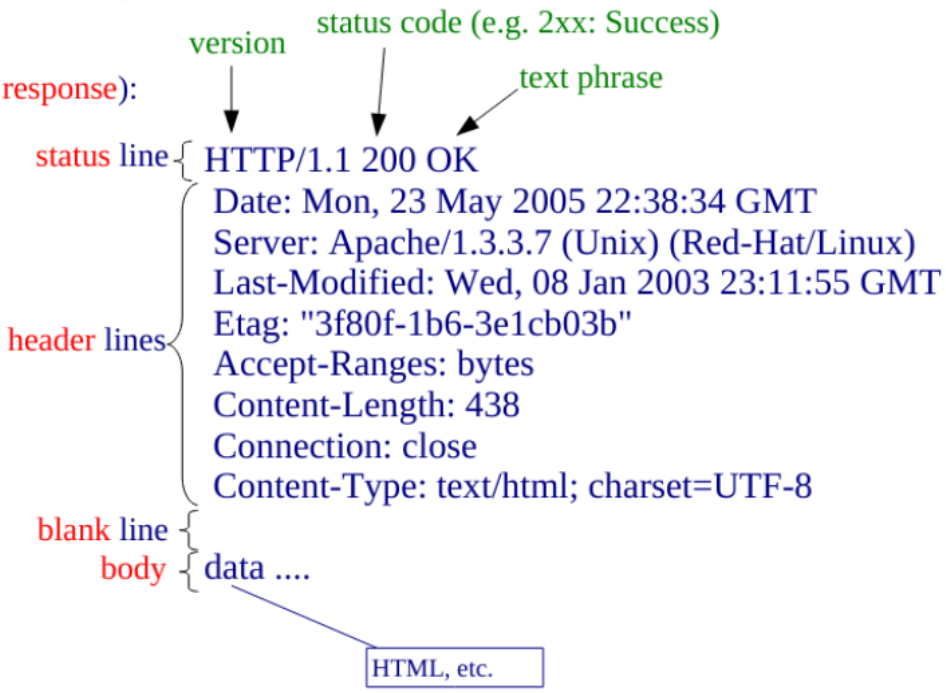
- Uniform Resource Identifier (URI) RFC3986
  - Generic syntax to identify a resource.
- Uniform Resource Locator (URL) RFC1738
  - Subset of URIs identifying the locating a resource in the Internet.
- The URL general syntax is

  scheme://username:password@domain:port/path?query_string#fragment_id
  - scheme: Purpose, and the syntax of the remaining part. http, gopher, file, ftp...
  - domain name or IP address gives the destination location. The port is optional.
  - query_string: contains data to be passed to the server.
  - fragment_id: specifies a position in the html page.
  - Examples:
    - http://tools.ietf.org/html/rfc1738
    - http://147.83.2.135
    - http://studies.ac.upc.edu/FIB/grau/XC/#Practs
    - file:///home/llorenc/gestio/2010/cd/autors.html
    - http://www.amazon.com/product/03879/refs9?pf_ra=ATVPD&pf_rd=07HR2

Web – HTTP Messages, RFC2616

- **Methods:**
  - GET Typical command. Requests an object.
  - POST Request an object qualified by the data in the body. This data is the contents of the HTML form fields, provided by the client.
  - HEAD the server returns only the header
  - OPTIONS request communication options
  - PUT store entity
  - PATCH modify an existing resource
  - DELETE delete entity
  - TRACE final recipient echoes the received message back
  - CONNECT used with a proxy
- NOTES
  - Most used: GET, POST
  - Safe and mandatory: GET, HEAD —— Any server must implement these methods at least

- Server (HTTP response):

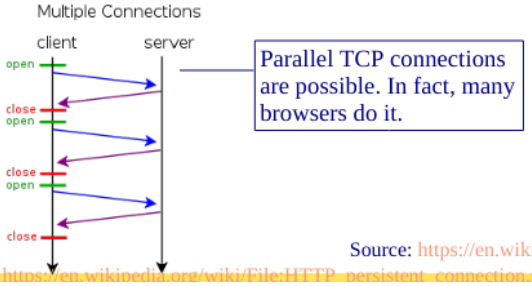


- Status codes:
  - 1xx informational response – the request was received, continuing process
  - 2xx successful – the request was successfully received, understood, and accepted
  - 3xx redirection – further action needs to be taken in order to complete the request
  - 4xx client error – the request contains bad syntax or cannot be fulfilled
  - 5xx server error – the server failed to fulfill an apparently valid request

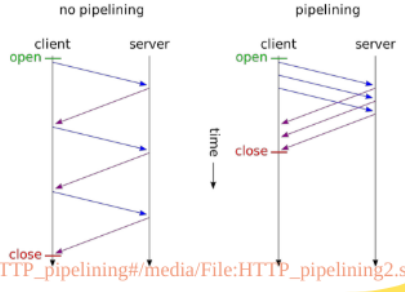
- Some well-know status codes:
  - 200 OK
  - 304 Not Modified — Used by proxies
  - 400 Bad Request
  - 403 Forbidden
  - 404 Not Found
  - 500 Internal Server Error
  - 502 Bad Gateway

Web – Persistent/non Persistent connections

- Non persistent (default in HTTP/1.0): The server close the TCP connection after every object. E.g, for an html page with 10 jpeg images, 11 TCP connections are sequentially opened.
- Persistent (default in HTTP/1.1) : The server maintains the TCP connection opened until an inactivity time. In the example: All 11 objects would be sent over the same TCP connection.
- Persistent connections with pipelining (supported only in HTTP/1.1): The client issues new requests as soon as it encounter new references, even if the objects have been not completely downloaded. In the example: All 11 objects would be sent over the same TCP connection.



Source: https://en.wikipedia.org/wiki/File:HTTP_persistent_connection.svg



Source: https://en.wikipedia.org/wiki/HTTP_pipelining#/media/File:HTTP_pipelining2.svg

Universal Coded Character Set
Unicode

All characters from all written languages + math + emoticons +
+=Universal Character set (ucs)

Encoding: UCS-4 bytes (fixed length)

Proportional spacing, language independent

Unicode consortium: synchronized with ISO,

- Unicode 9.0.0 (7/2016): 140,186 symbols (2023/05/29)
- U+hex code: U+0020 = ' '

Character Encoding: Universal Transformation Format (UTF)

- Difficulty or impossibility to transport 8 o 16 bits data in Internet protocols:
- **UTF-8**, UTF-16, UTF-32 (variable length)

• UTF-8 Encoding

- Determine high-order bits from the number of octets
- Fill in the bits marked x

Char. number range (hexadecimal)	UTF-8 octet sequence (binary)
0000 0000–0000 007F	0xxxxxxx
0000 0080–0000 07FF	110xxxxx 10xxxxxx
0000 0800–0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000–0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

• Self-synchronization => it is possible to identify any character at any time (no need to restart reading from the beginning of the communication -it is not the case in ASCII):
The number of bytes of the character is determined by the combination of the initial bits

- Example
  - character: €
  - code point: U+20AC
  - code point in binary (12 bits): 10 0000 1010 1100
  - 3 code units required:
  - UTF-8: 11100010 10000010 10101100
  - UTF-8 in hex: E282AC