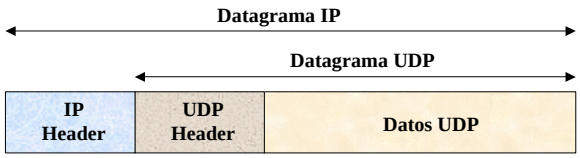
- **Tema 3: Transporte: TCP, UDP y APIs (5 semanas)**
  - UDP
  - Funcionalidades de TCP
    - Control de errores
    - Control de flujo
    - Control de la congestión
  - Grafo de estados TCP
    - 3wHS
    - Finalización de la conexión
  - Sockets
    - Mapeo de llamadas a sockets con el grafo de estados de TCP
    - Servidores concurrentes e interactivos

- **Transporte:** nivel 4 cuyas funciones principales son:
  - Proporcionar conectividad extremo-a-extremo entre hosts
  - Sólo se incluyen los protocolos de transporte en los hosts (nunca en los routers a no ser que sean por alguna razón especifica, e.g. Gateways de aplicación)
  - Proporcionan estructuración de la información (segmentos TCP y datagramas UDP)
  - Multiplexación/demultiplexación de aplicaciones en transporte (concepto de puertos como identificador de las aplicaciones)
  - Uso de comunicaciones fiables y no fiables a nivel de transporte:
    - TCP: detección y control de errores y control de flujo y de la congestión extremo a extremo (para aplicaciones de datos)
    - UDP: sólo detección de errores extremo a extremo (para aplicaciones en tiempo real como son audio y vídeo)

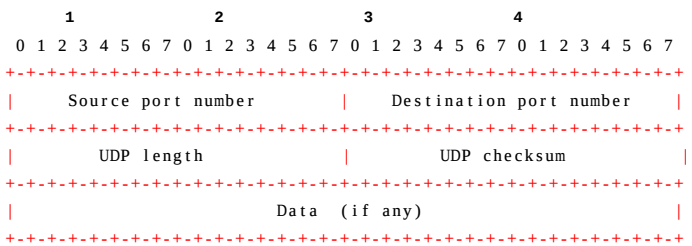
Tema 3: Transporte: UDP

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



Tema 3: Transporte: UDP

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



Tema 3: Transporte: UDP

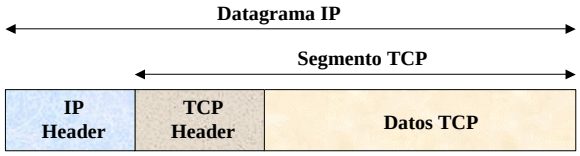
- **UDP (User Datagram Protocol):**
  - **UDP checksum**: detector de errores que aplica a TODO el datagrama (recordar que checksum IP sólo cubría la cabecera IP)
    - Para calcularlo necesitamos que la longitud del datagrama UDP sea un número par de octetos (para poder agruparlos en words de 16-bits)
    - Si no hay un número par entonces hacer padding (añadir un byte de 0s al final del datagrama)
    - El checksum además de la cabecera UDP cubre ciertos campos de la cabecera IP para hacer un "double-checking". Esos campos son:
      - Direcciones IP, protocol field, total IP length
    - Para calcular el checksum, UDP crea una pseudo-cabecera con estos campos además de los campos del datagrama UDP
    - Si se detecta un error a nivel 4, el datagrama UDP se descarta y NO se genera ningún tipo de mensaje hacia el origen
    - **¿Porqué el checksum?** para detectar que NADIE ha modificado el datagrama UDP

Tema 3: Transporte: UDP

- **UDP (User Datagram Protocol):**
  - **Maximum UDP size**:
    - En principio, como IP tiene una longitud máxima de 65335 bytes (16-bits de longitud de datagrama), la máxima longitud de un datagrama UDP sería 65335 – 20 bytes = 65315 bytes, lo que hace una longitud de datos de 65315 – 8 bytes = 65307 bytes
    - Pero **limitaciones del software**: casi todas las APIs limitan la longitud de los datagramas UDP (lo mismo para TCP) a la máxima longitud que los buffers de lectura y escritura de los sockets (llamadas al sistema "read" y "write")
    - Este límite es dependiente del SO: e.g: muchos de ellos (e.g.; FreeBSD) usan 8192 bytes como tamaño máximo del datagrama UDP

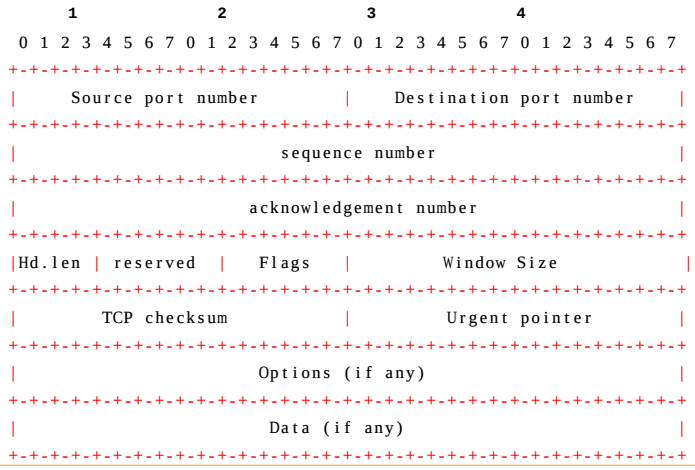
Tema 3: Transporte: TCP

- **TCP (Transmission Control Protocol):**
  - Protocolo orientado a la conexión (estableciemiento de la conexión, envío de datos y cierre de la conexión)
  - Tal vez el protocolo **MAS complejo** e **importante** de la pila de protocolos. Unidad de datos es el "segmento TCP"
  - Proporciona fiabilidad mediante el control de flujo, control de errores y control de la congestión
  - Los segmentos pueden llegar fuera de orden (debajo hay IP que es no orientado a la conexión, o sea, datagrama), por tanto, TCP debe reordenar los segmentos antes de pasarlos a la aplicación



Tema 3: Transporte: TCP

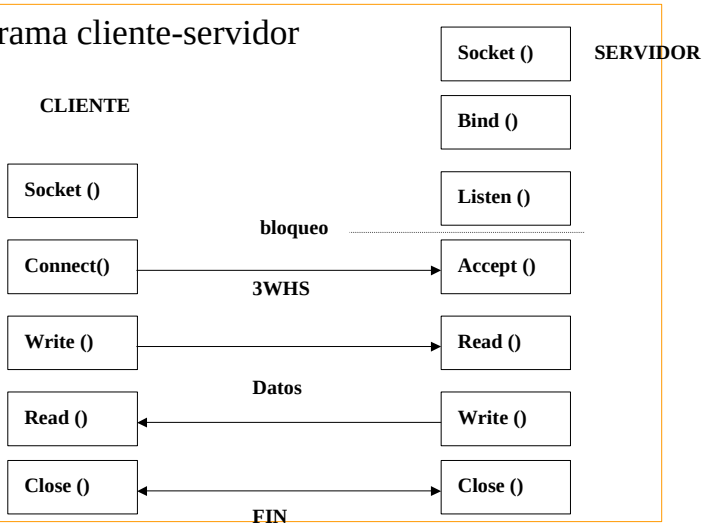
- **Cabecera TCP:**



- **Cabecera TCP:**
  - **Puertos:** identifican las aplicaciones
  - **Sequence number:** identifica el primer byte dentro de ese segmento de la secuencia de bytes enviados hasta ese momento.
    - ISN (Initial Seq. Number): primer número de secuencia escogida por el protocolo TCP
    - Seq. Number será un número a partir de ISN. Por tanto para saber cuantos bytes llevamos enviados hay que hacer *"Seq. Number – ISN"*
  - **Ack Number:** contiene el próximo número de seq. que el transmisor del ACK espera recibir
    - Por tanto es el "*Seq. Number+1"* del último byte recibido correctamente
    - Cuidado !!! TCP es FULL-DUPLEX: cada extremo mantiene un Seq. Number y un Ack Number
  - **Header length:** longitud total de la cabecera (opciones variable)

  - **Flags:** hay 6 flags (bits) en la cabecera
    - **URG:** Urgent Pointer field valido
    - **ACK:** Ack Number es valido
    - **PSH:** el receptor debe pasar los datos a la aplicación tan rápido como sea posible
    - **RST:** "Reset" la conexión
    - **SYN:** Sincronización de los números de secuencia al iniciar la conexión
    - **FIN:** termina la conexión
  - **Window Size**: tamaño de la ventana advertida por el receptor al transmisor (Sliding Window). Máxima ventana = 65535 bytes
  - **Checksum**: de todo el segmento TCP (igual que UDP)
  - **Urgent Pointer**: puntero al Seq. Number que indica la parte de datos urgentes dentro del campo de datos
  - **Options**: opción de anunciar el MSS (Maximum Segment Size)

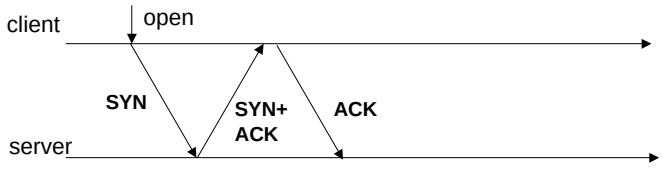
- Programa cliente-servidor



- Establecimiento de la conexión TCP:
  - Usa el 3-Way Handshake Algorithm:
    - El cliente envía un ***segmento SYN*** especificando el puerto destino del servidor, su puerto origen (escogido por el Kernel) y el ISN (Initial Seq Number) escogido al azar por el Kernel
    - El receptor (servidor) devuelve un ***segmento SYN+ACK*** reconociendo el segmento SYN e indicando su ISN
    - El cliente responde con un ***segmento ACK*** reconociendo el SYN+ACK
  - Una vez establecido la conexión se pasa a la fase de envio de datos
  - Es posible negociar **("indicar")** opciones (e.g.; indicar el MSS)
    - **MSS (Maximum Segment Size):** tamaño máximo de un segmento TCP.
      - Viene fijado por el Kernel: default = 536 bytes para un datagrama IP de 576 bytes (histórico X.25)
      - Sino, fijado por el MTU (Minimum Transfer Unit) menos cabeceras: e.g.; 1452 bytes en IEEE 802.3 (es 1500 – 8 LLC – 20 IP –20 TCP)

Tema 3: Transporte: TCP

- Establecimiento de la conexión TCP:
  - Usa el 3-Way Handshake Algorithm:



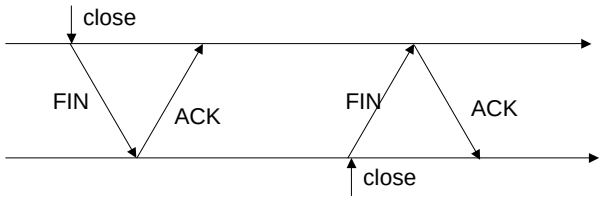

```
11:27:13.771041 147.83.35.18.3020 > 147.83.32.14.ftp: S
951111901:951111901(0) win 32120 <mss 1460,sackOK,timestamp
86683767 0,nop,wscale 0> (DF)
11:27:13.771491 147.83.32.14.ftp > 147.83.35.18.3020 : S
211543977:211543977(0) ack 951111902 win 10136 <nop,nop,timestamp
104199850 86683767,nop,wscale 0,nop,nop,sackOK,mss 1460> (DF)
11:27:13.771517 147.83.35.18.3020 > 147.83.32.14.ftp: . 1:1(0) ack
1 win 32120 <nop,nop,timestamp 86683767 104199850> (DF)
```

Tema 3: Transporte: TCP

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

Tema 3: Transporte: TCP

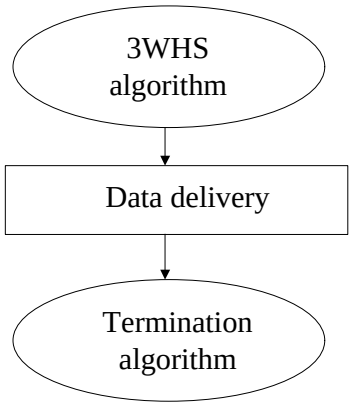
- Cierre de la conexión TCP




```
11:27:19.397349 147.83.32.14.ftp > 147.83.35.18.3020 : F
3658365:3658365(0) ack 1 win 10136 <nop,nop,timestamp 104200411
86684327> (DF)
11:27:19.397370 147.83.35.18.3020 > 147.83.32.14.ftp: . 1:1(0) ack
3658366 win 31856 <nop,nop,timestamp 86684330 104200411> (DF)
11:27:19.397453 147.83.35.18.3020 > 147.83.32.14.ftp: F 1:1(0) ack
3658366 win 31856 <nop,nop,timestamp 86684330 104200411> (DF)
11:27:19.398437 147.83.32.14.ftp > 147.83.35.18.3020 : .
3658366:3658366(0) ack 2 win 10136 <nop,nop,timestamp 104200412
86684330> (DF)
```

Tema 3: Transporte: TCP

- Grafo de estados en una conexión TCP
  - En total 11 estados distintos: CLOSE, ESTABLISHED, LISTEN, LAST_ACK, ...



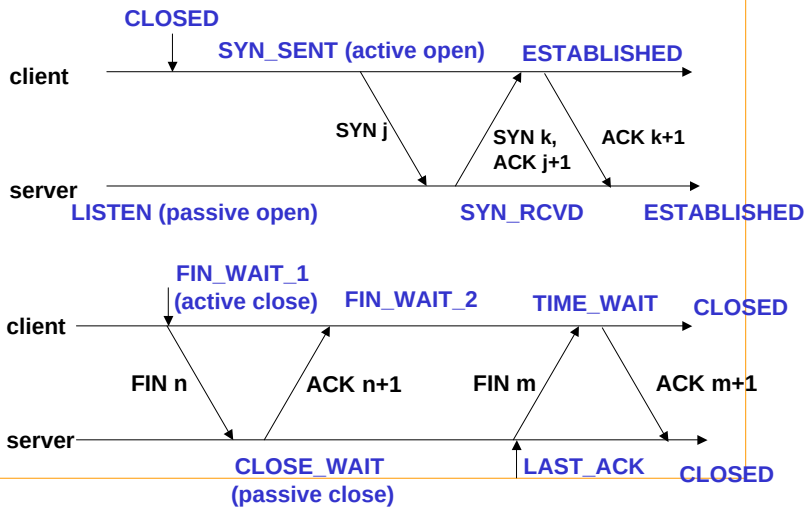
Tema 3: Transporte: TCP

- Estados TCP

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

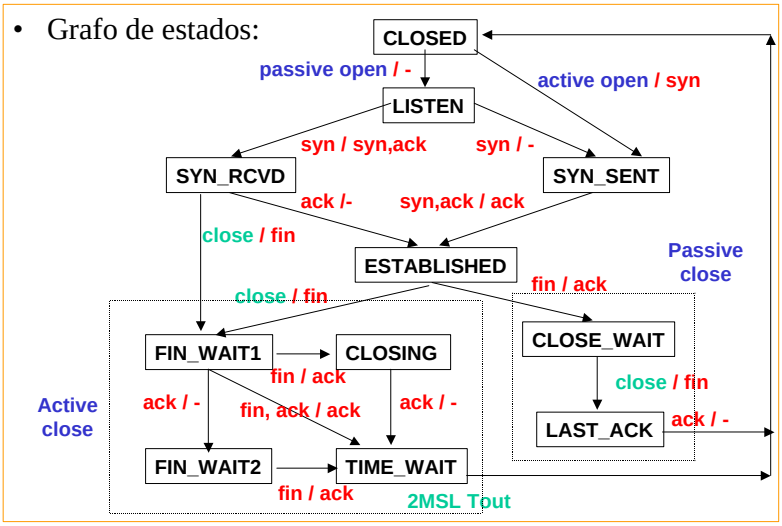
Tema 3: Transporte: TCP

- Grafo de estados:



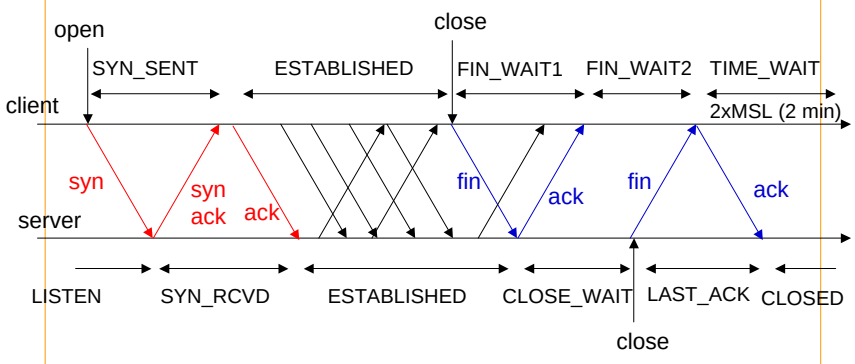
Tema 3: Transporte: TCP

- Grafo de estados:



Tema 3: Transporte: TCP

- Grafo de estados:



- Grafo de estados:
  - Comando **netstat**: permite listar las estructuras de datos de disversos protocolos de red (TCP, tablas de encaminamiento, tablas ARP, ...)

netstat [-g | -m | -p | -s | -r | -f address_family**] [**-n**] [**-P protocol **]**

g: multicast group

m: shows streams statistics

p: shows ARP table

s: shows per-protocol statistics

r: shows routing table

P: statitistics of all socket associated to protocol

Tema 3: Transporte: TCP

- Ejemplo netstat

```
rogent:~>netstat -fn inet

UDP
   Local Address         Remote Address     State
-------------------- -------------------- -------
127.0.0.1.32889      127.0.0.1.123          Connected


TCP
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
147.83.31.7.22       147.83.35.18.1020     8688   8687 10136      0 ESTABLISHED
147.83.31.7.40336    147.83.31.7.32775    32768      0 32768      0 TIME_WAIT
147.83.31.7.2003     147.83.33.6.52649    32768      0  8760      0 TIME_WAIT
147.83.31.7.40343    147.83.31.7.32775    32768      0 32768      0 TIME_WAIT
147.83.31.7.40359    147.83.31.7.32775    32768      0 32768      0 TIME_WAIT
147.83.31.7.512      147.83.35.110.1183    8687      0  8760      0 CLOSE_WAIT
147.83.31.7.39913    147.83.35.14.6000     7840      0  8760     32 ESTABLISHED
147.83.31.7.40361    147.83.35.106.19     17423      0  8760    512 ESTABLISHED
147.83.31.7.957      147.83.33.10.2049     8760   2483  8760      0 ESTABLISHED
```

bytes in Tx buffer

bytes in Rx buffer