TCP

Tema 3.- Nivel de transporte en Internet

Dr. Daniel Morató
Redes de Computadores
Ingeniero Técnico en Informática de Gestión, 2º curso

Material parcialmente adaptado del libro *Computer Networking: A Top Down Approach Featuring the Internet*, 3rd edition. Jim Kurose, Keith Ross, Ed. Addison-Wesley, Julio 2004

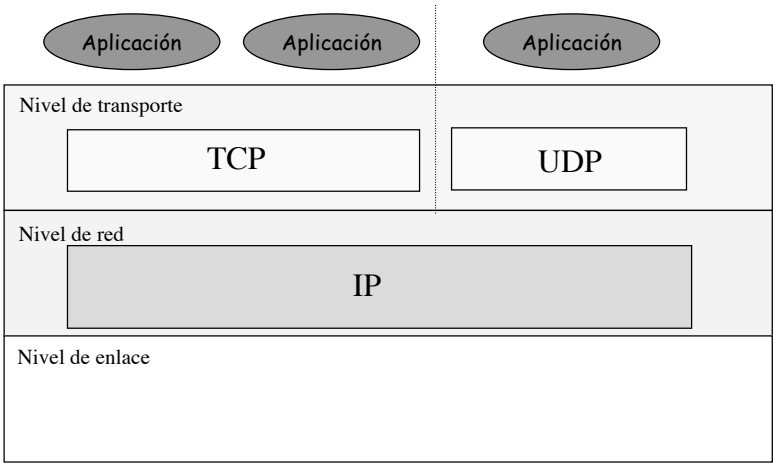
Temario

Temario

0.- Presentación de la asignatura
1.- Introducción
2.- Nivel de aplicación en Internet
3.- Nivel de transporte en Internet
- Principios
- Protocolo UDP
- Protocolo TCP

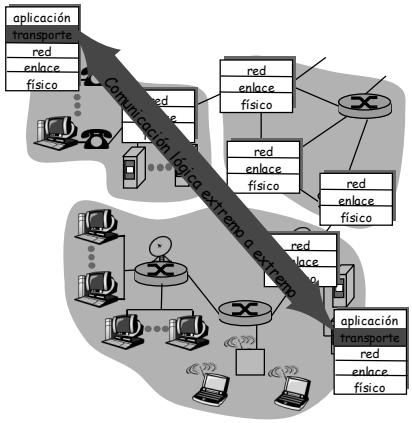
4.- Nivel de red en Internet
5.- Nivel de enlace

Protocolos de transporte



Protocolos y servicios de transporte

- Ofrece *comunicación lógica* entre procesos de aplicación corriendo en diferentes hosts
- Los protocolos de transporte funcionan en los end sistems
  - Emisor: separa los mensajes en segmentos, los pasa al nivel de red
  - Receptor: reensambla los segmentos en mensajes, los pasa al nivel de aplicación
- Más de un protocolo de transporte disponible para las aplicaciones
  - Internet: TCP y UDP



TCP

- RFCs: 793, 1122, 1323, 2018, 2581
- Punto a punto
- Orientado a conexión
- Flujo de datos:
  - Stream de bytes
  - Ordenados
  - Full-duplex
  - MSS: Maximum Segment Size
- Control de flujo
  - No saturar al receptor
- Control de congestión
  - No saturar la red

Contenido

- Multiplexación y demultiplexación en TCP
- Gestión de conexiones TCP
  - Establecimiento y liberación
- Transferencia fiable y control de flujo
  - Ventana deslizante
- Formato del segmento TCP
- Ejemplos

Contenido

- Multiplexación y demultiplexación en TCP
- Gestión de conexiones TCP
  - Establecimiento y liberación
- Transferencia fiable y control de flujo
  - Ventana deslizante
- Formato del segmento TCP
- Ejemplos

Cómo funciona la demux.

- Host recibe datagrama IP
  - Cada datagrama tiene una IP origen e IP destino
  - Cada datagrama lleva 1 segmento del nivel de transporte
  - Cada segmento tiene un puerto origen y uno destino
  - Nota: *well-known port numbers* para aplicaciones específicas (www.iana.org)
- El host podría emplear la dirección IP y el puerto destino para dirigir el segmento al socket apropiado

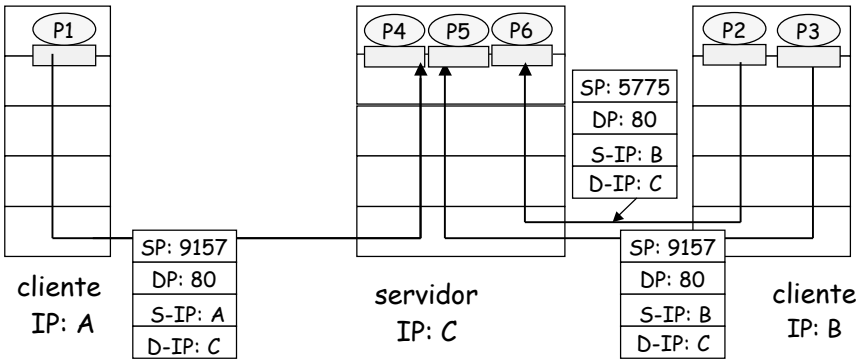
formato de mensaje TCP/UDP

Demultiplexación con conexión

- Conexión TCP identificada por 2 sockets
- Cada socket identificado por:
  - Dirección IP
  - Puerto TCP
- Es decir, la conexión viene identificada por:
  - Dirección IP (1)
  - Puerto TCP (1)
  - Dirección IP (2)
  - Puerto TCP (2)
- El receptor emplea la cuaterna para demultiplexar
- Cada host soporta múltiples conexiones TCP simultáneas
- Cada conexión identificada por esos 4 valores
- Con que uno sea diferente la conexión ya es diferente

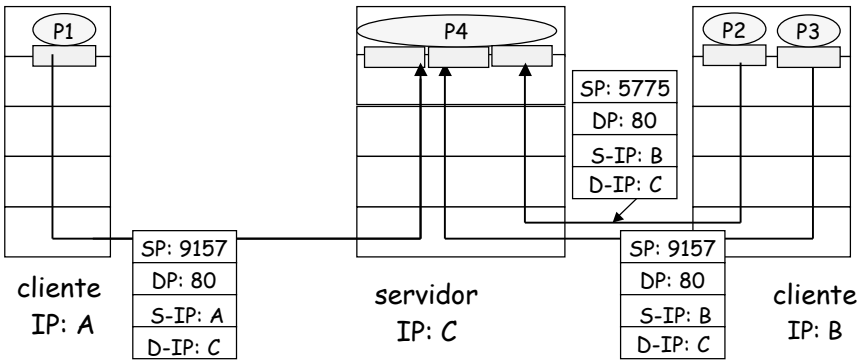
Demultiplexación con conexión

- Servidor web multiproceso



Demultiplexación con conexión

- Servidor web monoproceso



Contenido

- Multiplexación y demultiplexación en TCP
- Gestión de conexiones TCP
  - Establecimiento y liberación
- Transferencia fiable y control de flujo
  - Ventana deslizante
- Formato del segmento TCP
- Ejemplos

Gestión de conexiones TCP

Repaso:

- Ambos extremos establecen una “conexión” antes de intercambiar segmentos de datos
  - cliente: inicia la conexión

```
connect(sockcliente, (struct sockaddr*)&dirsock,
  sizeof(dirsock));
```

  - servidor: contactado por el cliente

```
sock=accept(sockservidor, (struct
  sockaddr*)&dirsock, &dirlen);
```

- Ambos extremos cierran la conexión

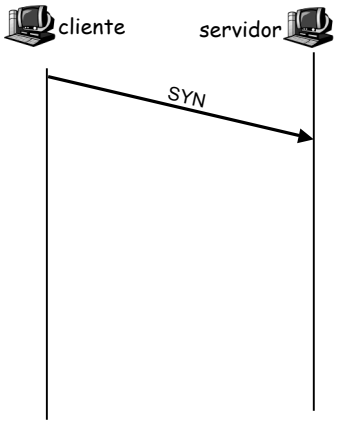
```
close(sockliente);
close(sock);
```

Gestión de conexiones

Estableciendo una conexión:
three way handshake

Paso 1:

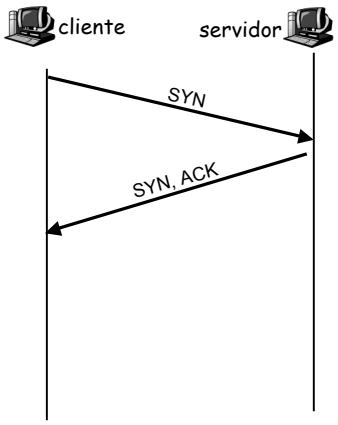
- El extremo cliente envía un segmento solicitando una conexión al servidor
- El segmento no tiene datos, solo cabecera
- *SYN*



Gestión de conexiones

Paso 2:

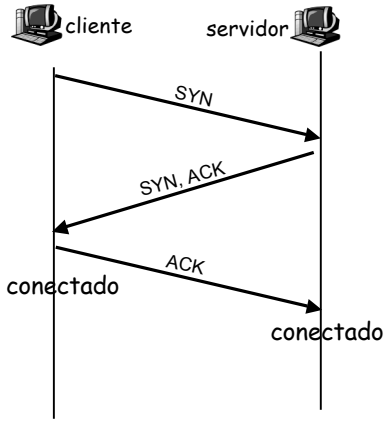
- El extremo servidor envía un segmento al cliente confirmando (*acknowledgement*) la recepción del SYN
- En el mismo segmento el servidor indica su deseo de establecer la conexión (SYN)
- El segmento no tiene datos, solo cabecera



Gestión de conexiones

Paso 3:

- El extremo cliente envía una confirmación al SYN del servidor
- El segmento no tiene datos, solo cabecera
- Conexión establecida



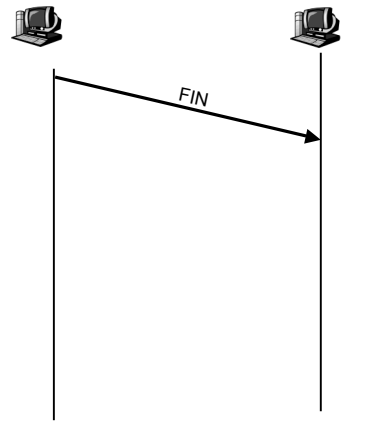
Transferencia de datos…

Gestión de conexiones

Cerrando una conexión

Paso 1:

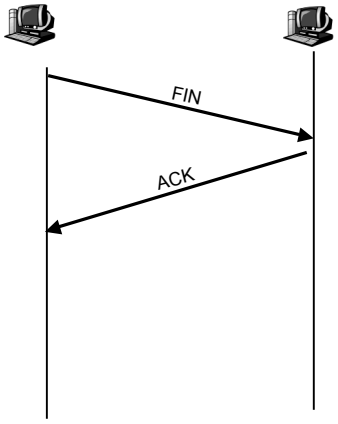
- Un extremo envía un segmento solicitando el cierre de la conexión
- El segmento no tiene datos, solo cabecera
- *FIN*



Gestión de conexiones

Paso 2:

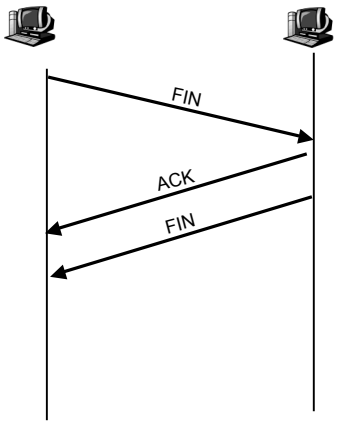
- El otro extremo confirma (ACK) la recepción del FIN
- El extremo que ha enviado el FIN ya no puede enviar más datos nuevos
- Cierre solo de un sentido de la comunicación



Gestión de conexiones

Paso 3:

- El otro extremo envía un segmento solicitando el cierre de la conexión
- El segmento no tiene datos, solo cabecera



Gestión de conexiones

Paso 4:

- Confirmación de ese segundo FIN
- Por si ese último ACK se pierde, el que lo envió espera un tiempo (podría tener que volverlo a enviar)
- Conexión cerrada

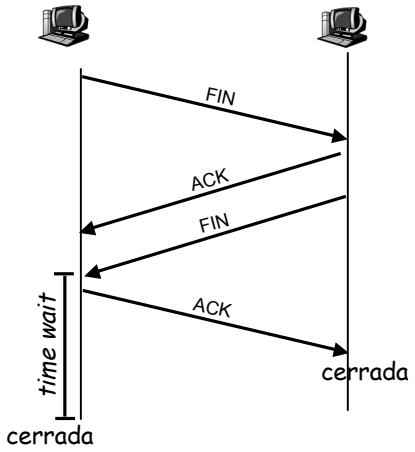
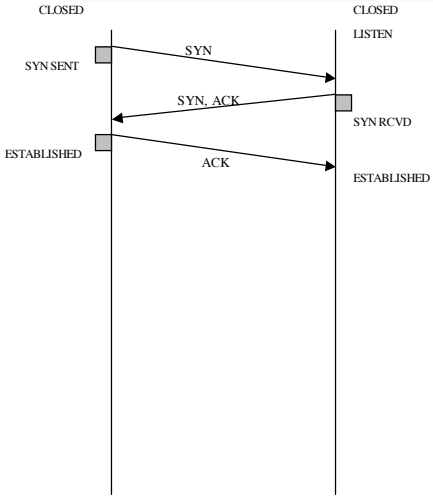


Diagrama de estados



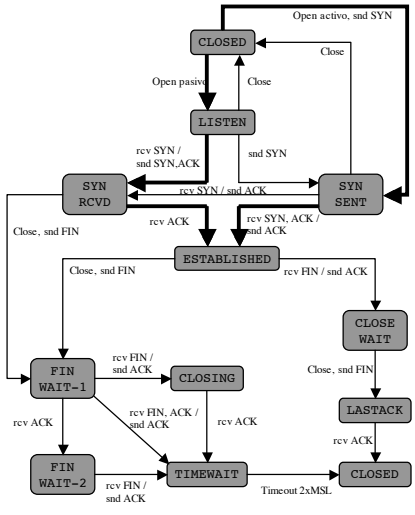
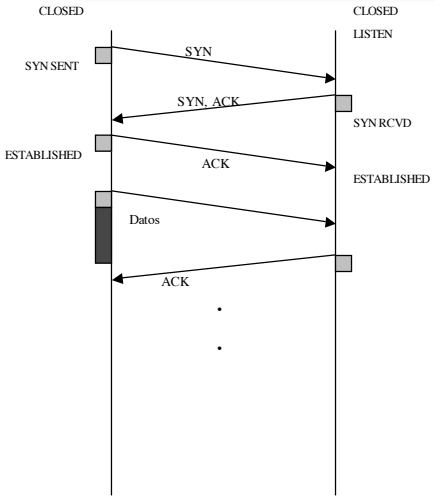


Diagrama de estados



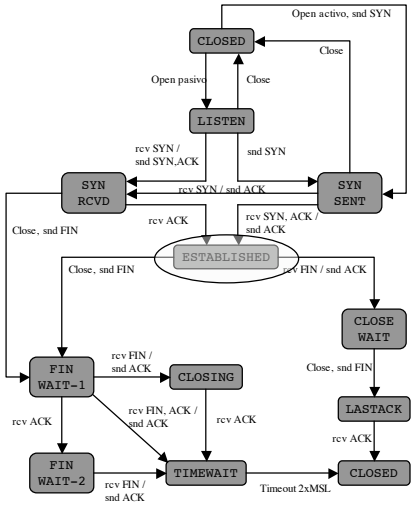
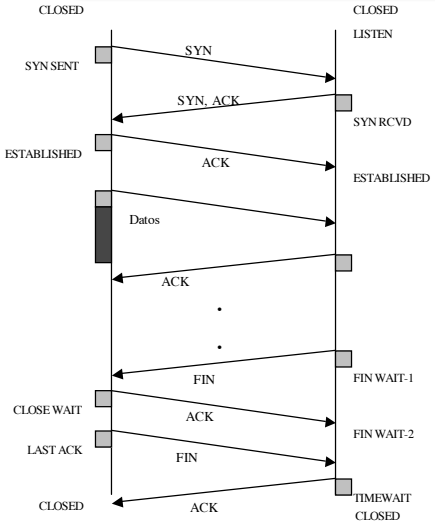
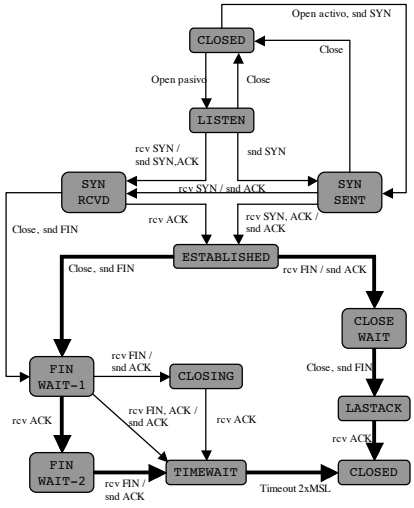
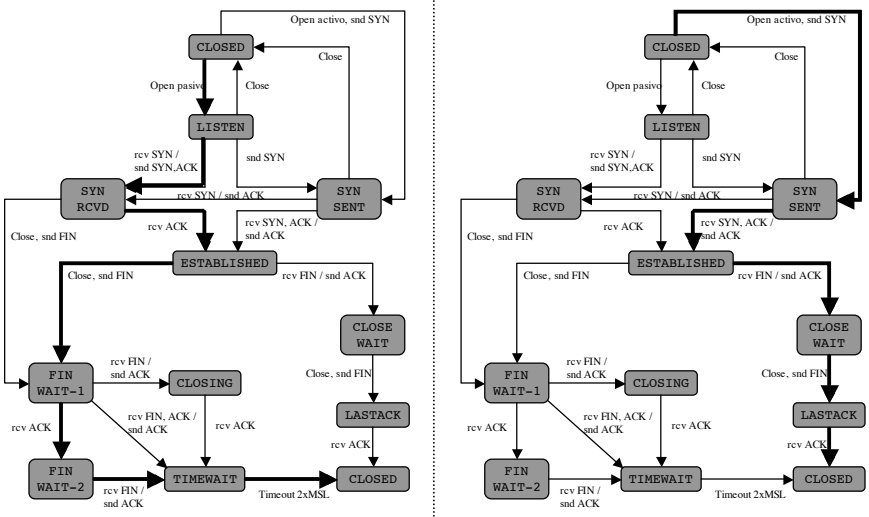


Diagrama de estados





Servidor Cliente



Ejemplo

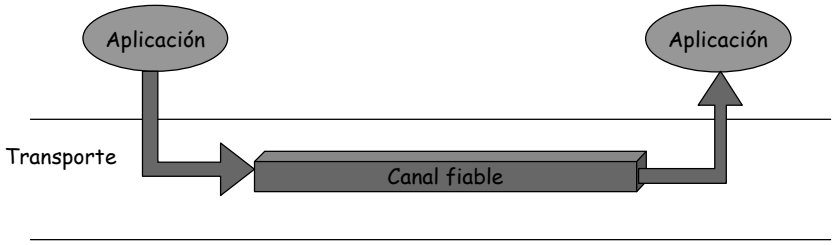
```
$ tcpdump -ttnlS tcp and host 10.1.11.1
Kernel filter, protocol ALL, datagram packet socket
tcpdump: listening on all devices
1005305154.171830 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: S 3462181145:3462181145(0)
1005305154.175780 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: S 1997882026:1997882026(0) ack 3462181146
1005305154.175929 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181146:3462181146(0) ack 1997882027

1005305154.177590 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181146:3462181173(27) ack 1997882027
1005305154.178398 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882027:1997882027(0) ack 3462181173
...

1005305166.816682 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: FP 1997882551:1997882559(8) ack 3462181333
1005305166.816794 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181333:3462181333(0) ack 1997882560
1005305166.817726 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: F 3462181333:3462181333(0) ack 1997882560
1005305166.818527 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882560:1997882560(0) ack 3462181334
```

Contenido

- Multiplexación y demultiplexación en TCP
- Gestión de conexiones TCP
  - Establecimiento y liberación
- Transferencia fiable y control de flujo
  - Ventana deslizante
- Formato del segmento TCP
- Ejemplos

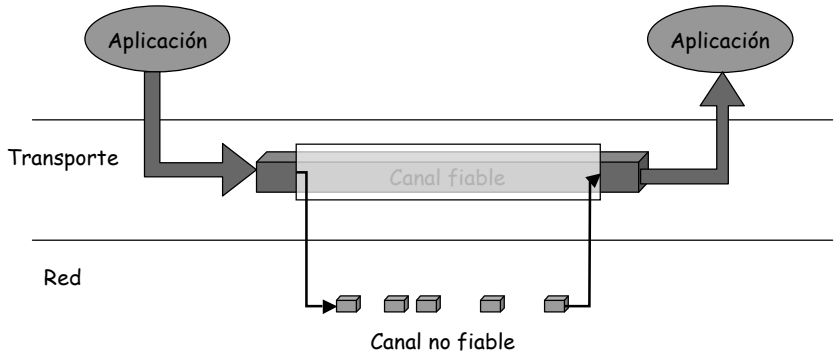
Transferencia fiable de datos

- Importante en nivel de aplicación, transporte, enlace



Transferencia fiable de datos

- Importante en nivel de aplicación, transporte, enlace



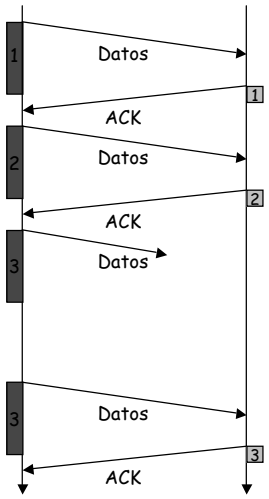
- Basado en nivel no fiable

Transferencia fiable

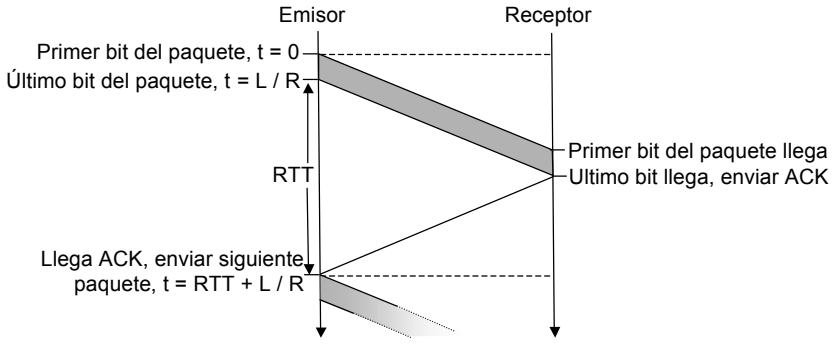
- ¿Cómo lograrla?
  - Emisor espera confirmación de la recepción del segmento
  - Si no recibe la confirmación en un tiempo *razonable* reenvía el segmento
  - Se numeran los segmentos o los bytes para confirmarlos
    - Una de las utilidades de los SYNs es establecer los números de secuencia iniciales para los datos

stop-and-wait

- Emisor, tras enviar espera la recepción de un ACK (confirmación)
- Si no la recibe tras un tiempo *razonable* retransmite el segmento



Prestaciones de *stop-and-wait*



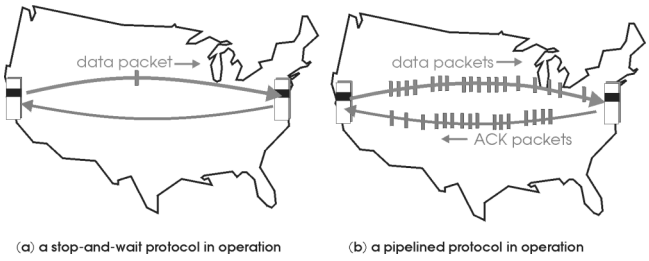
Ejemplo: 1 Gbps, 15 ms e-e (one way), 1KByte

- ¡33KBytes por segundo!
- ¡empleo el enlace el 0.027% del tiempo!

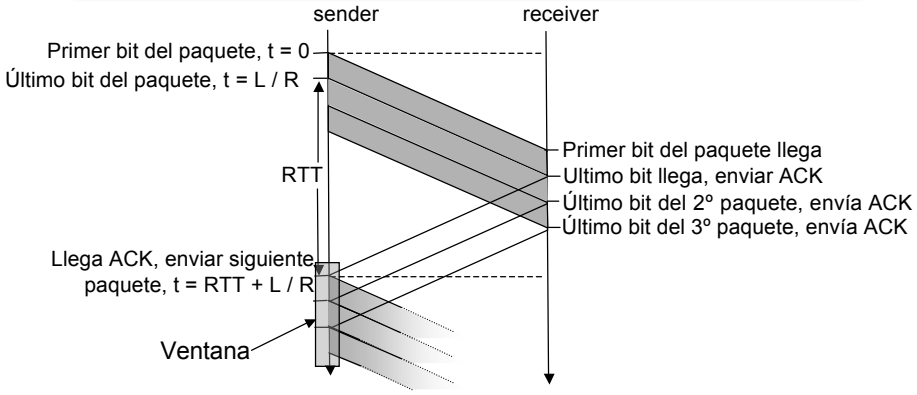
Pipelined protocols

Pipelining: emisor puede enviar varios segmento que estarán “en camino” sin aún haber sido confirmados

- buffering en el emisor y/o el receptor



Pipelining



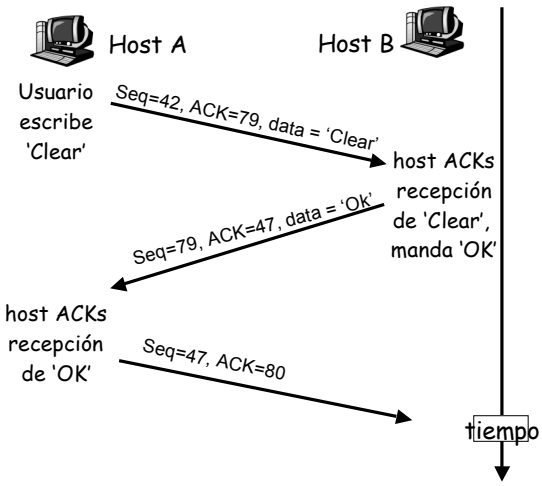
- Aumenta la *utilización* del canal

Números de sec. y ACK

Nº de sec.:
- Número en el stream del primer byte de datos en el segmento

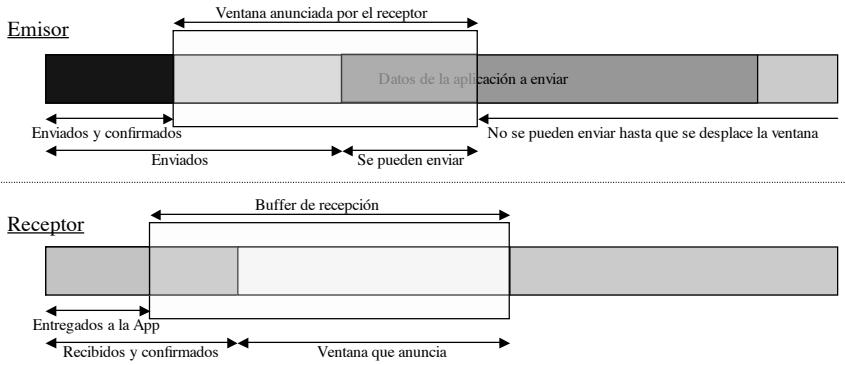
Nº de ACK:
- Número de secuencia del siguiente byte que se espera recibir
- ACK acumulado

SYN y FIN:
- Gastan 1 nº de secuencia
- Para poder ser confirmados



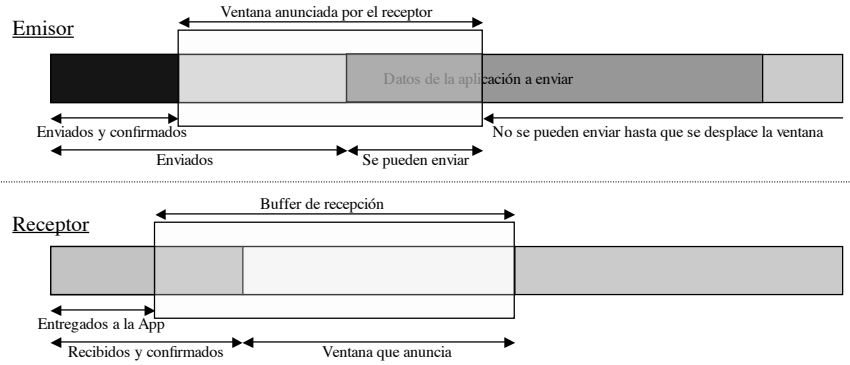
Ventana deslizante

- Full-duplex: Ambos extremos son emisor y receptor
- Por simplicidad analicemos solo un sentido



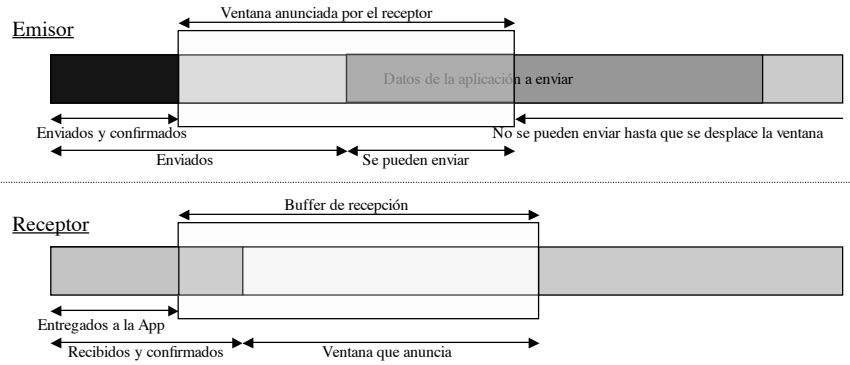
Ventana deslizante

- La aplicación receptor lee bytes del stream (la ventana se abre en el emisor, se desliza en el receptor)...



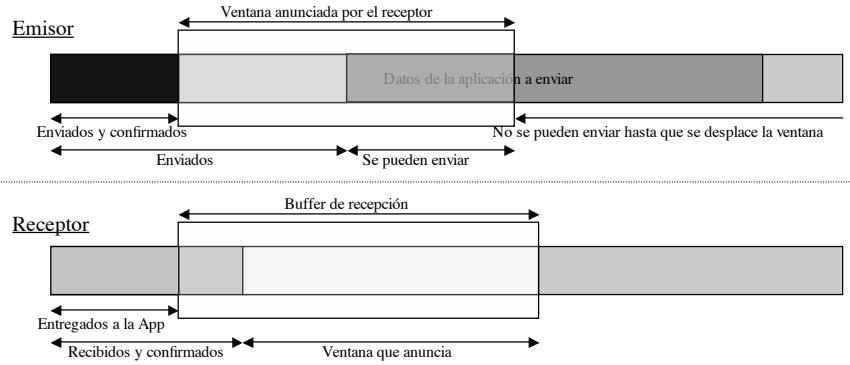
Ventana deslizante

- La aplicación receptor lee bytes del stream (la ventana se abre en el emisor, se desliza en el receptor)



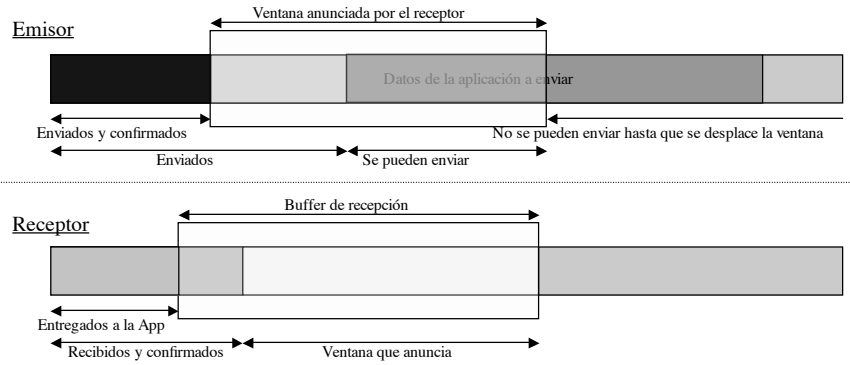
Ventana deslizante

- Se reciben más confirmaciones
- Se confirma siempre el último dato recibido consecutivo (sin huecos)
- La ventana se desliza en el emisor...



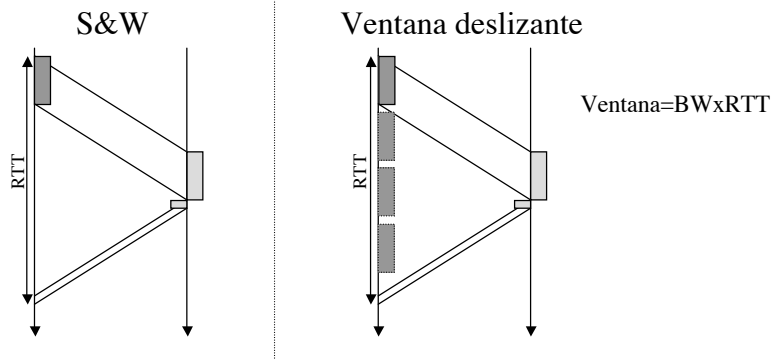
Ventana deslizante

- Se reciben más confirmaciones
- Se confirma siempre el último dato recibido consecutivo (sin huecos)
- La ventana se desliza en el emisor



Tamaño de la ventana

- Caso *Stop&Wait*, el tiempo de propagación y el de transmisión de la confirmación no se aprovecha
- Caso de ventana deslizante: ¿qué tamaño mínimo debe tener la ventana para aprovechar ese tiempo?...



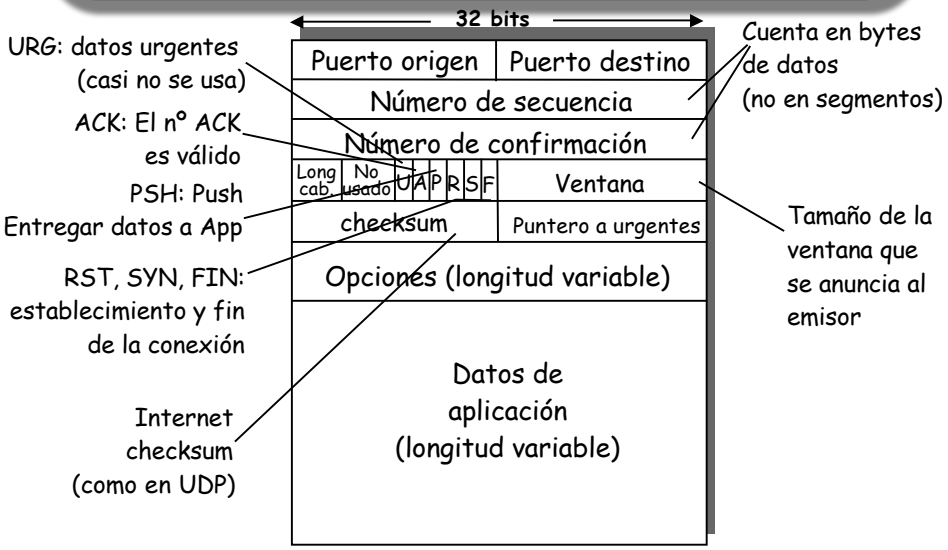
Producto RTTxBW

- Enlace E1 (2.048Kbps) a través de Europa (60ms)
  - Ventana > 15KBytes
  - Tamaño máximo de la ventana es 64KBytes
- Problemas:
  - Enlace de 10Mbps a través de Europa
    - Ventana > 74KBytes
    - Mayor que el máximo que permiten 16bits (!!)
  - Enlace E1 transoceánico (300ms)
    - Ventana > 76KBytes (!!)
  - Enlace Gigabit dentro de España (20ms)
    - Ventana > 2.5MBytes (!!!)
- Soluciones:
  - Aumentar el tamaño máximo de la ventana a 32bits (opción *window scale*)
  - Realizar varias conexiones simultáneamente (empleado por algunos sistemas peer-to-peer)

Contenido

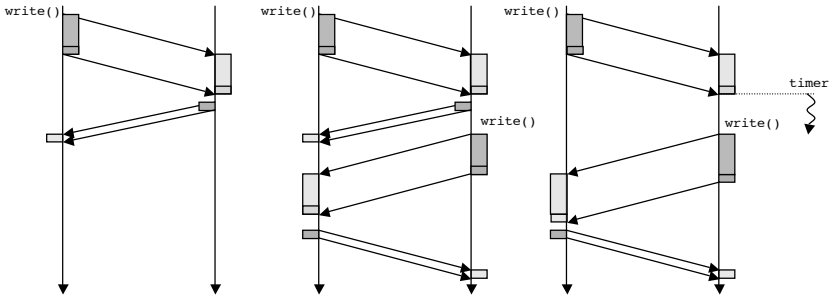
- Multiplexación y demultiplexación en TCP
- Gestión de conexiones TCP
  - Establecimiento y liberación
- Transferencia fiable y control de flujo
  - Ventana deslizante
- Formato del segmento TCP
- Ejemplos

Segmento TCP



Piggybacking

- Todos los segmentos incluyen la cabecera
- Los campos para confirmaciones viajan siempre en la cabecera
- Se puede aprovechar el flujo de datos en un sentido para enviar las confirmaciones del otro



Contenido

- Multiplexación y demultiplexación en TCP
- Gestión de conexiones TCP
  - Establecimiento y liberación
- Transferencia fiable y control de flujo:
  - Ventana deslizante
- Formato del segmento TCP
- Ejemplos

Ejemplo de ventana deslizante
(Receptor lento)

- `tcpdump` en el emisor

```
0               eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: S 0:0(0) win 32120 <mss 1460>
0.000211954     eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: S 0:0(0) ack 1 win 32120 <mss 1460>
0.000458002     eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 1 win 32120

0.00218892      eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 1:1449(1448) ack 1 win 32120
0.00224495      eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 1449:2897(1448) ack 1 win 32120

0.00646901      eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 1449 win 31856
0.00651395      eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 2897:4345(1448) ack 1 win 32120
0.00652695      eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 4345:5793(1448) ack 1 win 32120

0.00855601      eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 4345 win 31856
0.00858796      eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 5793:7241(1448) ack 1 win 32120
0.00859892      eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 7241:8689(1448) ack 1 win 32120
0.00860894      eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 8689:10137(1448) ack 1 win 32120

0.016923        eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 5793 win 30408
0.016958        eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 10137:11585(1448) ack 1 win 32120
0.0169699       eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 11585:13033(1448) ack 1 win 32120

0.0180379       eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 8689 win 28960
0.0180709       eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 13033:14481(1448) ack 1 win 32120
0.018082        eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 14481:15929(1448) ack 1 win 32120
0.0180919       eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 15929:17377(1448) ack 1 win 32120

...
```


Ejemplo (cont.)

```
...
0.159851        eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 52129 win 5792
0.159884        eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 56473:57921(1448) ack 1 win 32120

0.160865        eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 55025 win 4344
0.160898        eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 57921:59369(1448) ack 1 win 32120

0.166768        eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 57921 win 2896
0.166797        eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 59369:60817(1448) ack 1 win 32120

0.173574        eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 60817 win 1448
0.173606        eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 60817:62265(1448) ack 1 win 32120

0.191377        eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 62265 win 0

...

2.01157         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 62265 win 1448
2.0116          eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 62265:63713(1448) ack 1 win 32120

2.01395         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 63713 win 1448
2.01399         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 63713:65161(1448) ack 1 win 32120

2.21217         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 63713:65161(1448) ack 1 win 32120
2.21372         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 65161 win 0

...
```

Ejemplo (cont.)

...

```
3.02153          eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 65161 win 1448
3.02155         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 65161:66609(1448) ack 1 win 32120

3.0216          eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 65161 win 2896
3.02162         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 66609:68057(1448) ack 1 win 32120


3.0217          eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 65161 win 5792
3.02616         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 65161 win 11584
3.02623         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 65161 win 23168
3.02835         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 68057 win 31856


3.02838         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 68057:69505(1448) ack 1 win 32120
3.02839         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 69505:70953(1448) ack 1 win 32120
3.02841         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 70953:72401(1448) ack 1 win 32120
3.03392         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 70953 win 31856
3.03395         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: . 72401:73849(1448) ack 1 win 32120
3.03396         eth0 > 1.1.1.13.1510 > 1.1.1.12.2704: P 73849:75297(1448) ack 1 win 32120
3.03761         eth0 < 1.1.1.12.2704 > 1.1.1.13.1510: . 1:1(0) ack 73849 win 31856
```


Ejemplo de HTTP

```
17.797014 IP localhost.53434 > localhost.http: S 0:0(0) win 65535
17.797158 IP localhost.http > localhost.53434: S 0:0(0) ack 1 win 65535
17.797188 IP localhost.53434 > localhost.http: . ack 1 win 65535
19.417759 IP localhost.53434 > localhost.http: P 1:8(7) ack 1 win 65535
19.502435 IP localhost.http > localhost.53434: . ack 8 win 65535
19.635105 IP localhost.http > localhost.53434: P 1:1457(1456) ack 8 win 65535
19.644994 IP localhost.http > localhost.53434: F 1457:1457(0) ack 8 win 65535
19.645084 IP localhost.53434 > localhost.http: . ack 1458 win 65535
19.645459 IP localhost.53434 > localhost.http: F 8:8(0) ack 1458 win 65535
19.645524 IP localhost.http > localhost.53434: . ack 9 win 65535
```

Ejemplo de *telnet*

```
$ telnet 10.1.11.1
Trying 10.1.11.1...
Connected to 10.1.11.1.
Escape character is '^]'.

Red Hat Linux release 6.0 (Hedwig)
Kernel 2.2.5-15 on an i586
login: ro
Password:
Last login: Fri Nov  9 09:30:27 from lucas.net.tlm.unavarra.es
[ro@pc1r11 ro]$ ls -al
total 3
drwxr-xr-x   2 ro       users        1024 Oct 31 20:10 .
drwxr-xr-x   5 root     root         1024 Sep 25 19:25 ..
-rw-------   1 ro       users         482 Nov  9 09:30 .bash_history
[ro@pc1r11 ro]$ date
Fri Nov  9 09:50:57 CET 2001
[ro@pc1r11 ro]$ ls
[ro@pc1r11 ro]$ exit
logout
Connection closed by foreign host.
```


Ejemplo de *telnet*
(Establecimiento + login)

```
$ /opt3/ro/ficheros/bin/tcpdump_ro -ttnlS tcp and host 10.1.11.1
Kernel filter, protocol ALL, datagram packet socket
tcpdump: listening on all devices
1005305154.171830 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: S 3462181145:3462181145(0)
1005305154.175780 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: S 1997882026:1997882026(0) ack 3462181146
1005305154.175929 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181146:3462181146(0) ack 1997882027

1005305154.177590 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181146:3462181173(27) ack 1997882027
1005305154.178398 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882027:1997882027(0) ack 3462181173
1005305154.215773 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882027:1997882039(12) ack 3462181173
1005305154.215882 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181173:3462181173(0) ack 1997882039
1005305154.216635 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882039:1997882078(39) ack 3462181173
1005305154.218233 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181173:3462181291(118) ack 1997882078
1005305154.222194 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882078:1997882081(3) ack 3462181291
1005305154.222356 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181291:3462181294(3) ack 1997882081
1005305154.241460 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882081:1997882081(0) ack 3462181294
1005305154.242896 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882081:1997882150(69) ack 3462181294
1005305154.243574 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181294:3462181297(3) ack 1997882150
1005305154.261456 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882150:1997882150(0) ack 3462181297

1005305154.275262 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882150:1997882157(7) ack 3462181297
1005305154.292590 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181297:3462181297(0) ack 1997882157
1005305155.980047 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181297:3462181298(1) ack 1997882157
1005305155.980947 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882157:1997882158(1) ack 3462181298
1005305155.992578 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181298:3462181298(0) ack 1997882158
1005305156.111700 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181298:3462181299(1) ack 1997882158
1005305156.112556 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882158:1997882159(1) ack 3462181299
1005305156.132580 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181299:3462181299(0) ack 1997882159
1005305156.279616 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181299:3462181301(2) ack 1997882159
1005305156.280493 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882159:1997882161(2) ack 3462181301
1005305156.292581 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181301:3462181301(0) ack 1997882161
```

Negociación de opciones

login (ro)

Ejemplo de *telnet*
(password)

```
005305156.298750 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882161:1997882171(10) ack 3462181301
1005305156.312576 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181301:3462181301(0) ack 1997882171
1005305156.847827 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181301:3462181302(1) ack 1997882171
1005305156.861613 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882171:1997882171(0) ack 3462181302
1005305156.991599 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181302:3462181303(1) ack 1997882171
1005305157.011618 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882171:1997882171(0) ack 3462181303
1005305157.167585 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181303:3462181304(1) ack 1997882171
1005305157.181630 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882171:1997882171(0) ack 3462181304
1005305157.303549 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181304:3462181305(1) ack 1997882171
1005305157.321640 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882171:1997882171(0) ack 3462181305
1005305157.483585 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181305:3462181306(1) ack 1997882171
1005305157.501653 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882171:1997882171(0) ack 3462181306
1005305157.643631 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181306:3462181307(1) ack 1997882171
1005305157.661667 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882171:1997882171(0) ack 3462181307
1005305157.823141 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181307:3462181309(2) ack 1997882171
1005305157.847153 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882171:1997882171(0) ack 3462181309
1005305157.871390 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882171:1997882173(2) ack 3462181309
1005305157.882568 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181309:3462181309(0) ack 1997882173

1005305157.883451 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882173:1997882237(64) ack 3462181309
1005305157.902564 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181309:3462181309(0) ack 1997882237
1005305158.011039 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882237:1997882253(16) ack 3462181309
1005305158.022565 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181309:3462181309(0) ack 1997882253
```

password (roupna)

prompt

Ejemplo de *telnet*
(ls -al)

```
1005305158.907613 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181309:3462181310(1) ack 1997882253
1005305158.908789 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882253:1997882254(1) ack 3462181310
1005305158.922561 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181310:3462181310(0) ack 1997882254
1005305159.007422 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181310:3462181311(1) ack 1997882254
1005305159.008554 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882254:1997882255(1) ack 3462181311
1005305159.022561 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181311:3462181311(0) ack 1997882255
1005305159.119386 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181311:3462181312(1) ack 1997882255
1005305159.120520 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882255:1997882256(1) ack 3462181312
1005305159.132558 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181312:3462181312(0) ack 1997882256
1005305159.327436 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181312:3462181313(1) ack 1997882256
1005305159.328572 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882256:1997882257(1) ack 3462181313
1005305159.342558 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181313:3462181313(0) ack 1997882257
1005305159.707464 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181313:3462181314(1) ack 1997882257
1005305159.708600 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882257:1997882258(1) ack 3462181314
1005305159.722555 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181314:3462181314(0) ack 1997882258
1005305159.775309 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181314:3462181315(1) ack 1997882258
1005305159.776442 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882258:1997882259(1) ack 3462181315
1005305159.792557 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181315:3462181315(0) ack 1997882259
1005305160.119438 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181315:3462181317(2) ack 1997882259
1005305160.120588 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882259:1997882261(2) ack 3462181317
1005305160.132552 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181317:3462181317(0) ack 1997882261

1005305160.133281 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882261:1997882270(9) ack 3462181317
1005305160.152551 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181317:3462181317(0) ack 1997882270
1005305160.153862 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882270:1997882473(203) ack 3462181317
1005305160.172553 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181317:3462181317(0) ack 1997882473
```

ls -al

Ejemplo de *telnet*
(date, ls)

```
1005305162.031585 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181317:3462181318(1) ack 1997882473
1005305162.032830 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882473:1997882474(1) ack 3462181318
1005305162.052543 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181318:3462181318(0) ack 1997882474
1005305162.128271 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181318:3462181319(1) ack 1997882474
1005305162.129759 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882474:1997882475(1) ack 3462181319
1005305162.142847 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181319:3462181319(0) ack 1997882475
1005305162.355287 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181319:3462181320(1) ack 1997882475
1005305162.356477 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882475:1997882476(1) ack 3462181320
1005305162.372538 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181320:3462181320(0) ack 1997882476
1005305162.423120 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181320:3462181321(1) ack 1997882476
1005305162.424266 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882476:1997882477(1) ack 3462181321
1005305162.442538 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181321:3462181321(0) ack 1997882477
1005305162.611164 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181321:3462181323(2) ack 1997882477
1005305162.612321 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882477:1997882479(2) ack 3462181323
1005305162.622537 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181323:3462181323(0) ack 1997882479
1005305162.623313 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882479:1997882509(30) ack 3462181323
1005305162.642536 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181323:3462181323(0) ack 1997882509

1005305162.643257 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882509:1997882525(16) ack 3462181323
1005305162.662537 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181323:3462181323(0) ack 1997882525
1005305165.247354 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181323:3462181324(1) ack 1997882525
1005305165.248601 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882525:1997882526(1) ack 3462181324
1005305165.262519 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181324:3462181324(0) ack 1997882526
1005305165.306850 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181324:3462181325(1) ack 1997882526
1005305165.307981 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882526:1997882527(1) ack 3462181325
1005305165.322521 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181325:3462181325(0) ack 1997882527
1005305165.406785 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181325:3462181327(2) ack 1997882527
1005305165.407919 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882527:1997882529(2) ack 3462181327
1005305165.422520 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181327:3462181327(0) ack 1997882529
1005305165.423257 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882529:1997882545(16) ack 3462181327
1005305165.442517 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181327:3462181327(0) ack 1997882545
```

date

ls

Ejemplo de *telnet*
(exit + cierre)

```
1005305165.998183 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181327:3462181328(1) ack 1997882545
1005305165.999316 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882545:1997882546(1) ack 3462181328
1005305166.012516 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181328:3462181328(0) ack 1997882546
1005305166.254940 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181328:3462181329(1) ack 1997882546
1005305166.256135 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882546:1997882547(1) ack 3462181329
1005305166.272514 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181329:3462181329(0) ack 1997882547
1005305166.351498 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181329:3462181330(1) ack 1997882547
1005305166.352510 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882547:1997882548(1) ack 3462181330
1005305166.372516 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181330:3462181330(0) ack 1997882548
1005305166.490834 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181330:3462181331(1) ack 1997882548
1005305166.491998 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882548:1997882549(1) ack 3462181331
1005305166.502510 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181331:3462181331(0) ack 1997882549
1005305166.807062 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: P 3462181331:3462181333(2) ack 1997882549
1005305166.808036 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: P 1997882549:1997882551(2) ack 3462181333

1005305166.816682 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: FP 1997882551:1997882559(8) ack 3462181333
1005305166.816794 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: . 3462181333:3462181333(0) ack 1997882560
1005305166.817726 eth0 P 1.1.1.12.1798 > 10.1.11.1.telnet: F 3462181333:3462181333(0) ack 1997882560
1005305166.818527 eth0 P 10.1.11.1.telnet > 1.1.1.12.1798: . 1997882560:1997882560(0) ack 3462181334
```

exit

Piggybacking del FIN

Resumen

- Principios detrás de los servicios del nivel de transporte:
  - multiplexación, demultiplexación
  - transferencia fiable de datos
  - control de flujo
  - *control de* congestión
- Implementaciones en Internet
  - UDP
  - TCP

Temario

0.- Presentación de la asignatura
1.- Introducción
2.- Nivel de aplicación en Internet
3.- Nivel de transporte en Internet
- Principios
- Protocolo UDP
- Protocolo TCP

4.- Nivel de red en Internet
5.- Nivel de enlace

Próxima clase

Nivel de red: funciones
Enrutamiento

- Lecturas recomendadas:
  - [1] 4.1, 4.5, 4.6, 1.5