

# Protocolos de Transporte (Intro)

Redes y Comunicaciones

# Introducción

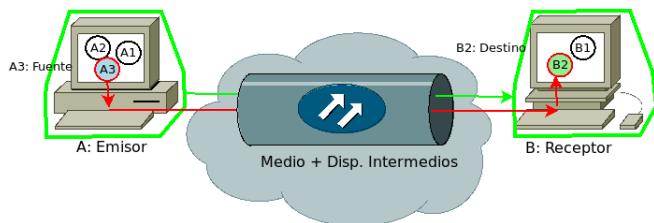
- Transporte de Internet, TCP/IP, usa IP como servicio.
- IP provee un servicio “débil”, pero eficiente: “Best-effort”.
- En IP los paquetes/datagramas pueden ser descartados, des-ordenados, retardados duplicados o corrompidos.
- Paquetes IP solo dirección DST y SRC, ¿ Cómo elegir la aplic. ?
- IP corre en “**todos**” **los nodos de la red (internet)** (routers y host), protocolos de transporte solo necesario en **end-points** (hosts).
- IP: comunicación lógica **HOP-BY-HOP**, comunica **hosts**.
- Transporte: comunicación lógica **HOST-TO-HOST (END-TO-END)**, comunica **procesos**.
- Aplicación: comunicación lógica **PROCESS-TO-PROCESS (END-TO-END)** comunica usuarios, agentes, etc.

# Stack del modelo TCP/IP, 4 capas



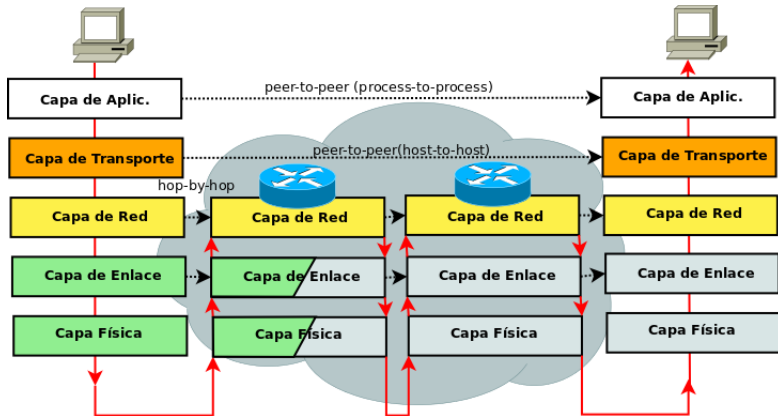
- Capa de transporte brinda servicio a capa de aplicación y usa servicios de capa de internet, (red IP).

# Direccionamiento a nivel de Transporte



- Red (IP) direcciona hosts, transporta al mensaje del transporte hasta el host (dir IP).
- Transporte (TCP, UDP, u otro) transporta al mensaje de aplicación hasta el proceso (puerto).

# Comunicación en Capas

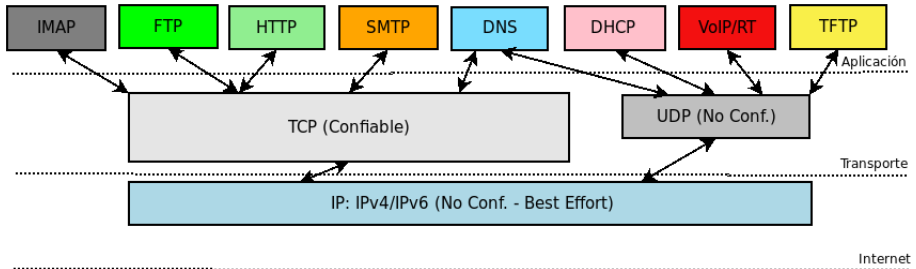


- Aplicación “habla” de proceso a proceso, servicio provisto por la capa de transporte.
- Transporte “habla” de host a host, donde los paquetes (segmentos) son llevados por la red.

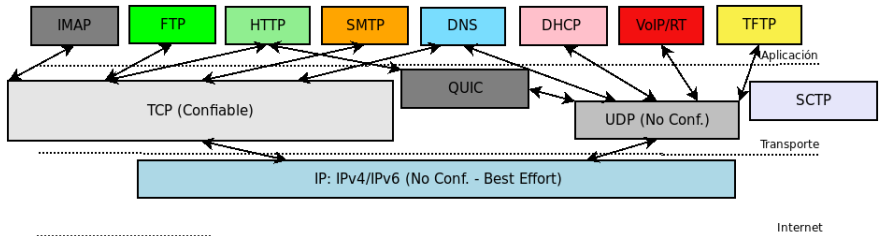
# Funcionalidad de la Capa de Transporte

- Encapsulación, define PDU donde se envía los mensajes de la aplicación.
- MUX/DEMUX process-to-process (puertos, Ports).
- Soporte de datos de tamaños arbitrarios.
- Control y Detección de Errores, pérdida, duplicación, se corrompen.
- ¿Cómo enviar info sobre la red de acuerdo al estado de la misma?  
¿Cuándo y Cómo una aplic. debe enviar datos?
  - Control de Flujo.
  - Control de Congestión.
- Dos modelos básicos:
  - Modelo Confiable: TCP.
  - Modelo NO Confiable: UDP.

# Protocolos de Transporte

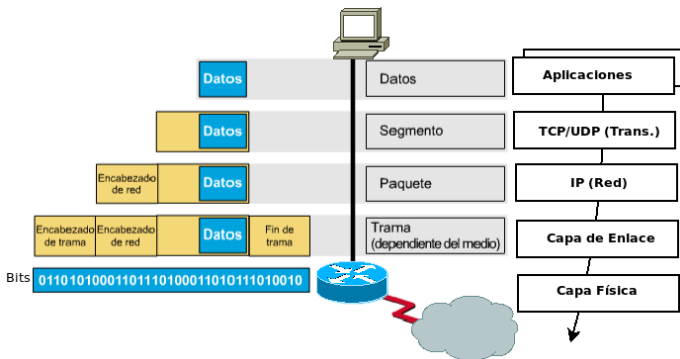


# Protocolos de Transporte, Alt.



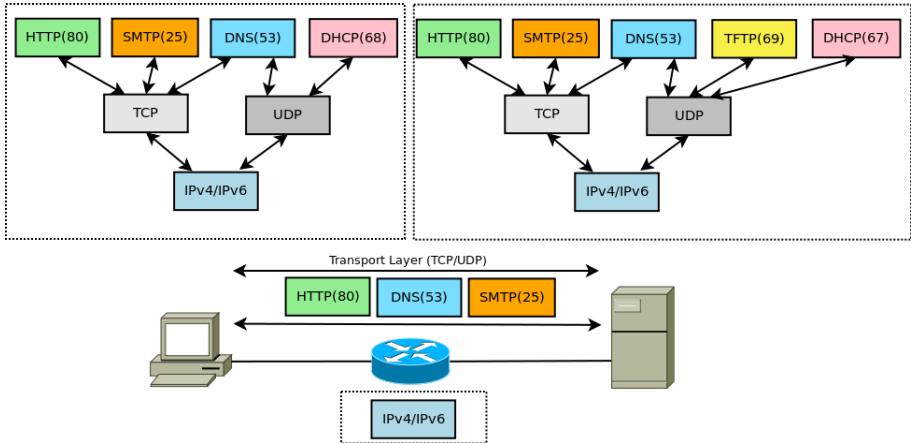


# Protocolos de Transporte, Encapsulación

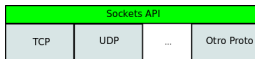
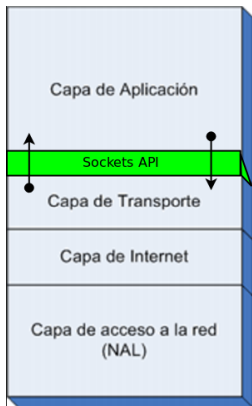


- PDU: Protocol Data Unit.
- IP (Red), capa de Internet en el modelo TCP/IP, PDU: paquete o datagrama.
- Transporte, PDU: de forma general segmento, UDP: datagrama.

# Multiplexación del Transporte



# Selección del Protocolo de Transporte



- La aplicación de acuerdo a como esta programada selecciona el transporte.
- El acceso a los servicios de transporte se hace mediante API: **Network socket**.

# UDP

- User Datagram Protocol (RFC-768).
- Protocolo Minimalista. Menor Overhead.
- Características de IP: best-effort.
- Orientado a Packets/Datagramas (mensajes auto-contenidos).
- PDU: Datagrama (Por coherencia con nivel Transporte se suele llamar Segmento).
- Solo provee MUX/DEMUX y detección de algunos errores(UDP4, puede desact.).
- No incrementa Overhead end-to-end.
- No requiere establecimiento de conexión.
- Servicio FDX.
- Aplicaciones: video/voz streaming/TFTP/DNS/Bcast/Mcast, transporte de transporte (QUIC).

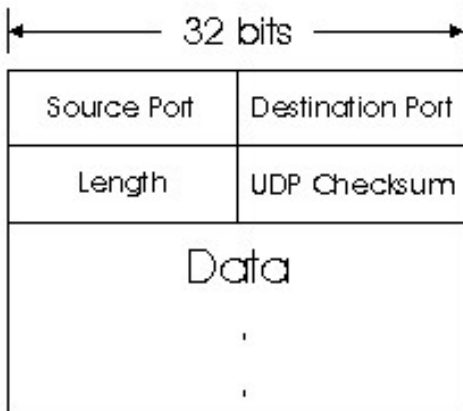
# Headers/Encabezados y Servicios de UDP

- El encabezado IP provee: Ruteo, Fragmentación, Detección de algunos errores.
- El encabezado UDP provee: MUX/DEMUX de aplic. , Detección de errores (no obligatorio para UDP4, UDP sobre IPv4).
- IP indica que lleva UDP con el código de protocolo 17.

```
? grep udp /etc/protocols
```

```
udp      17      UDP # user datagram protocol
```

# Datagrama UDP



# Campos del Datagrama UDP

- Puertos: MUX/DEMUX.
- Longitud: UDP HDR + Payload.
- Checksum
  - Cálculo Ca1, Opcional. 0 = Sin checksum.
  - Calculado HDR + PseudoHDR + Payload.
  - PseudoHDR:  $IP.SRC + IP.DST + Zero + IP.PROTO + UDP.LENGTH$ .
  - PseudoHDR: protección contra paquetes mal enrutados.
  - Aplicaciones de LAN por eficiencia lo podrían deshabilitar.
  - Si tiene error se descarta silenciosamente.

# Cálculo de Checksum

udp-2.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

udp

No.	Time	Source	Destination	Protocol	Length	Type	Info
15	62.110473	10.0.2.10	10.0.4.10	ECHO	47		Request
16	62.262709	10.0.4.10	10.0.2.10	ECHO	47		Response

▶ Frame 15: 47 bytes on wire (376 bits), 47 bytes captured (376 bits)  
 ▶ Ethernet II, Src: 00:00:00\_aa:00:00 (00:00:00:aa:00:00), Dst: 00:00:00\_aa:00:01 (00:00:00:aa:00:01)  
 ▶ Internet Protocol Version 4, Src: 10.0.2.10, Dst: 10.0.4.10  
 ▼ User Datagram Protocol, Src Port: 9000, Dst Port: 7  
     Source Port: 9000  
     Destination Port: 7  
     Length: 13  
     ▶ Checksum: 0x10f8 [correct]  
         [Checksum Status: Good]  
         [Stream index: 2]  
 ▶ Echo

```

0000  00 00 00 aa 00 01 00 00 00 aa 00 00 08 00 45 00  .....E.
0010  00 21 e0 61 40 00 3f 11 41 57 0a 00 02 0a 0a 00  -!.a@-?.Aw.....
0020  04 0a 23 28 00 07 00 0d 10 f8 54 45 53 54 0a    -..[.....TEST.
  
```

Details at: [http://www.wireshark.org/docs/wsug\\_html\\_chunked/ChAdvChecksums.html](http://www.wireshark.org/docs/wsug_html_chunked/ChAdvChecksums.html) (udp.checksum), 2 bytes    Packets: 46 · Displayed: 26 (56.5%)    Profile: Default



# Cálculo de Checksum (Cont.)

Pseudo header SRC=10.0.2.10, DST=10.0.4.10, PROTO=17  
 0A00 020A 0A00 040A 0011

UDP header SRCP=9000, DSTP=7, LEN=13, LEN-PH=13, CKSUM=0  
 2328 0007 000D 000D

DATA 5445 5354 0A00

```

00001010 00000000 = 10.0
00000010 00001010 = 2.10
00001010 00000000 = 10.0
00000100 00001010 = 4.10
00000000 00010001 = 17
00100011 00101000 = 9000
00000000 00000111 = 7
00000000 00001101 = 13
00000000 00001101 = 13
00101010 01000101
00101001 01010100
00001010 00000000
-----

```

```

~11101111 00000111    EF07 ~EF07 = 10F8
00010000 11111000

```

# Cálculo de Checksum (ejemplo con Carry(C))

Pseudo header SRC=10.0.2.10, DST=10.0.4.10, PROTO=17

0A00 020A 0A00 040A 0011

UDP header SRCP=9000, DSTP=7, LEN=15, LEN-PH=15, CKSUM=0

2328 0007 000F 000F

DATA 5445 5354 0A00 FF[00] [00] = v. padding

00001010 00000000 = 10.0

00000010 00001010 = 2.10

00001010 00000000 = 10.0

00000100 00001010 = 4.10

00000000 00010001 = 17

00100011 00101000 = 9000

00000000 00000111 = 7

00000000 00001111 = 15

00000000 00001111 = 15

01010100 01000101

01010011 01010100

00001010 00000000

11111111 00000000

-----  
{1}11101110 00010111  
                  {1}

-----  
~11101110 00011000

00010001 11100111

EE18 ^EE18 = 11E7

# TCP

- Transport Control Protocol (RFC-793).
- Protocolo confiable, ordenado, con buffering, control de errores, flujo y de congestión.
- Orientado a Streams (secuencia de bytes,  $\equiv$  archivo).
- PDU: Segmento, una porción del stream de bytes.
- Provee MUX/DEMUX.
- Incrementa Overhead end-to-end para ofrecer confiabilidad.
- Requiere establecimiento de conexión (y cierre).
- Servicio FDX.
- Aplicaciones: transferencia de archivos, FTP/HTTP/SMTP/acceso remoto(SSH, telnet,...)/Unicast.

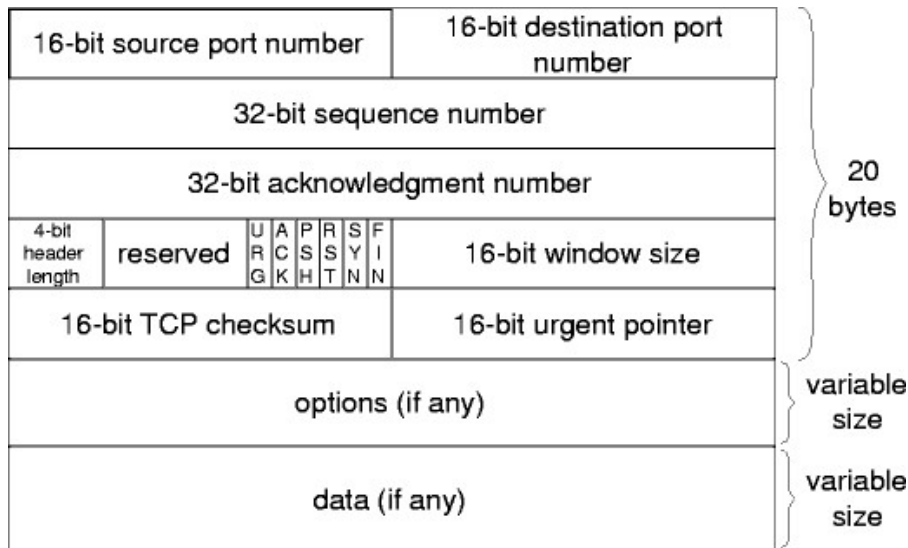
# Headers/Encabezados y Servicios de TCP

- El encabezado IP provee: Ruteo, Fragmentación, Detección de algunos errores, pero no hace controles.
- El encabezado TCP provee: MUX/DEMUX de aplic. , Detección de errores (obligatorio) y controles para hacerlo confiable.
- IP indica que lleva TCP con el código de protocolo 6.

```
? grep tcp /etc/protocols
```

```
tcp      6          TCP # transmission control protocol
```

# Segmento TCP



# Segmento TCP (Cont.)

The image shows a Wireshark packet capture window titled "00-tcp.pcap". The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar with various icons. A packet list on the left shows a single packet (No. 1) at time 0.000000, from source 172.20.1.1 to destination 172.20.1.100, protocol TCP, length 74 bytes. The packet details pane on the right shows the following information:

- Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
- Ethernet II, Src: 32:26:ff:bc:e7:b2 (32:26:ff:bc:e7:b2), Dst: RealtekU\_12:34:56 (52:54:00:12:34:56)
- Internet Protocol Version 4, Src: 172.20.1.1, Dst: 172.20.1.100
- Transmission Control Protocol, Src Port: 41749, Dst Port: 11111, Seq: 0, Len: 0
  - Source Port: 41749
  - Destination Port: 11111
  - [Stream index: 0]
  - [Conversation completeness: Complete, WITH\_DATA (31)]
  - [TCP Segment Len: 0]
  - Sequence Number: 0 (relative sequence number)
  - Sequence Number (raw): 3933822137
  - [Next Sequence Number: 1 (relative sequence number)]
  - Acknowledgment Number: 0
  - Acknowledgment number (raw): 0
  - 1010 .... = Header Length: 40 bytes (10)
  - Flags: 0x002 (SYN)
  - Window: 5840
    - [Calculated window size: 5840]
  - Checksum: 0xa1bb [unverified]
  - [Checksum Status: Unverified]
  - Urgent Pointer: 0
  - Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale

The status bar at the bottom indicates "Transmission Control Protocol: Protocol", "Packets: 16 · Displayed: 16 (100.0%)", and "Profile: Classic".

# Campos del Segmento TCP

- Puertos: MUX/DEMUX.
- No tiene Longitud total, si de HDR LEN (variable, max 60B Unit=4B).
- Total LEN se computa para PseudoHDR, no viaja en el segmento.
- El tamaño del segmento se calcula dentro del datagrama IP.
- Checksum:
  - Cálculo Ca1. Obligatorio, calculado, igual que UDP.
  - Si tiene error podría pedir retransmisión, implementación de TCP descarta y espera RTO (Retransmission Timer).
- Necesidad de manejar Timers, RTO (tmout. por cada segmento). (implementaciones lo manejan más eficientemente).

# Campos del Segmento TCP (Cont.)

- Campos de Sesiones: Flags: SYN(Synchronize), FIN(Finish), RST(Reset).
- Campo de Detección de Errores: Checksum.
- Campos de Control de Errores: ACK, Num. Sec (\#Seq), Num. Ack (#Ack).
- Campo de Control de Flujo: a los de control de errores se agrega, Win.
- Campos de Control de Congestión: se agregan flags si participa la red.



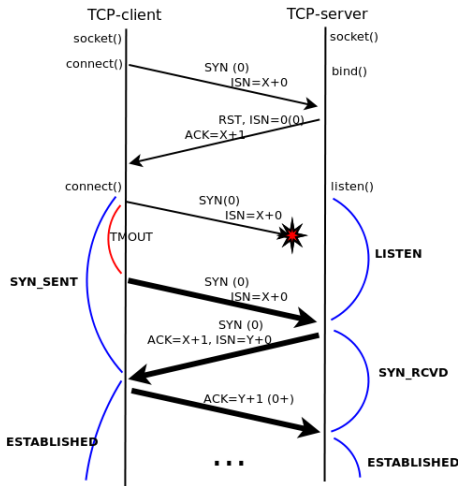
# Funcionamiento de TCP

- TCP entrega y envía los datos agrupados o separados de forma dis-asociada de la aplicación:
  - La aplicación puede enviar 3000 bytes en un write y TCP lo podría enviar en 3 segmentos separados de 1000 bytes c/u.
  - La aplicación puede enviar 100 bytes y luego otros 200 y TCP esperar para enviarlos todos juntos en un segmento de 300 bytes.
  - La aplicación puede intentar leer 200 bytes del buffer y TCP solo entregar 150 bytes y luego el resto.
- Requiere mantener “recursos” en cada extremo para los controles:
  - Buffers de Rx y Tx.
  - Timer(s) RTO.
  - Variables: datos enviados, no confirmados, retransmisiones, umbrales, RTT, etc.
  - Estado de la conexión.

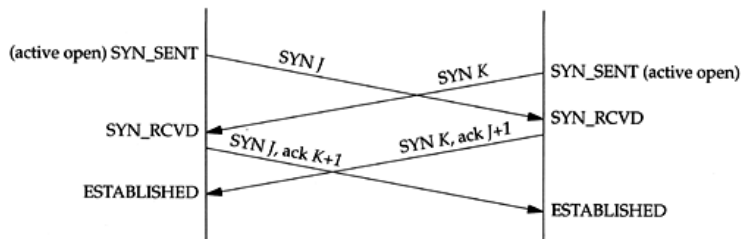
# TCP Establecimiento de Conexión

- Flags: SYN (Synchronize), ACK (Acknowledge) y RST (Reset).
- 3Way-Handshake (3WH).
- En el 3 segmento se puede enviar info.
- el ISN (Initial Sequence Number), se utiliza un contador que se incrementa cada 4 mseg.
- RST si no hay proceso en estado LISTEN.
- Open Pasivo (servidor) y Activo (cliente).
- Open simultáneo.

# 3 Way Handshake



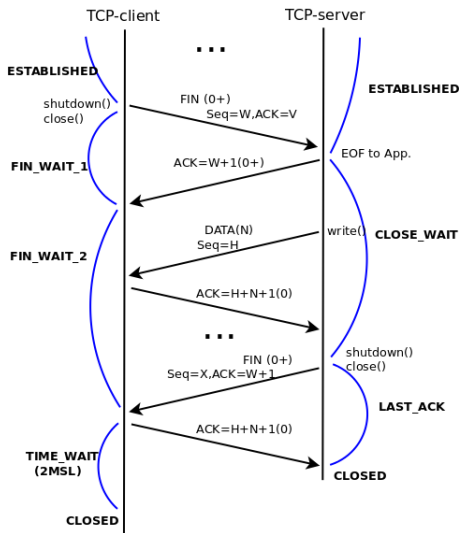
# 3 Way Handshake (Open Simultáneo)



# TCP Cierre de Conexión

- Flags: FIN (Finish), ACK y RST.
- 4Way-Close (4WC).
- Posibilidad de Half-Close.
- Podría cerrarse en 3WC.
- Espera en TIME\_WAIT, 2MSL (aprox. 2\*2min).
- Evitar con SO\_REUSEADDR.
- Cierre incorrecto con RST.
- Close simultáneo.

# TCP Close



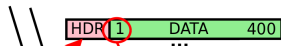
# Orientado a streams (secuencia en orden de bytes)

time (tiempo) →

Sender

TCP Segment data

1 2 3 4 ... 200 ... 301 ... 400 | 401... 500 ... 600 ... 800 | 801... 900 ... 1000 ... 1100



1 2 3 4 ... 200 ... 301 ... 400 401... 500 ... 600 ... 800 801... 900

Receiver

#Seq

# Nums. de Secuencia/ACK TCP

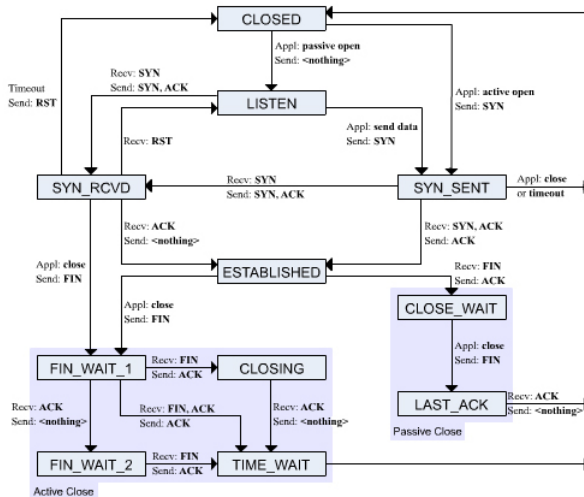
Time	172.20.1.1	172.20.1.100	Comment
0.000	(41749) →	SYN → (11111)	Seq = 0
0.001	(41749) ←	SYN, ACK → (11111)	Seq = 0 Ack = 1
0.001	(41749) →	ACK → (11111)	Seq = 1 Ack = 1
90.730	(41749) →	PSH, ACK - Len: 5 → (11111)	Seq = 1 Ack = 1
90.730	(41749) →	ACK → (11111)	Seq = 1 Ack = 6
100.150	(41749) →	PSH, ACK - Len: 16 → (11111)	Seq = 1 Ack = 6
100.150	(41749) →	ACK → (11111)	Seq = 6 Ack = 17
104.580	(41749) →	PSH, ACK - Len: 5 → (11111)	Seq = 6 Ack = 17
104.580	(41749) →	ACK → (11111)	Seq = 17 Ack = 11
112.290	(41749) →	PSH, ACK - Len: 6 → (11111)	Seq = 17 Ack = 11
112.290	(41749) →	ACK → (11111)	Seq = 11 Ack = 23
114.890	(41749) →	PSH, ACK - Len: 6 → (11111)	Seq = 23 Ack = 11
114.890	(41749) →	ACK → (11111)	Seq = 11 Ack = 29
120.620	(41749) →	FIN, ACK → (11111)	Seq = 29 Ack = 11
120.620	(41749) →	FIN, ACK → (11111)	Seq = 11 Ack = 30
120.620	(41749) →	ACK → (11111)	Seq = 30 Ack = 12



# Control de Errores TCP

- Errores que pueden existir en IP:
  - Pérdida de paquetes: descartados.
  - Des-ordenados, retardados.
  - Duplicados.
  - Corrompidos.
- TCP intenta solucionarlos con el control de Errores que implementa.

# TCP, Diagrama de Estados (FSM)



- [Stevl] TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1 ed. 1994. W. Richard Stevens. 2 ed. K. Fall, W. Stevens.
- [KR] Kurose/Ross: Computer Networking (8th Edition).
- [RFC-768] <http://www.rfc-editor.org/rfc/rfc768.txt>. User Datagram Protocol (Jon Postel 1980 USC-ISI IANA).
- [RFC-793] <http://www.rfc-editor.org/rfc/rfc793.txt>. TCP Transmission Control Protocol (Jon Postel 1981 USC-ISI IANA).
- [STANFORD-CS144] Introduction to Computer Networking, Stanford Course.
- [TCPIPG] TCP/IP Guide: <http://www.tcpipguide.com/>.
- [COM05] Ethereal, Wireshark. Autor original Gerald Combs, 2005.  
<http://www.ethereal.com/>.  
<http://www.wireshark.org/>.