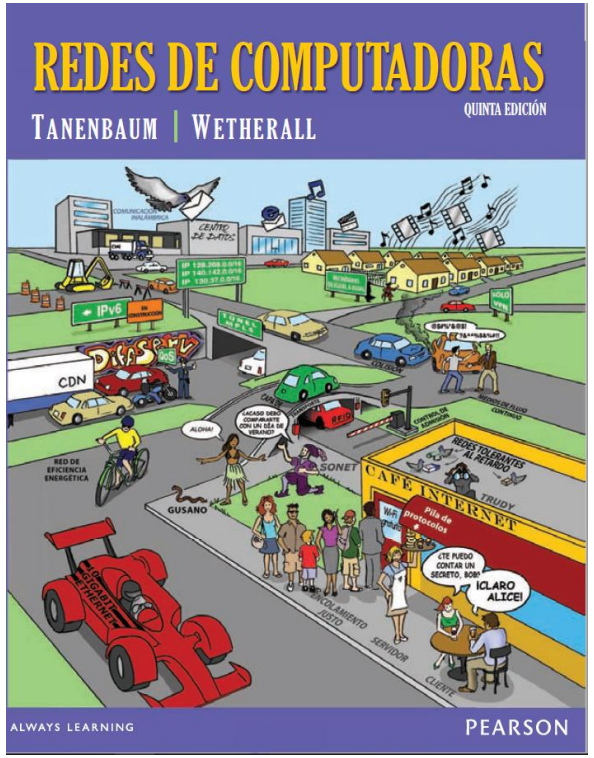
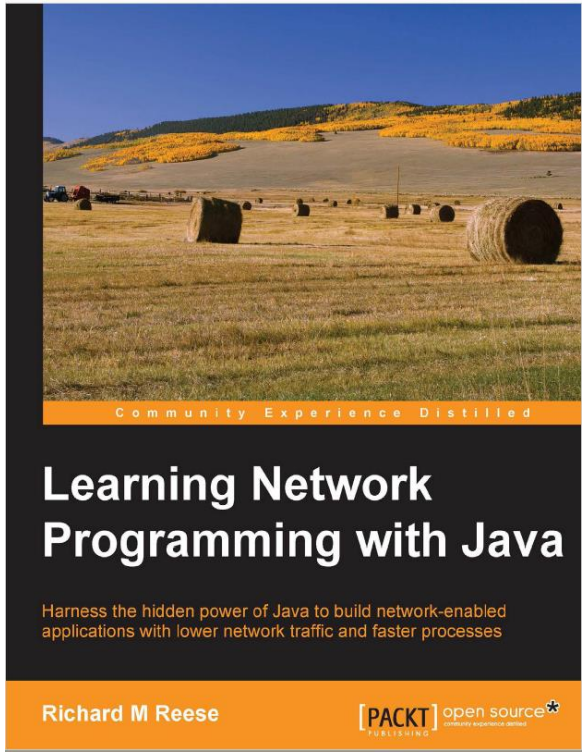
REDES DE COMPUTADORES Y LABORATORIO

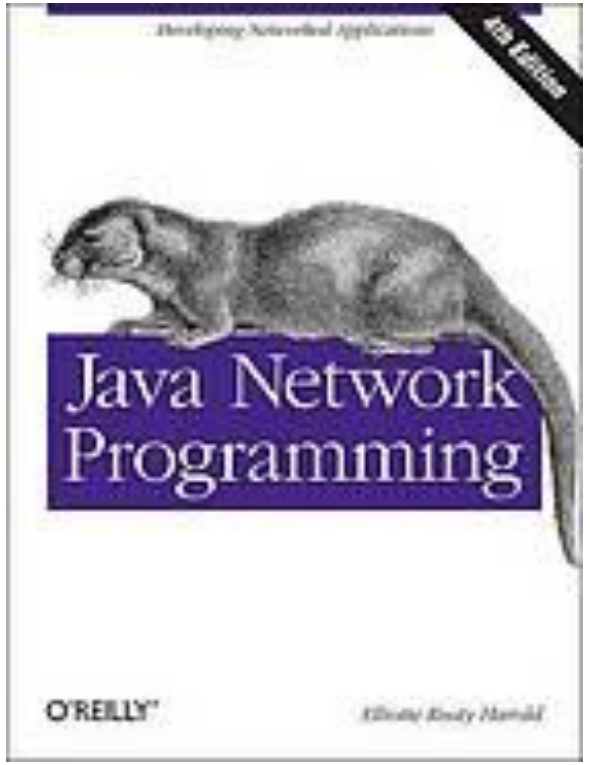
Christian Camilo Urcuqui López, MSc

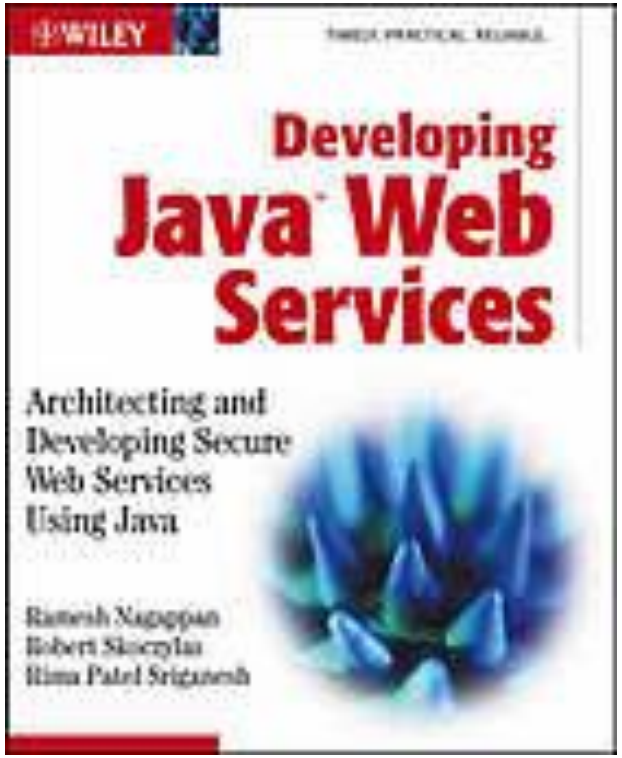
UNIVERSIDAD ICESI | A OTRO NIVEL

BIBLIOGRAFÍA



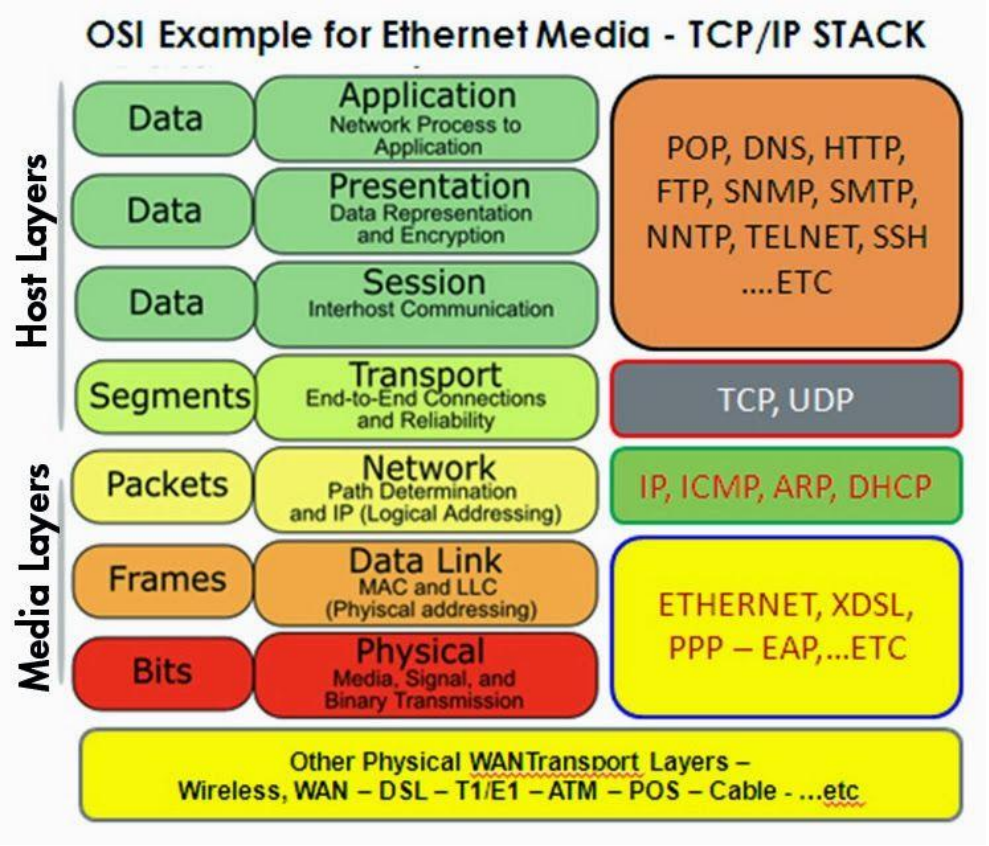






COMPETENCIAS

- Describir el uso de las primitivas.
- Describir UDP.
- Describir TCP.



LA CAPA DE TRANSPORTE

- Recordemos…. La capa de red provee entrega de paquetes punto a punto mediante el uso de datagramas o circuitos virtuales.
- El objetivo de la capa de transporte es proporcionar un servicio de transmisión de datos eficiente, confiable y económico a sus usuarios, procesos que normalmente son de la capa de aplicación.
- Gracias a esta capa, los programadores pueden escribir código de acuerdo con un conjunto estándar de **primitivas**; estos programas pueden funcionar en una amplia variedad de redes sin necesidad de preocuparse por lidiar con diferentes interfaces de red y distintos niveles de confiabilidad.

PRIMITIVAS DEL SERVICIO DE TRANSPORTE

- La capa de transporte debe proporcionar algunas operaciones a los programas de aplicación (software de alto nivel) a través de una interfaz de servicios.
- Cada servicio tiene su propia interfaz.
- Interfaz orientada a la conexión.

Primitiva	Paquete enviado	Significado
LISTEN	(ninguno)	Se bloquea hasta que algún proceso intenta conectarse.
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión.
SEND	DATA	Envía información.
RECEIVE	(ninguno)	Se bloquea hasta que llegue un paquete DATA.
DISCONNECT	DISCONNECTION REQ.	Solicita que se libere la conexión

- Los mensajes enviados a través de una entidad de transporte a otra se conocen como **segmentos.**

PRIMITIVAS DEL SERVICIO DE TRANSPORTE

- Interfaz **orientada a la conexión.**

Primitiva	Paquete enviado	Significado
LISTEN	(ninguno)	Se bloquea hasta que algún proceso intenta conectarse.
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión.
SEND	DATA	Envía información.
RECEIVE	(ninguno)	Se bloquea hasta que llegue un paquete DATA.
DISCONNECT	DISCONNECTION REQ.	Solicita que se libere la conexión

- Los mensajes enviados a través de una entidad de transporte a otra se conocen como **segmentos.**

PRIMITIVAS DEL SERVICIO DE TRANSPORTE

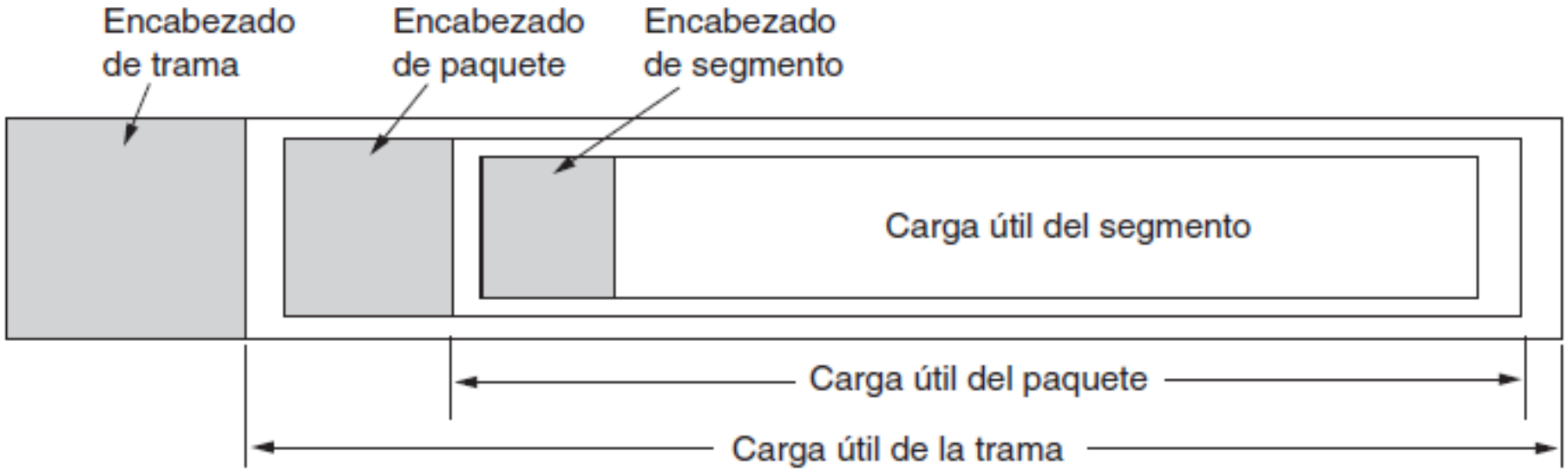


Figura 6-3. Anidamiento de segmentos, paquetes y tramas.

Primitiva	Paquete enviado	Significado
LISTEN	(ninguno)	Se bloquea hasta que algún proceso intenta conectarse.
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión.
SEND	DATA	Envía información.
RECEIVE	(ninguno)	Se bloquea hasta que llegue un paquete DATA.
DISCONNECT	DISCONNECTION REQ.	Solicita que se libere la conexión

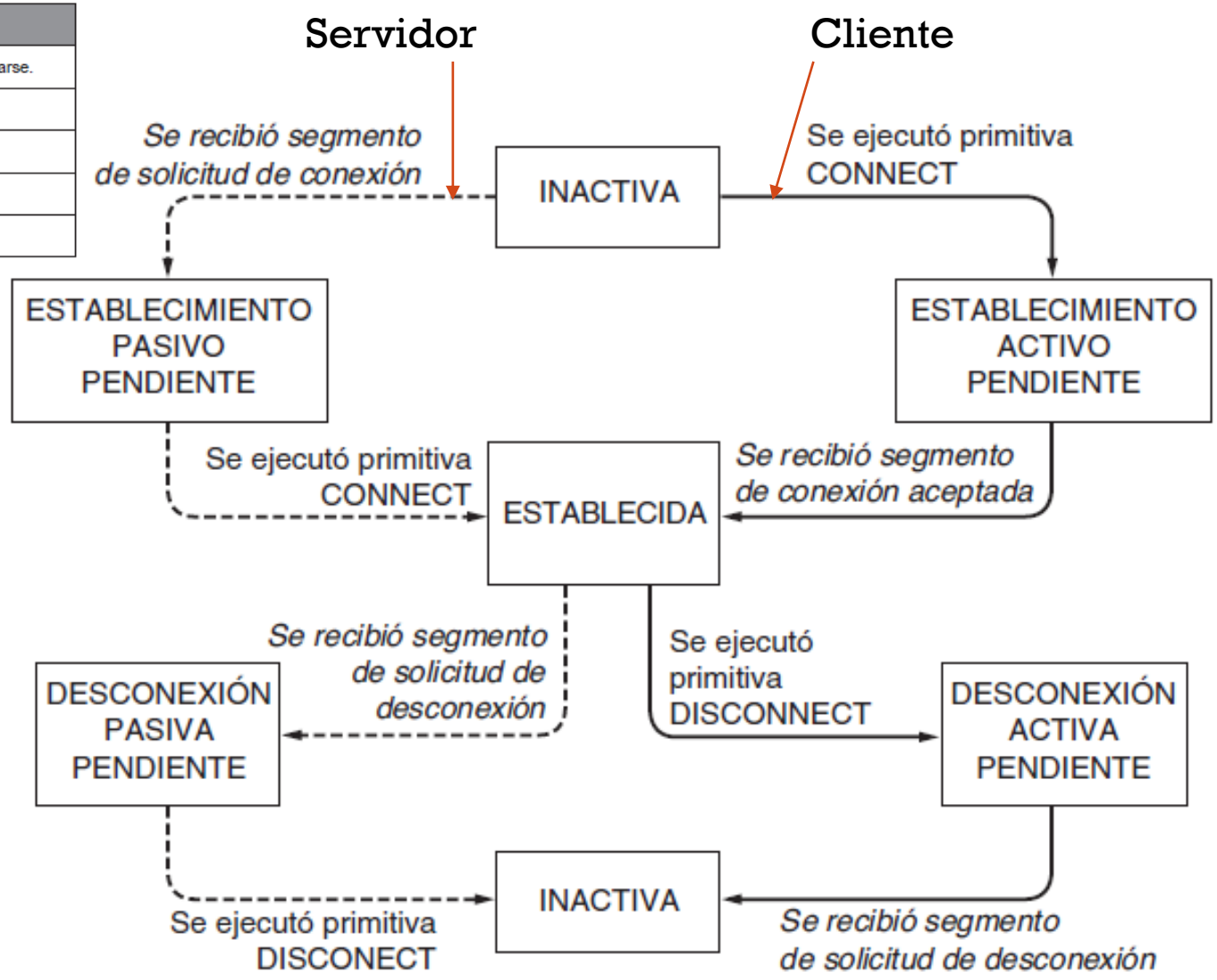


Figura 6-4. Un diagrama de estado para un esquema simple de manejo de conexiones. Las transiciones etiquetadas en cursiva se producen debido a la llegada de paquetes. Las líneas continuas muestran la secuencia de estados del cliente. Las líneas punteadas muestran la secuencia de estados del servidor.

SOCKETS DE BERKERLEY

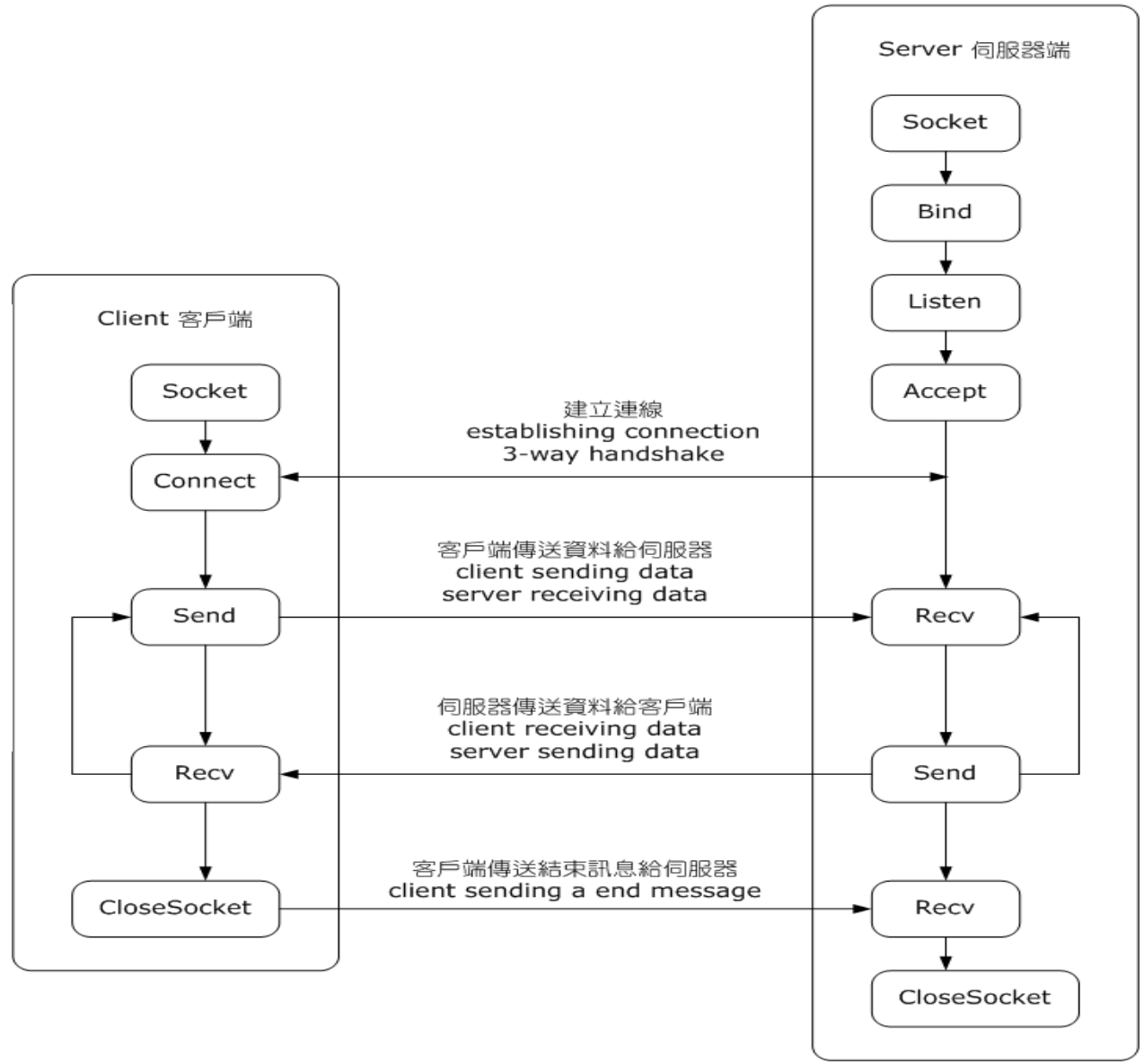
- Son un conjunto de primitivas de transporte: las primitivas de socket que se utilizan para TCP.

Primitiva	Significado
SOCKET	Crea un nuevo punto terminal de comunicación.
BIND	Asocia una dirección local con un socket.
LISTEN	Anuncia la disposición de aceptar conexiones; indica el tamaño de la cola.
ACCEPT	Establece en forma pasiva una conexión entrante.
CONNECT	Intenta establecer activamente una conexión.
SEND	Envía datos a través de la conexión.
RECEIVE	Recibe datos de la conexión.
CLOSE	Libera la conexión.

Figura 6-5. Las primitivas de socket para TCP.

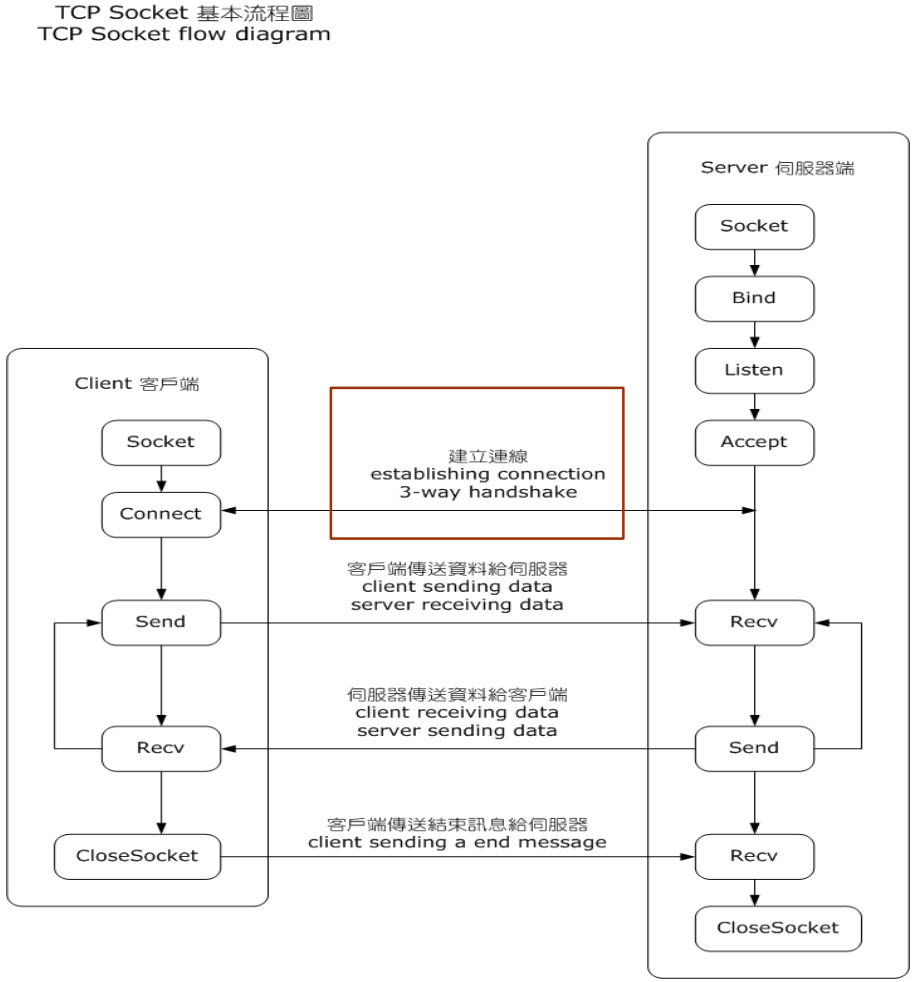
Primitiva	Significado
SOCKET	Crea un nuevo punto terminal de comunicación.
BIND	Asocia una dirección local con un socket.
LISTEN	Anuncia la disposición de aceptar conexiones; indica el tamaño de la cola.
ACCEPT	Establece en forma pasiva una conexión entrante.
CONNECT	Intenta establecer activamente una conexión.
SEND	Envía datos a través de la conexión.
RECEIVE	Recibe datos de la conexión.
CLOSE	Libera la conexión.

Figura 6-5. Las primitivas de socket para TCP.



DIRECCIONAMIENTO Y ESTABLECIMIENTO DE CONEXIÓN

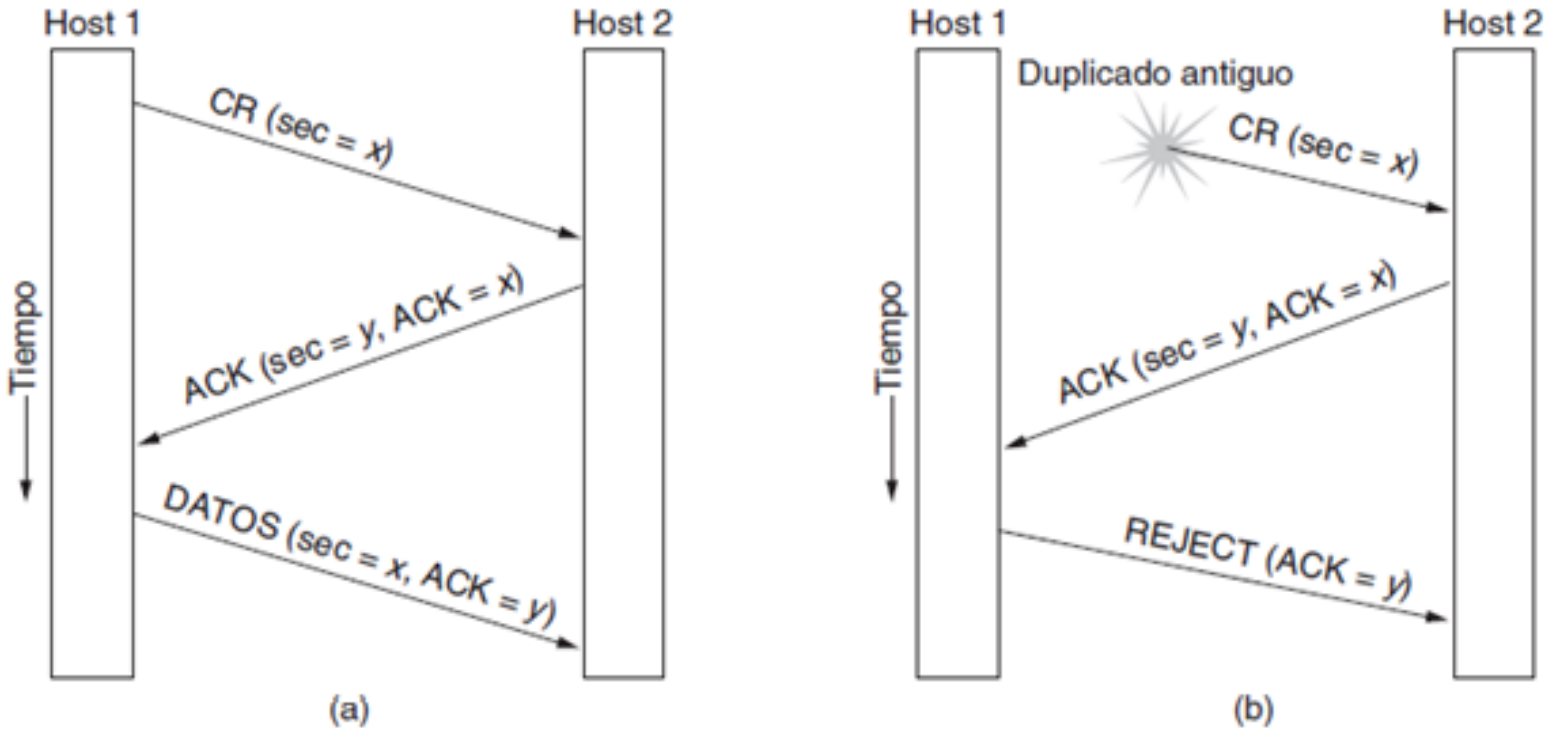
- El método que permite definir las direcciones de transporte en las que los procesos pueden escuchar solicitudes de conexión es a través de puntos terminales conocidos cómo **puertos.**
- **Acuerdo de tres vías** (*three-way handhsake*). Es un protocolo que implica que un igual (receptor) verifique que la conexión sea realmente la actual.



UNIVERSIDAD ICESI | A OTRO NIVEL

SEGMENTOS

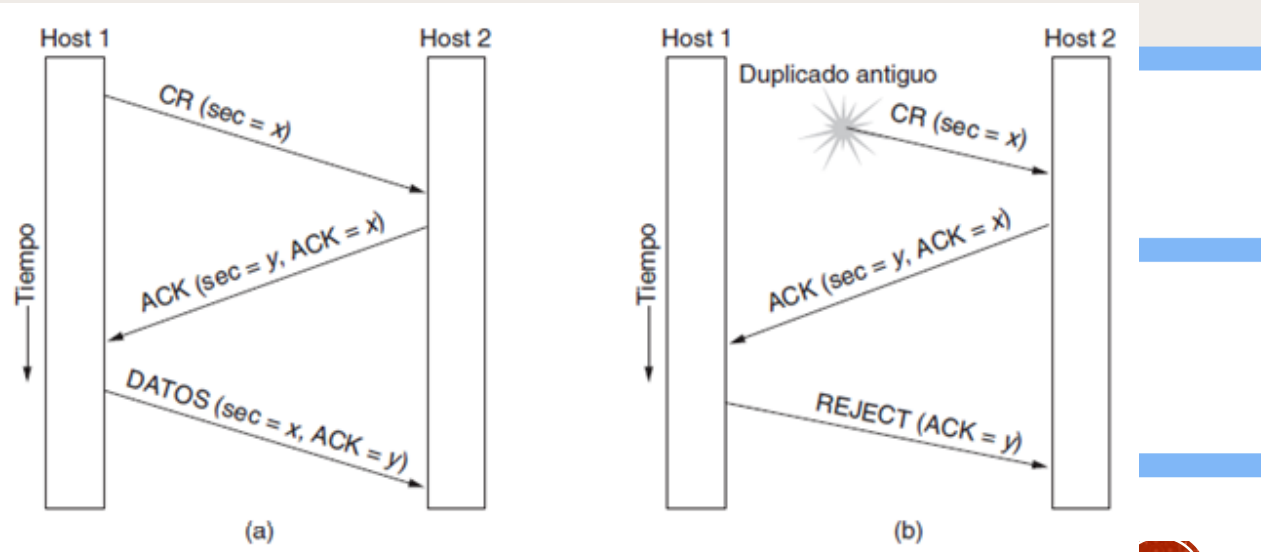
- CR (CONNECTION REQUEST)
- DR (DISCONNECTION REQUEST)
- ACK (acknowledgement), es un mensaje que el destino de la comunicación envía al origen de esta para confirmar la recepción del mensaje.



Establecimiento de una conexión

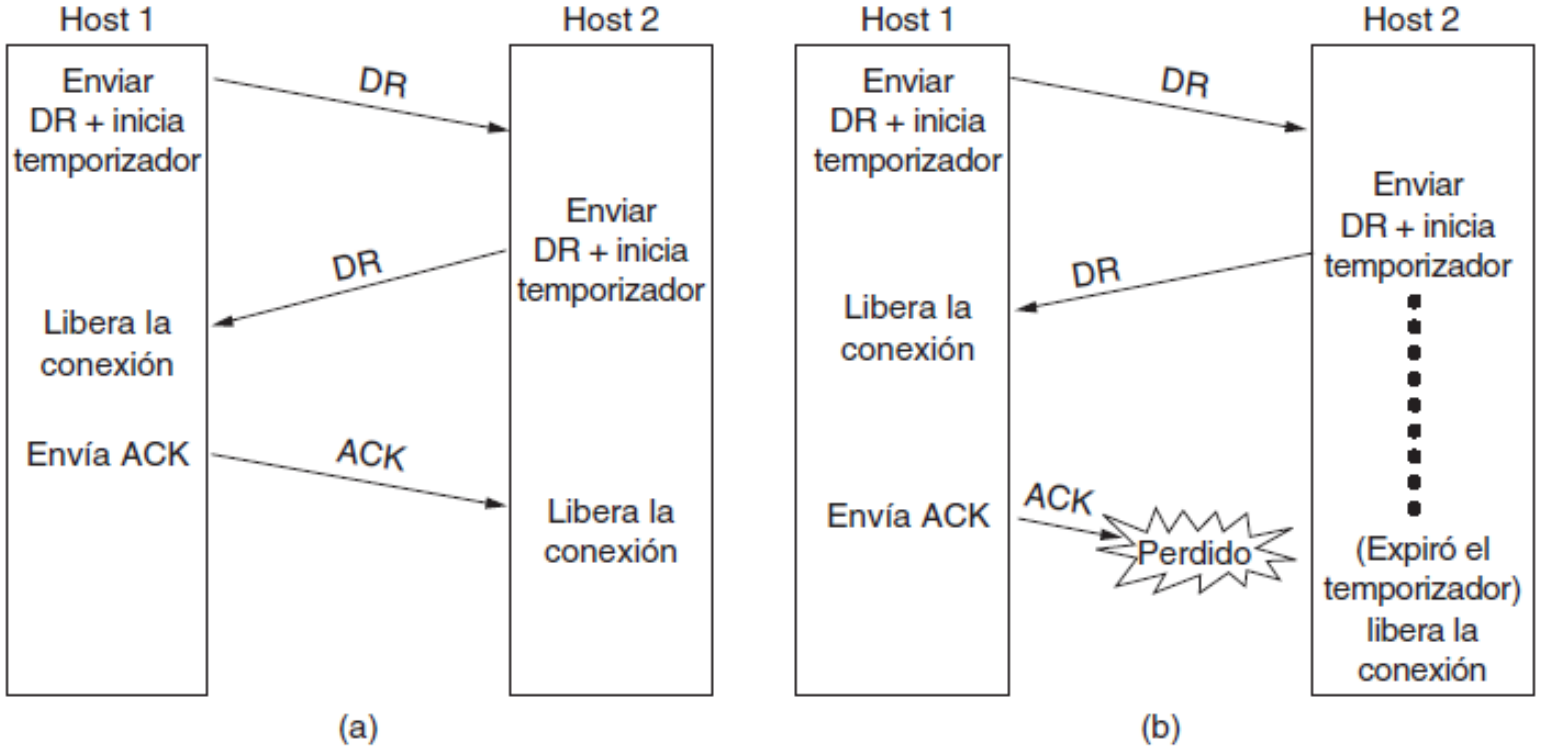
1	0.000000000	192.168.17.131	52.84.82.78	TCP	54	41020 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0
2	0.000369246	52.84.82.78	192.168.17.131	TCP	60	[TCP ACKed unseen segment] 443 → 41020 [ACK] Seq=1 Ack=2 Win=64240 Len=0
3	3.597386275	192.168.17.131	192.168.17.2	DNS	76	Standard query 0xaeb6 A www.icesi.edu.co
4	3.597661674	192.168.17.131	192.168.17.2	DNS	76	Standard query 0xd535 AAAA www.icesi.edu.co
5	3.602281448	192.168.17.2	192.168.17.131	DNS	126	Standard query response 0xd535 AAAA www.icesi.edu.co SOA kodos
6	3.602381441	192.168.17.2	192.168.17.131	DNS	111	Standard query response 0xaeb6 A www.icesi.edu.co A 200.3.192.46 NS kodos
7	3.603280817	192.168.17.131	200.3.192.46	TCP	74	57204 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=582435103 TSecr=0 WS=128
8	3.608419269	200.3.192.46	192.168.17.131	TCP	60	80 → 57204 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
9	3.608551103	192.168.17.131	200.3.192.46	TCP	54	57204 → 80 [ACK] Seq=1 Ack=1 Win=29200 Len=0
10	3.609932566	192.168.17.131	200.3.192.46	HTTP	370	GET / HTTP/1.1
11	3.610799542	200.3.192.46	192.168.17.131	TCP	60	80 → 57204 [ACK] Seq=1 Ack=317 Win=64240 Len=0
12	4.162888054	200.3.192.46	192.168.17.131	TCP	29254	[TCP Window Full] 80 → 57204 [PSH, ACK] Seq=1 Ack=317 Win=64240 Len=29200 [TCP segment of a reassembled PDU]
13	4.163036367	192.168.17.131	200.3.192.46	TCP	54	57204 → 80 [ACK] Seq=317 Ack=29201 Win=64240 Len=0
14	4.163265121	200.3.192.46	192.168.17.131	TCP	3559	80 → 57204 [PSH, ACK] Seq=29201 Ack=317 Win=64240 Len=3505 [TCP segment of a reassembled PDU]
15	4.163294728	192.168.17.131	200.3.192.46	TCP	54	57204 → 80 [ACK] Seq=317 Ack=32706 Win=64240 Len=0
16	4.169553490	200.3.192.46	192.168.17.131	TCP	64293	80 → 57204 [PSH, ACK] Seq=32706 Ack=317 Win=64240 Len=64239 [TCP segment of a reassembled PDU]
17	4.169655717	192.168.17.131	200.3.192.46	TCP	54	57204 → 80 [ACK] Seq=317 Ack=96945 Win=64240 Len=0
18	4.170104567	200.3.192.46	192.168.17.131	TCP	64293	80 → 57204 [PSH, ACK] Seq=96945 Ack=317 Win=64240 Len=64239 [TCP segment of a reassembled PDU]
19	4.170143670	192.168.17.131	200.3.192.46	TCP	54	57204 → 80 [ACK] Seq=317 Ack=161184 Win=65535 Len=0
20	4.170466272	200.3.192.46	192.168.17.131	HTTP	13600	HTTP/1.1 200 OK (text/html)
21	4.170499231	192.168.17.131	200.3.192.46	TCP	54	57204 → 80 [ACK] Seq=317 Ack=174731 Win=65535 Len=0
22	4.171320679	192.168.17.131	200.3.192.46	TCP	54	57204 → 80 [FIN, ACK] Seq=317 Ack=174731 Win=65535 Len=0
23	4.171805001	200.3.192.46	192.168.17.131	TCP	60	80 → 57204 [ACK] Seq=174731 Ack=318 Win=64239 Len=0
24	4.288429458	192.168.17.131	200.3.192.46	TCP	74	57206 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=582435788 TSecr=0 WS=128

```
▸ Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▸ Ethernet II, Src: Vmware_b3:d3:87 (00:0c:29:b3:d3:87), Dst: Vmware_e3:c6:70 (00:50:56:e3:c6:70)
▸ Internet Protocol Version 4, Src: 192.168.17.131, Dst: 200.3.192.46
▾ Transmission Control Protocol, Src Port: 57204, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 57204
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
    1010 .... = Header Length: 40 bytes (10)
  ▾ Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ▸ .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ··········S·]
    Window size value: 29200
```



Algunos **segmentos**:

- ACK (acknowledgement), es un mensaje que el destino de la comunicación envía al origen de esta para confirmar la recepción del mensaje.
- CR (CONNECTION REQUEST)
- DR (DISCONNECTION REQUEST)



Liberación de una conexión

474	10.382472924	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=40554 Win=64240 Len=0
475	10.383543723	200.3.192.46	192.168.17.131	TCP	38718	80 → 57482 [PSH, ACK] Seq=40554 Ack=517 Win=64240 Len=38664 [TCP segment of a reassembled PDU]
476	10.383675862	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=79218 Win=64240 Len=0
477	10.385289185	200.3.192.46	192.168.17.131	TCP	21666	80 → 57482 [PSH, ACK] Seq=79218 Ack=517 Win=64240 Len=21612 [TCP segment of a reassembled PDU]
478	10.385343941	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=100830 Win=64240 Len=0
479	10.385641183	200.3.192.46	192.168.17.131	TCP	14750	80 → 57482 [PSH, ACK] Seq=100830 Ack=517 Win=64240 Len=14696 [TCP segment of a reassembled PDU]
480	10.385679735	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=115526 Win=64240 Len=0
481	10.387050851	200.3.192.46	192.168.17.131	TCP	31371	80 → 57482 [PSH, ACK] Seq=115526 Ack=517 Win=64240 Len=31317 [TCP segment of a reassembled PDU]
482	10.387265402	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=146843 Win=65535 Len=0
483	10.387717411	200.3.192.46	192.168.17.131	TCP	6885	80 → 57482 [PSH, ACK] Seq=146843 Ack=517 Win=64240 Len=6831 [TCP segment of a reassembled PDU]
484	10.387993981	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=153674 Win=65535 Len=0
485	10.390163796	200.3.192.46	192.168.17.131	TCP	64293	80 → 57482 [PSH, ACK] Seq=153674 Ack=517 Win=64240 Len=64239 [TCP segment of a reassembled PDU]
486	10.390219669	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=217913 Win=65535 Len=0
487	10.390597089	200.3.192.46	192.168.17.131	TCP	1728	80 → 57482 [PSH, ACK] Seq=217913 Ack=517 Win=64240 Len=1674 [TCP segment of a reassembled PDU]
488	10.390631450	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=219587 Win=65535 Len=0
489	10.391192691	200.3.192.46	192.168.17.131	TCP	39634	80 → 57482 [PSH, ACK] Seq=219587 Ack=517 Win=64240 Len=39580 [TCP segment of a reassembled PDU]
490	10.391241021	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=259167 Win=65535 Len=0
491	10.391591064	200.3.192.46	192.168.17.131	TCP	6933	80 → 57482 [PSH, ACK] Seq=259167 Ack=517 Win=64240 Len=6879 [TCP segment of a reassembled PDU]
492	10.391627381	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=266046 Win=65535 Len=0
493	10.392097829	200.3.192.46	192.168.17.131	TCP	8772	80 → 57482 [PSH, ACK] Seq=266046 Ack=517 Win=64240 Len=8718 [TCP segment of a reassembled PDU]
494	10.392135263	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=274764 Win=65535 Len=0
495	10.393834351	200.3.192.46	192.168.17.131	TCP	7778	80 → 57482 [PSH, ACK] Seq=274764 Ack=517 Win=64240 Len=7724 [TCP segment of a reassembled PDU]
496	10.393872065	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=282488 Win=65535 Len=0
497	10.394896770	200.3.192.46	192.168.17.131	TCP	27781	80 → 57482 [PSH, ACK] Seq=282488 Ack=517 Win=64240 Len=27727 [TCP segment of a reassembled PDU]
498	10.395080591	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=310215 Win=65535 Len=0
499	10.398276787	200.3.192.46	192.168.17.131	TCP	8025	80 → 57482 [PSH, ACK] Seq=310215 Ack=517 Win=64240 Len=7971 [TCP segment of a reassembled PDU]
500	10.398316177	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=318186 Win=65535 Len=0
501	10.398632137	200.3.192.46	192.168.17.131	TCP	6336	80 → 57482 [PSH, ACK] Seq=318186 Ack=517 Win=64240 Len=6282 [TCP segment of a reassembled PDU]
502	10.398662029	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=324468 Win=65535 Len=0
503	10.401867165	200.3.192.46	192.168.17.131	TCP	10273	80 → 57482 [PSH, ACK] Seq=324468 Ack=517 Win=64240 Len=10219 [TCP segment of a reassembled PDU]
504	10.401906834	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=334687 Win=65535 Len=0
505	10.402630385	200.3.192.46	192.168.17.131	TCP	32742	80 → 57482 [PSH, ACK] Seq=334687 Ack=517 Win=64240 Len=32688 [TCP segment of a reassembled PDU]
506	10.402676480	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=367375 Win=65535 Len=0
507	10.404840429	200.3.192.46	192.168.17.131	TCP	64293	80 → 57482 [PSH, ACK] Seq=367375 Ack=517 Win=64240 Len=64239 [TCP segment of a reassembled PDU]
508	10.405029558	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=431614 Win=65535 Len=0
509	10.405360324	200.3.192.46	192.168.17.131	TCP	25890	80 → 57482 [PSH, ACK] Seq=431614 Ack=517 Win=64240 Len=25836 [TCP segment of a reassembled PDU]
510	10.405399714	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=457450 Win=65535 Len=0
511	10.407244350	200.3.192.46	192.168.17.131	TCP	59533	80 → 57482 [PSH, ACK] Seq=457450 Ack=517 Win=64240 Len=59479 [TCP segment of a reassembled PDU]
512	10.407294636	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=516929 Win=65535 Len=0
513	10.409103234	200.3.192.46	192.168.17.131	TCP	64293	80 → 57482 [PSH, ACK] Seq=516929 Ack=517 Win=64240 Len=64239 [TCP segment of a reassembled PDU]
514	10.409182573	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=581168 Win=65535 Len=0
515	10.409493784	200.3.192.46	192.168.17.131	TCP	8946	80 → 57482 [PSH, ACK] Seq=581168 Ack=517 Win=64240 Len=8892 [TCP segment of a reassembled PDU]
516	10.410489994	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=590060 Win=65535 Len=0
517	10.417655384	200.3.192.46	192.168.17.131	HTTP	27939	HTTP/1.1 200 OK (text/css)
518	10.417773834	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [ACK] Seq=517 Ack=617946 Win=65535 Len=0
519	10.418843237	192.168.17.131	200.3.192.46	TCP	54	57482 → 80 [FIN, ACK] Seq=517 Ack=617946 Win=65535 Len=0
520	10.419155006	200.3.192.46	192.168.17.131	TCP	60	80 → 57482 [ACK] Seq=617946 Ack=518 Win=64239 Len=0

LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: UDP

UDP (USER DATAGRAM PROTOCOL)

- Proporciona una forma para que las aplicaciones envíen datagramas IP encapsulados sin tener que establecer una conexión. RFC 768.
- UDP es un protocolo simple para interacciones cliente/servidor y multimedia.
- UDP trasmite segmentos que consisten en un encabezado de 8 bytes seguido de la carga útil.

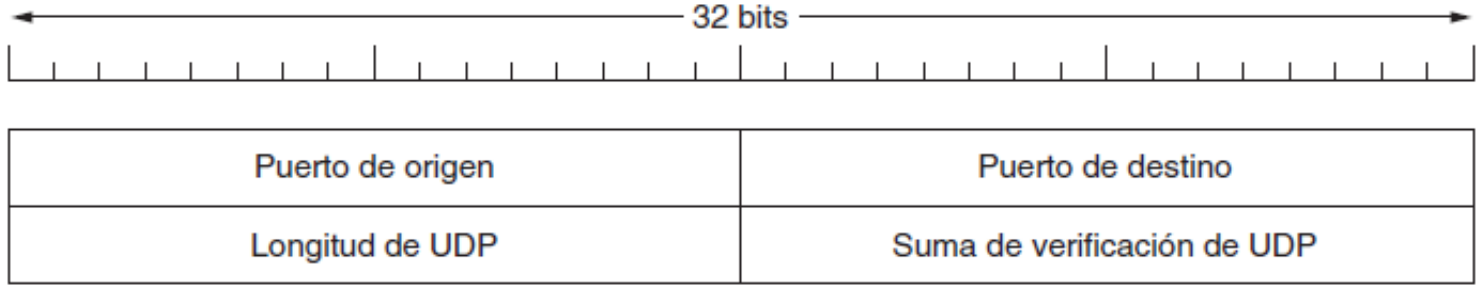


Figura 6-27. El encabezado UDP.

- La longitud incluye el encabezado de 8 bytes y los datos. La longitud mínima es de 8 bytes y la máxima es de 655515 bytes.

LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: TCP

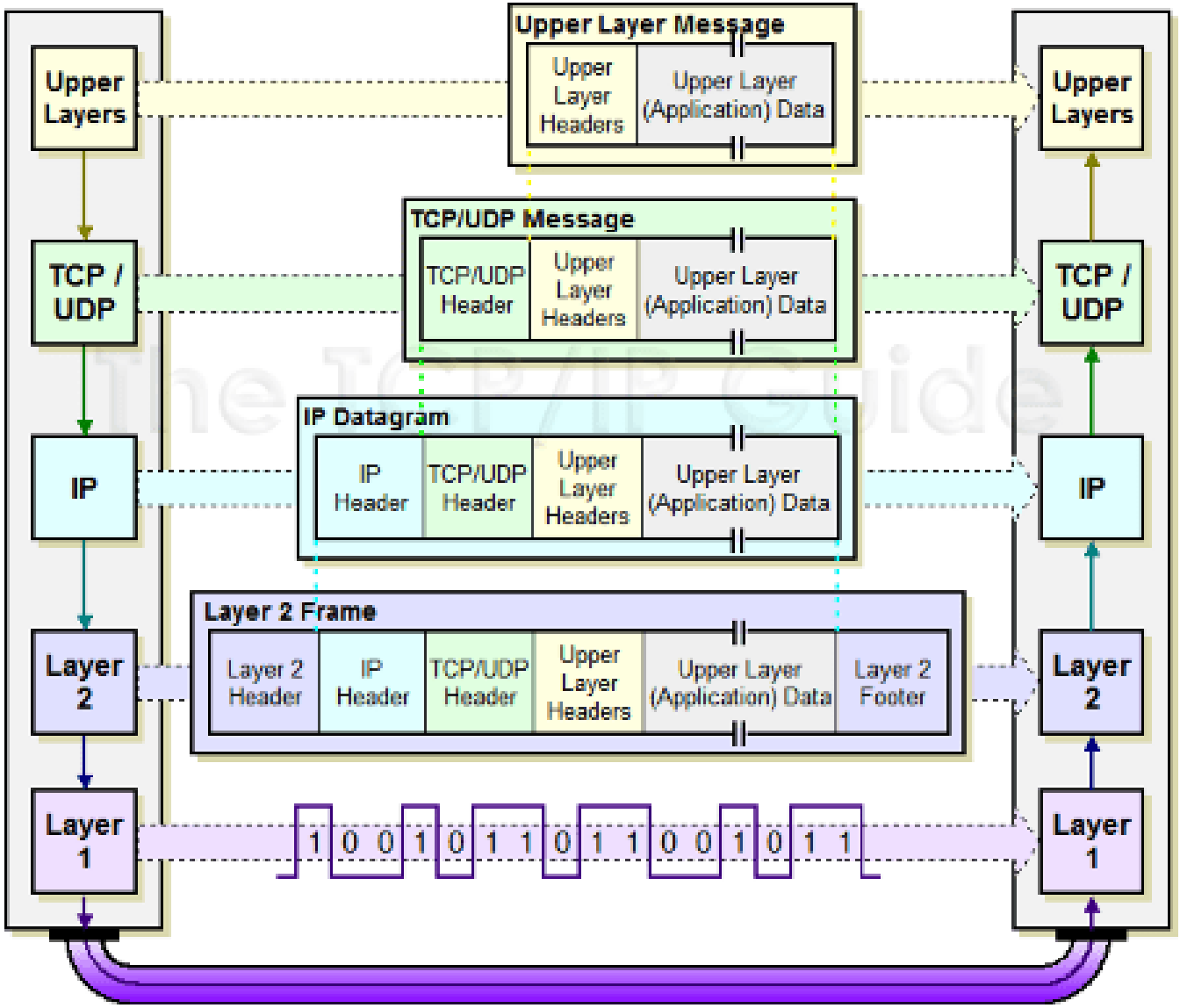
TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN

- Se diseño con la finalidad de proporcionar un flujo de bytes confiable de extremo a extremo a través de una interred no confiable. Roadmap RFC
- El servicio TCP se obtiene al hacer que tanto el servidor como el receptor creen puntos terminales, llamados **sockets.**
- Cada socket tiene un número (dirección) que consiste en la dirección IP del host y un número de 16 bits que es local para ese host, llamado **puerto**

Puerto	Protocolo	Uso
20, 21	FTP	Transferencia de archivos.
22	SSH	Inicio de sesión remoto, reemplazo de Telnet.
25	SMTP	Correo electrónico.
80	HTTP	World Wide Web.
110	POP-3	Acceso remoto al correo electrónico.
143	IMAP	Acceso remoto al correo electrónico.
443	HTTPS	Acceso seguro a web (HTTP sobre SSL/TLS).
543	RTSP	Control del reproductor de medios.
631	IPP	Compartición de impresoras.

Figura 6-34. Algunos puertos asignados.

La capa IP no ofrece garantía de que los datagramas se entregarán de manera apropiada y en el orden correcto; es trabajo de TCP incorporar los mecanismos que permitan tener un buen desempeño y confiabilidad.



TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN

- Todas las conexiones TCP son *full dúplex* y punto a punto.
- Una conexión TCP es un flujo de bytes, no un flujo de mensajes.

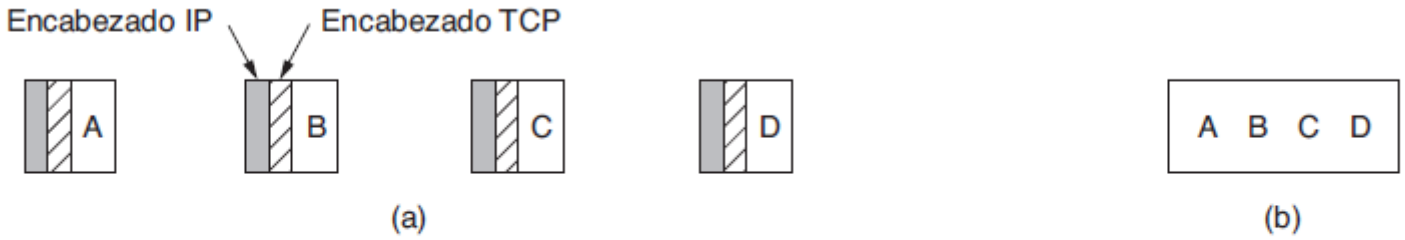


Figura 6-35. (a) Cuatro segmentos de 512 bytes que se envían como diagramas IP separados. (b) Los 2 048 bytes de datos que se entregan a la aplicación en una sola llamada READ.

- Cuando una aplicación pasa datos a TCP, éste decide entre enviarlos de inmediato o almacenarlos en el búfer. La bandera PUSH (PSH) sirve para forzar la salida de datos.

TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN

- URGENT es otra bandera que le indica a TCP que deje de acumular datos y transmita de inmediato todo lo que tiene para esa conexión (se utiliza cuando de un usuario presiona CTRL-C para interrumpir un cálculo remoto).
- Cada byte de una conexión de TCP tiene su propio número de secuencia de 32 bits.
- Un **segmento TCP** consiste en un encabezado fijo de 20 bytes, es decir, 160 bits (más una parte opcional), seguido de cero o más bytes de datos. **El software de TCP decide qué tan grandes deben ser los segmentos**
- Hay dos límites que restringen el tamaño del segmento.
  1. Cada segmento debe caber en la carga útil de 65515 bytes del IP.
  2. Cada segmento debe caber en la MTU (Unidad Máxima de Transferencia) del emisor y receptor con el fin de evitar fragmentación.

```
▸ Frame 509: 25890 bytes on wire (207120 bits), 25890 bytes captured (207120 bits) on interface 0
▸ Ethernet II, Src: Vmware_e3:c6:70 (00:50:56:e3:c6:70), Dst: Vmware_b3:d3:87 (00:0c:29:b3:d3:87)
▸ Internet Protocol Version 4, Src: 200.3.192.46, Dst: 192.168.17.131
▾ Transmission Control Protocol, Src Port: 80, Dst Port: 57482, Seq: 431614, Ack: 517, Len: 25836
    Source Port: 80
    Destination Port: 57482
    [Stream index: 13]
    [TCP Segment Len: 25836]
    Sequence number: 431614    (relative sequence number)
    [Next sequence number: 457450    (relative sequence number)]
    Acknowledgment number: 517    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  ▾ Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·······AP···]
    Window size value: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0xbf64 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
    TCP payload (25836 bytes)
    [Reassembled PDU in frame: 517]
    TCP segment data (25836 bytes)
```

TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN

- Las implementaciones modernas de TCP realizan el descubrimiento de MTU de la ruta mediante mensajes de error de ICMP con el fin de encontrar la ruta más pequeña

- 5 tupla: protocolo (TCP), dirección IP destino y origen, puerto destino y origen.
- Acknowledgement: confirmación de recepción.
- *CWR y ECE* para indicar congestión
- *SYN* se usa para establecer conexiones.
- *RST* se utiliza para reestablecer la conexión.
- *FIN* libera la conexión.

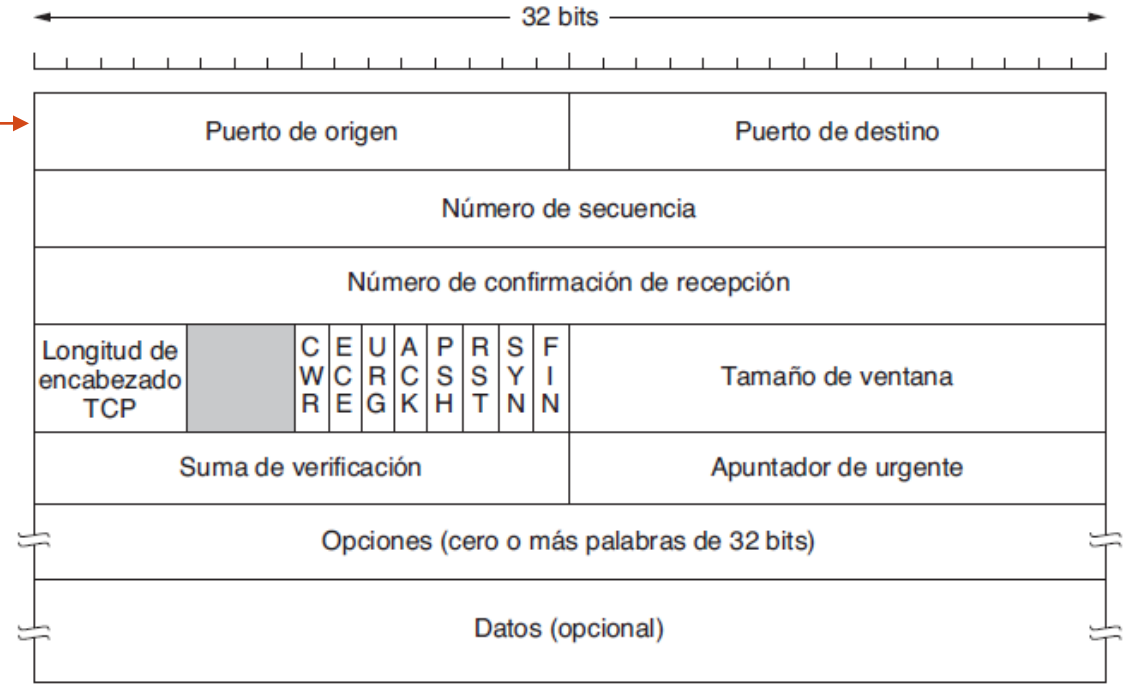


Figura 6-36. El encabezado TCP.

TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN

- Las implementaciones modernas de TCP realizan el descubrimiento de MTU de la ruta mediante mensajes de error de ICMP con el fin de encontrar la ruta más pequeña

- 5 tupla: protocolo (TCP), dirección IP destino y origen, puerto destino y origen.
- Acknowledgement: confirmación de recepción
- *CWR y ECE* para indicar congestión
- *SYN* se usa para establecer conexiones.
- *RST* se utiliza para reestablecer la conexión.
- *FIN* libera la conexión.

```
▼ Transmission Control Protocol, Src Port: 57488, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 57488
    Destination Port: 80
    [Stream index: 16]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
    Window size value: 29200
    [Calculated window size: 29200]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0x5a78 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ [SEQ/ACK analysis]
```

TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN

- El control de flujo en TCP se maneja mediante una ventana deslizante de tamaño variable.
  - El campo *Tamaño de ventana* indica la cantidad de bytes que se pueden enviar.
  - Un campo de Tamaño de ventana de 0 es válido e indica que se han recibido los bytes hasta *Número de confirmación de recepción*
  - *-1* para cuando el receptor no ha tenido oportunidad de consumir los y ya no desea más.

- *Suma de verificación* para agregar confiabilidad.
- *Opciones* agrega las características adicionales que no están cubiertas por el encabezado normal.
  - MSS (Tamaño Máximo de Segmento) que un host permite aceptar.
  - Tamaño de escala permite al emisor y al receptor negociar un factor de escala de ventana al inicio de la conexión.

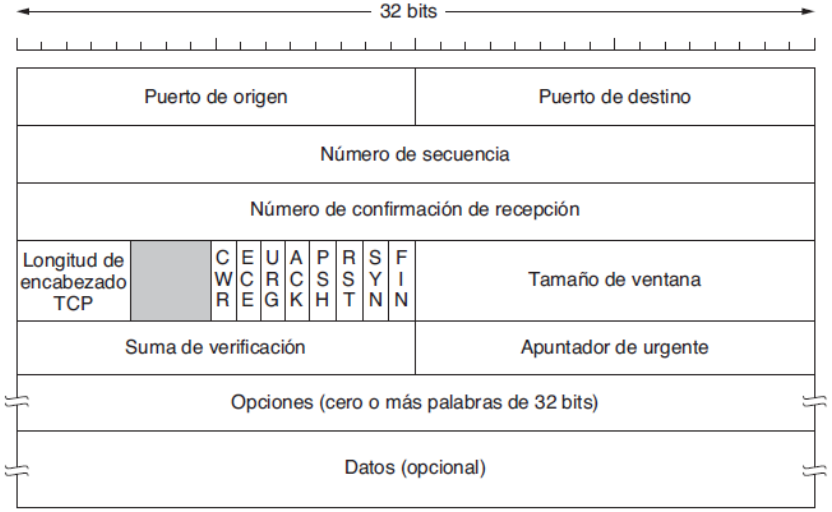
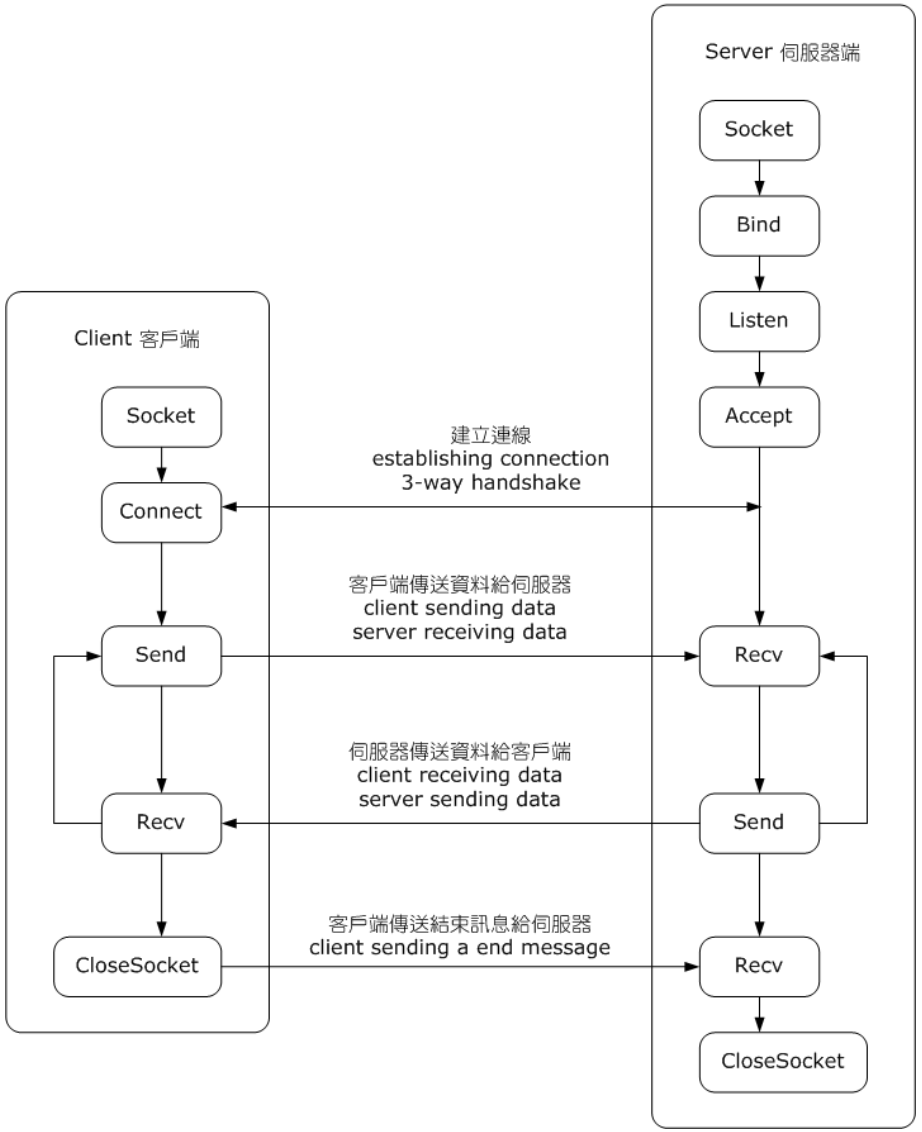


Figura 6-36. El encabezado TCP.

TCP Socket 基本流程圖
TCP Socket flow diagram



CONNECT

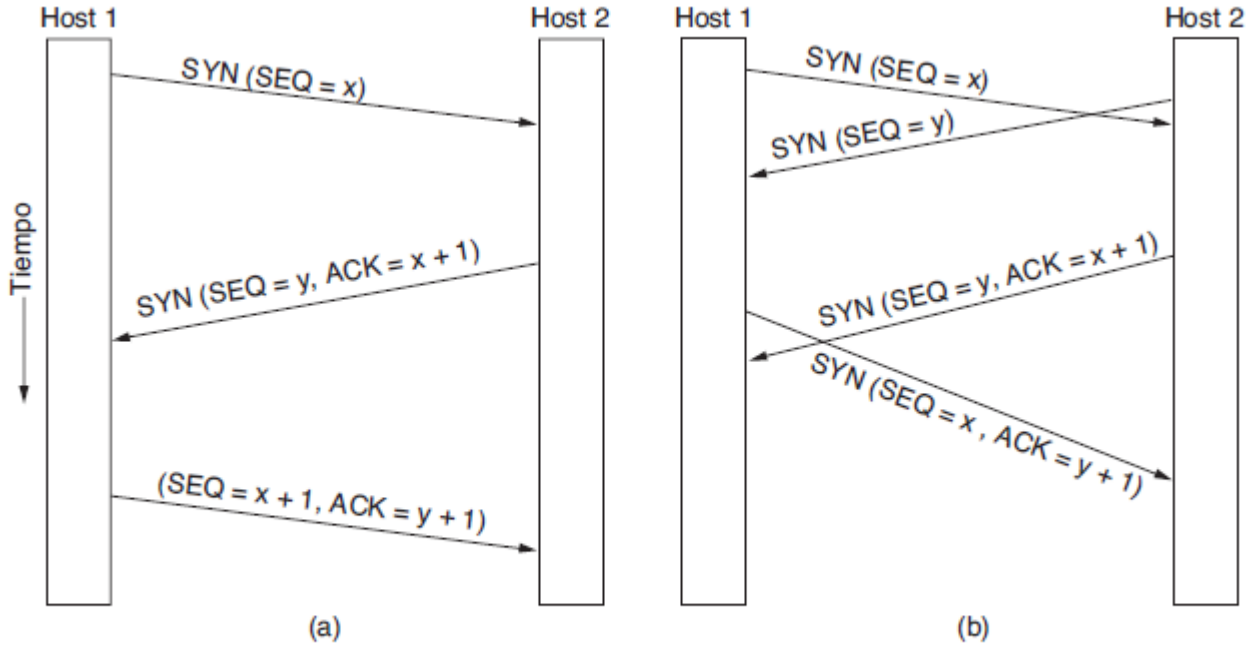


Figura 6-37. (a) Establecimiento de una conexión TCP en el caso normal. (b) Establecimiento de una conexión simultánea en ambos lados.

LECTURAS

Material utilizado	1. Arboleda, L. (2012). Programación en Red con Java. 2. Harold, E. (2004). Java network programming. " O'Reilly Media, Inc.". 3. Tanenbaum, A. S. (2003). Redes de computadoras. Pearson educación. 4. Reese, R. M. (2015). Learning Network Programming with Java. Packt Publishing Ltd.
Actividades DESPUÉS clase	A1. A1. Leer del libro 3 el contenido desde la sección 6.5.1 hasta la 6.5.7

REFERENCIAS

1. https://www.google.com.co/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwju5o62x-vdAhXjtlkKHaouDcAQjRx6BAgBEAU&url=http%3A%2F%2Feltallerdelbit.com%2Fdireccionamiento-ip%2F&psig=AOvVaw3E_T5IpV-ANtL0eEQbHtkg&ust=1538700259199893
2. https://www.cisco.com/c/dam/en/us/support/docs/ip/dynamic-address-allocation-resolution/19580-dhcp-multintwk-4.gif
3. https://en.wikipedia.org/wiki/Berkeley_sockets#/media/File:InternetSocketBasicDiagram_zhtw.png
4. https://buildingautomationmonthly.com/what-is-the-tcp-ip-stack/