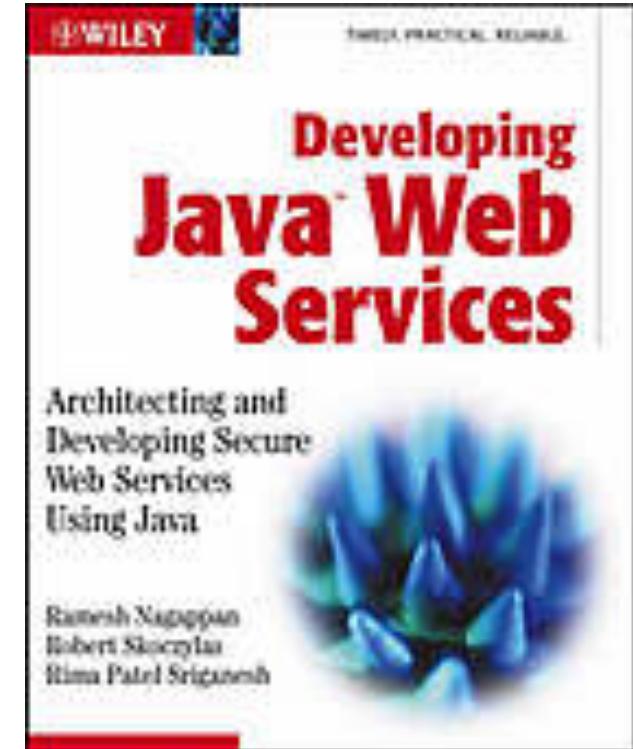
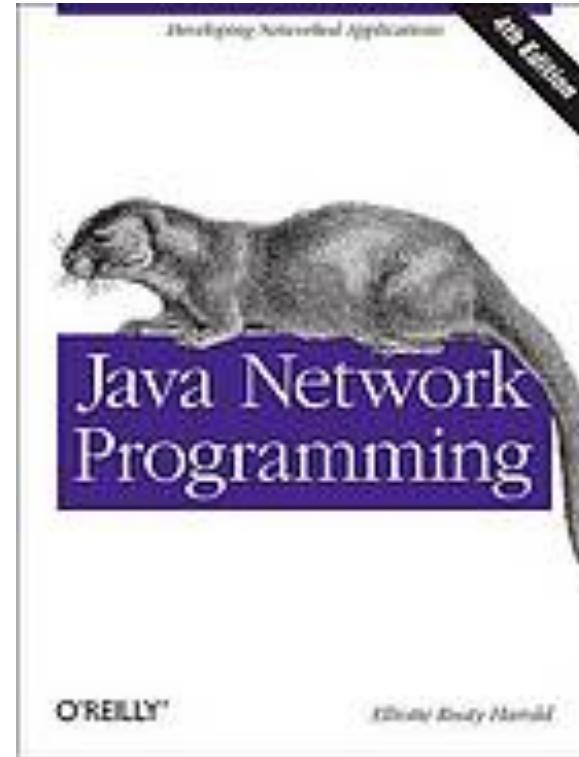
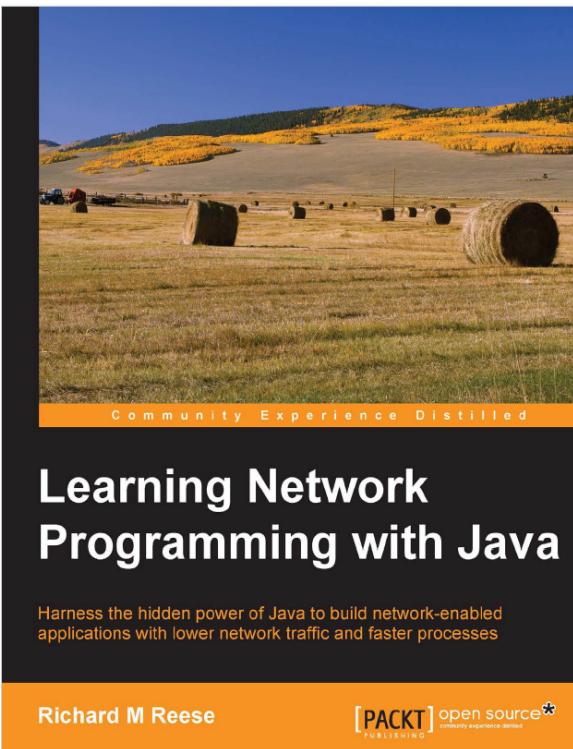
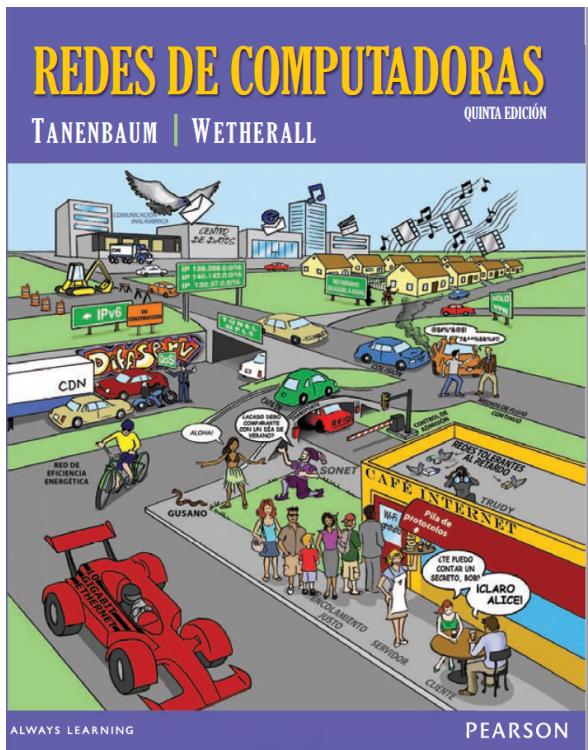


REDES DE COMPUTADORES Y LABORATORIO

Yor Jaggy Castaño, MSc



BIBLIOGRAFÍA



DE APLICACIÓN A TRANSPORTE

- ¿Que sabemos del modelo OSI y de la pila TCP/IP?
- ¿Que sabemos de la capa de aplicación?
- ¿Que sabemos sobre los servicios y protocolos de la capa de aplicación?
- ¿Esta claro el concepto de alta disponibilidad (HA: High Availability)?
- ¿Por que es importante HA?
- ¿Que no entendemos?

CONCEPTOS IMPORTANTES

- Modelo OSI
- TCP/IP
- Capa presentación y sesión en OSI
- Capa Aplicación en OSI y TCP/IP
- DNS
- EMAIL
- POP
- IMAP
- SMTP (MUA, MTA)

CONCEPTOS IMPORTANTES

- Modelo OSI
- TCP/IP
- Capa presentación y sesión en OSI
- Capa Aplicación en OSI y TCP/IP
- DNS
- EMAIL
- POP
- IMAP
- SMTP (MUA, MTA)
- WWW
- HTTP
- ARQUITECTURA CLIENTE/SERVIDOR
- CLIENTE
- SERVIDOR
- PUERTOS (Well Known)
- SOCKETS
- IETF
- IANA
- RFC

TIPOS DE ARQUITECTURA CLIENTE/SERVIDOR

Arquitectura de dos niveles

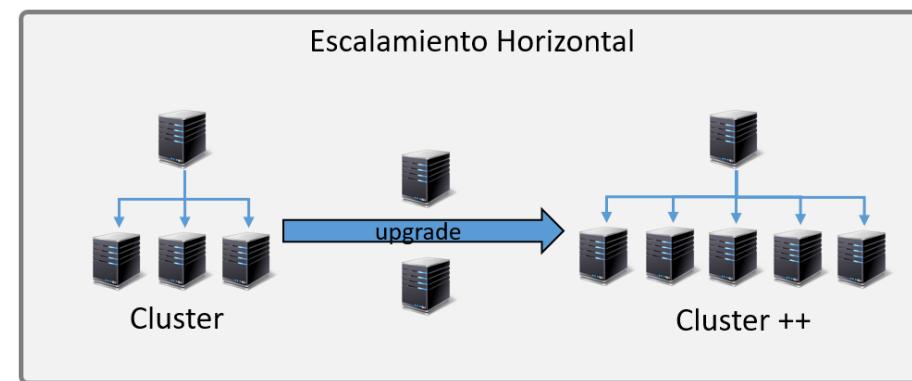
- Código cliente de la aplicación
- Servidor de base de datos

Arquitectura de tres niveles

- La capa de presentación
- La capa de funcionalidad
- La capa de lógica de los datos

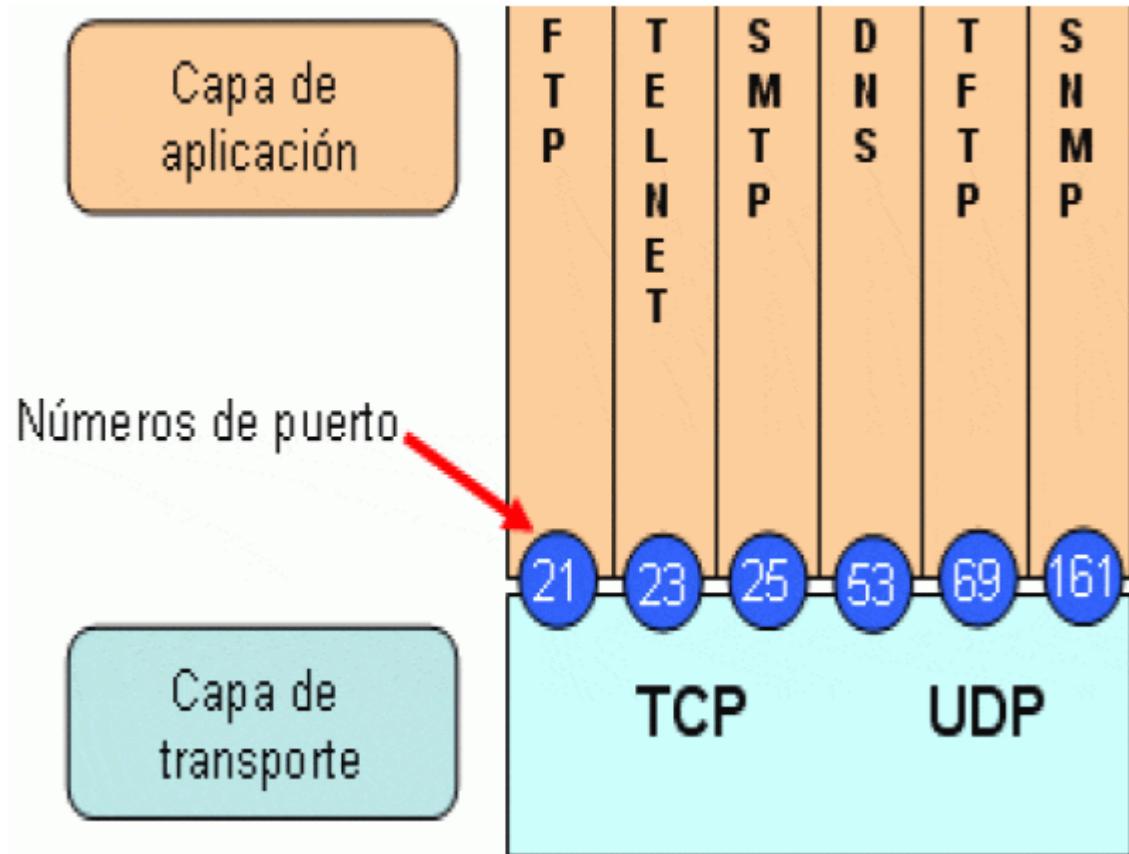
Arquitectura de N-Niveles

- Escalar vertical / horizontal



PUERTO

- Número asignado a un proceso, siempre y cuando el proceso solicite escuchar algún tipo de tráfico mediante **Sockets**
- Cantidad de puertos 65.536, de este total hay 1024 reservados



SOCKET

Una aplicación crea un socket al especificar tres elementos:

- Dirección IP del equipo.
- El puerto que la aplicación esté utilizando.
- El tipo de servicio (UDP o TCP) utilizado por la aplicación.

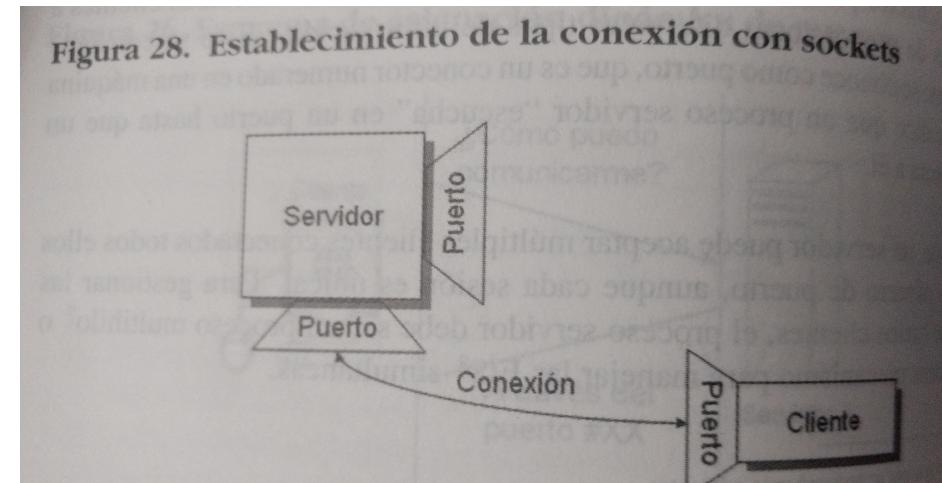
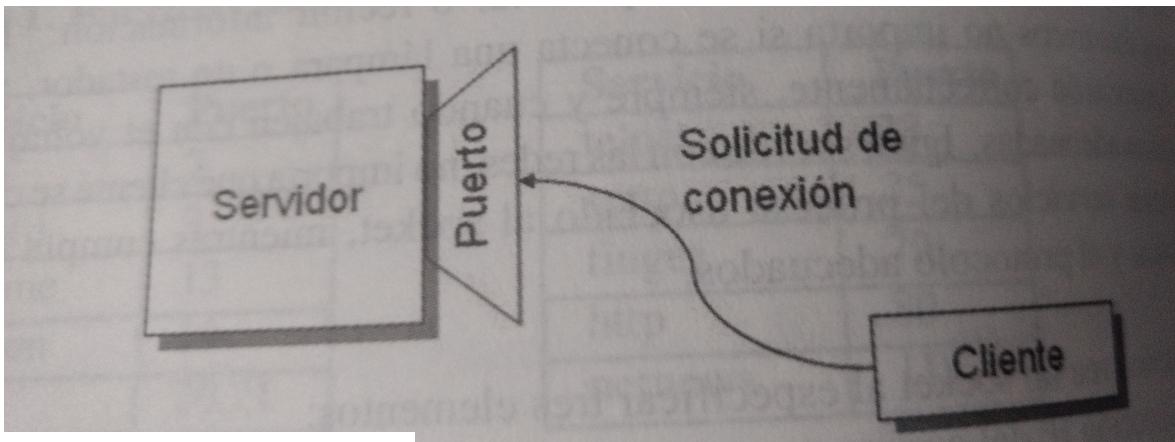


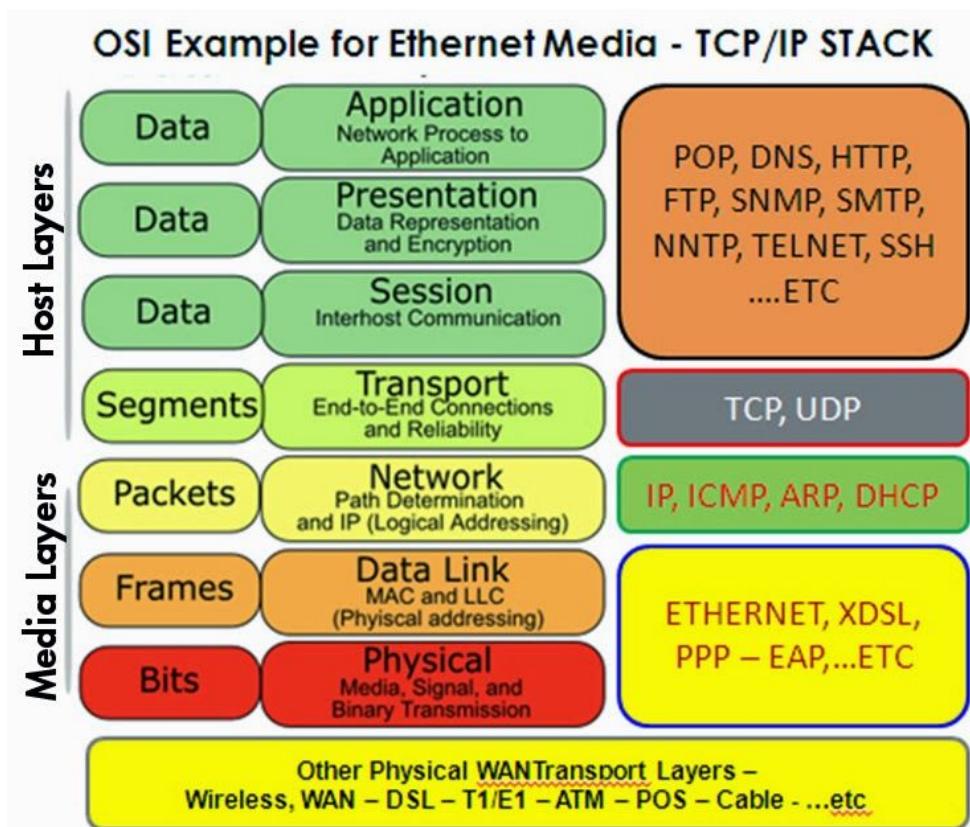
Figura 28. Establecimiento de la conexión con sockets

OTROS CONCEPTOS

- Nslookup
- Ping
- Traceroute
- Webhook Relay (tunneling)
- Docker
- Jmeter (Testing tools)
- Postman/Postwoman (HTTP Request)
- Whois § Namechecker
- Microservicios
- Python y Flask
- Balanceadores de carga (nginx)
- Proxy y proxy reverso
- DynDNS
- ORM en bases de datos
- Cloud Providers
- PaaS, IaaS, CaaS
- Packet Tracer

COMPETENCIAS

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



- *De ahora en adelante entendamos **la capa de transporte** como los vehículos de comunicación*
- *Mientras que **la capa de red** es la autopista en la que transitan esos vehículos.*

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

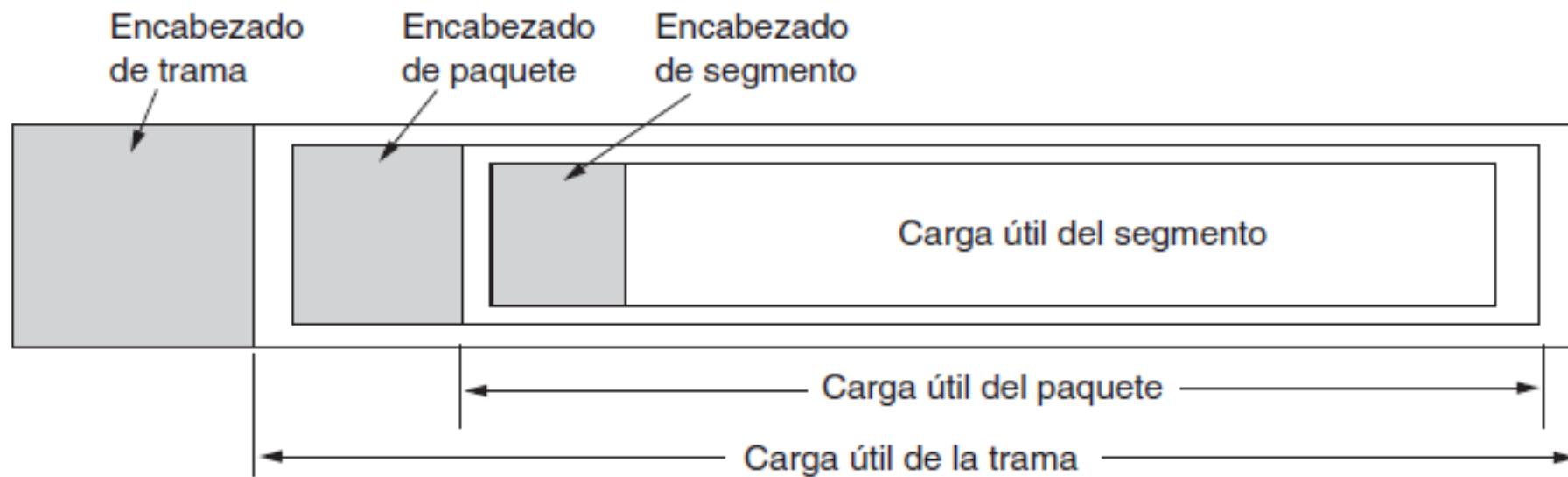


Figura 6-3. Anidamiento de segmentos, paquetes y tramas.

| 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 |

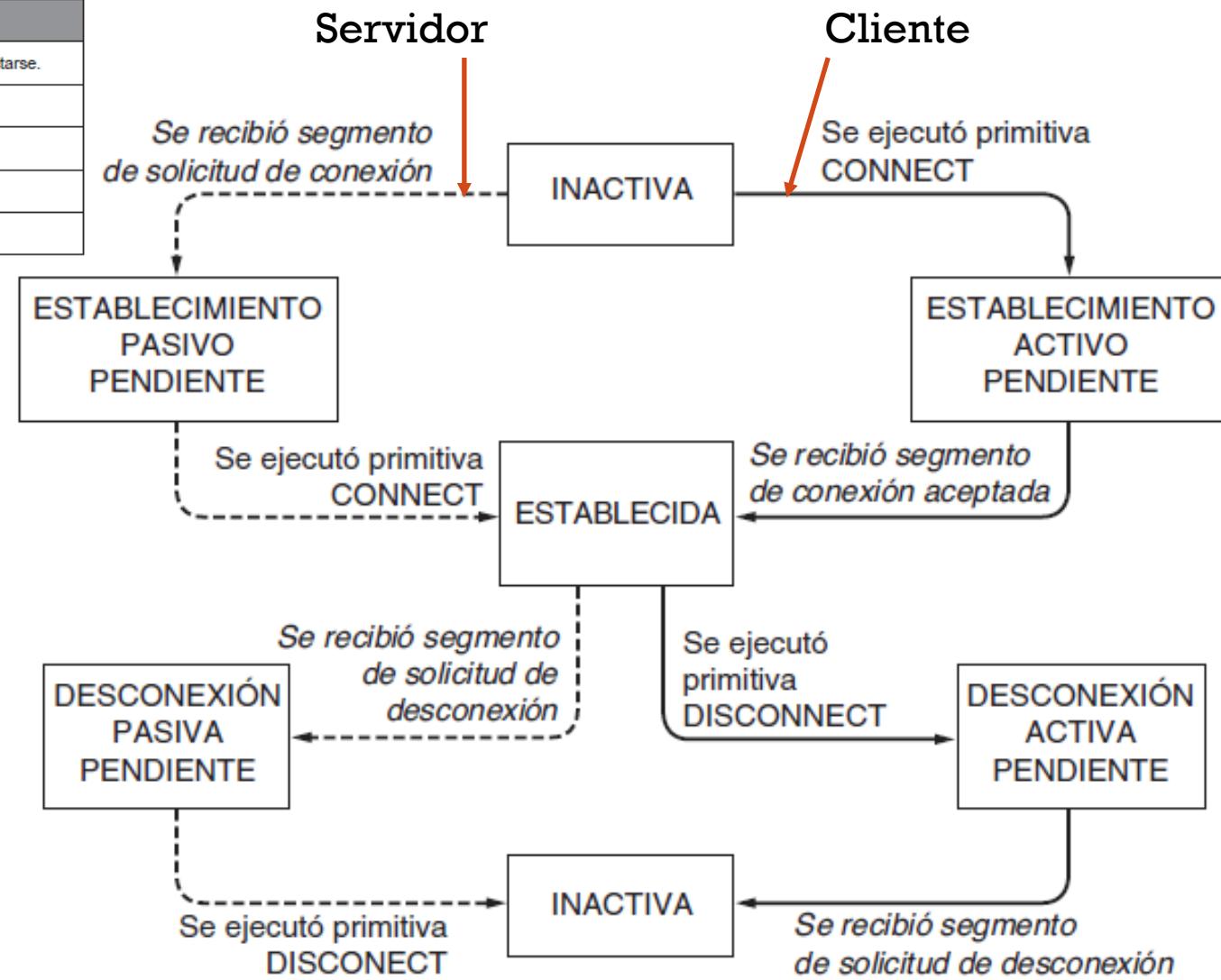


Figura 6-4. Un diagrama de estado para un esquema simple de manejo de conexiones. Las transiciones etiquetadas en cursiva se producen debido a la llegada de paquetes. Las líneas continuas muestran la secuencia de estados del cliente. Las líneas punteadas muestran la secuencia de estados del servidor.

SOCKETS DE BERKLEY

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

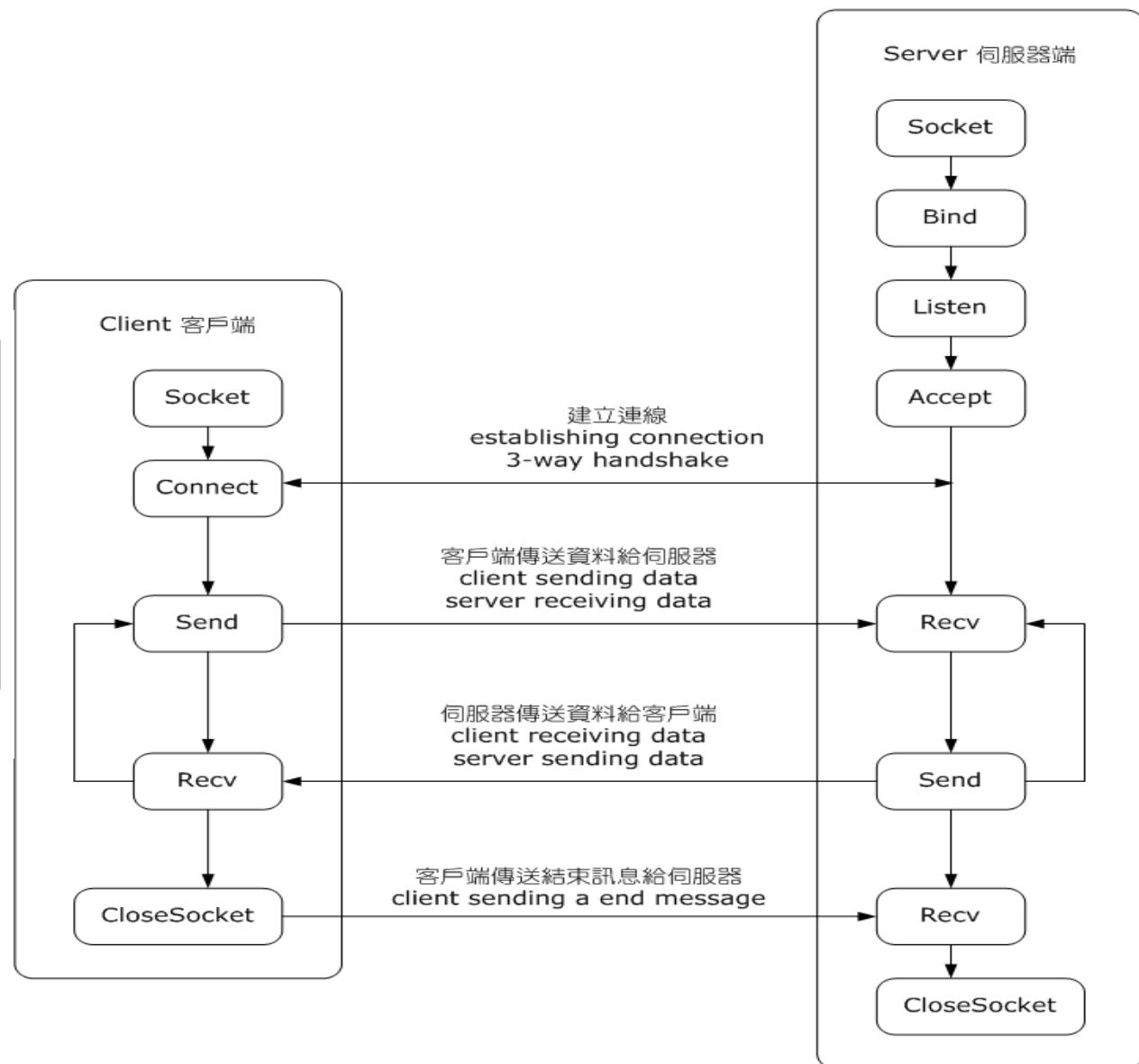
| Primitiva | Significado |
|-----------|--|
| SOCKET | Crea un nuevo punto terminal de comunicación. |
| BIND | Asocia una dirección local con un socket. |
| LISTEN | Anuncia la disposición de aceptar conexiones; indica el tamaño de la cola. |
| ACCEPT | Establece en forma pasiva una conexión entrante. |
| 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.

TCP Socket 基本流程圖
TCP Socket flow diagram

| 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. |

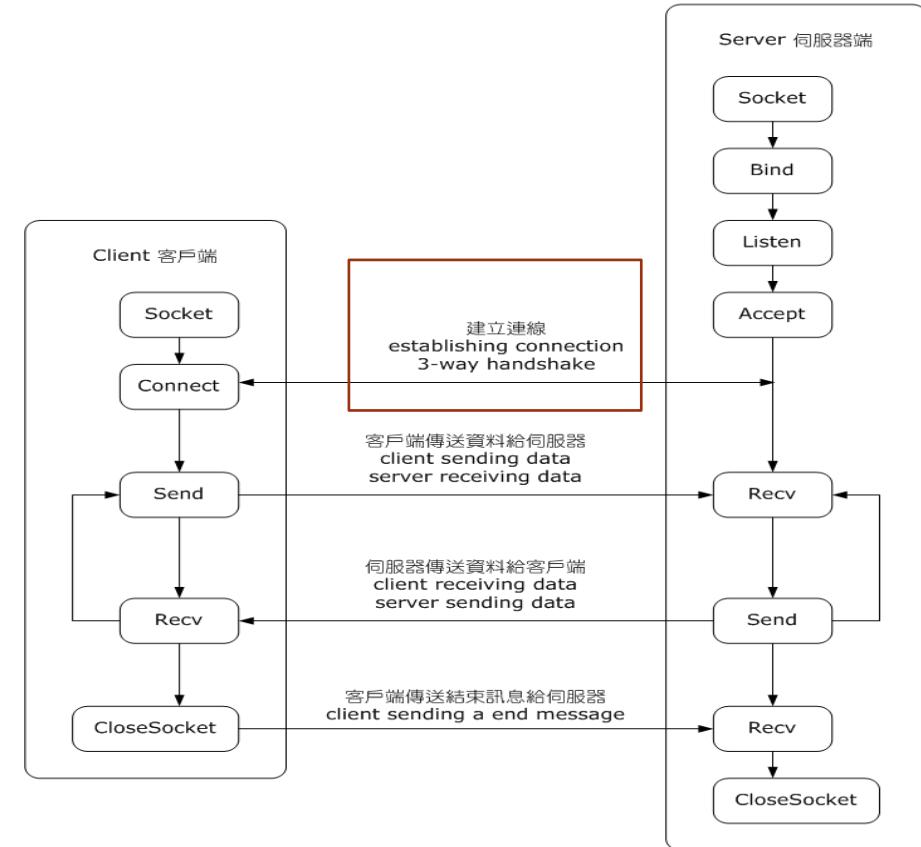
Figura 6-5. Las primitivas de socket para TCP.



DIRECCIONAMIENTO Y ESTABLECIMIENTO DE CONEXIÓN

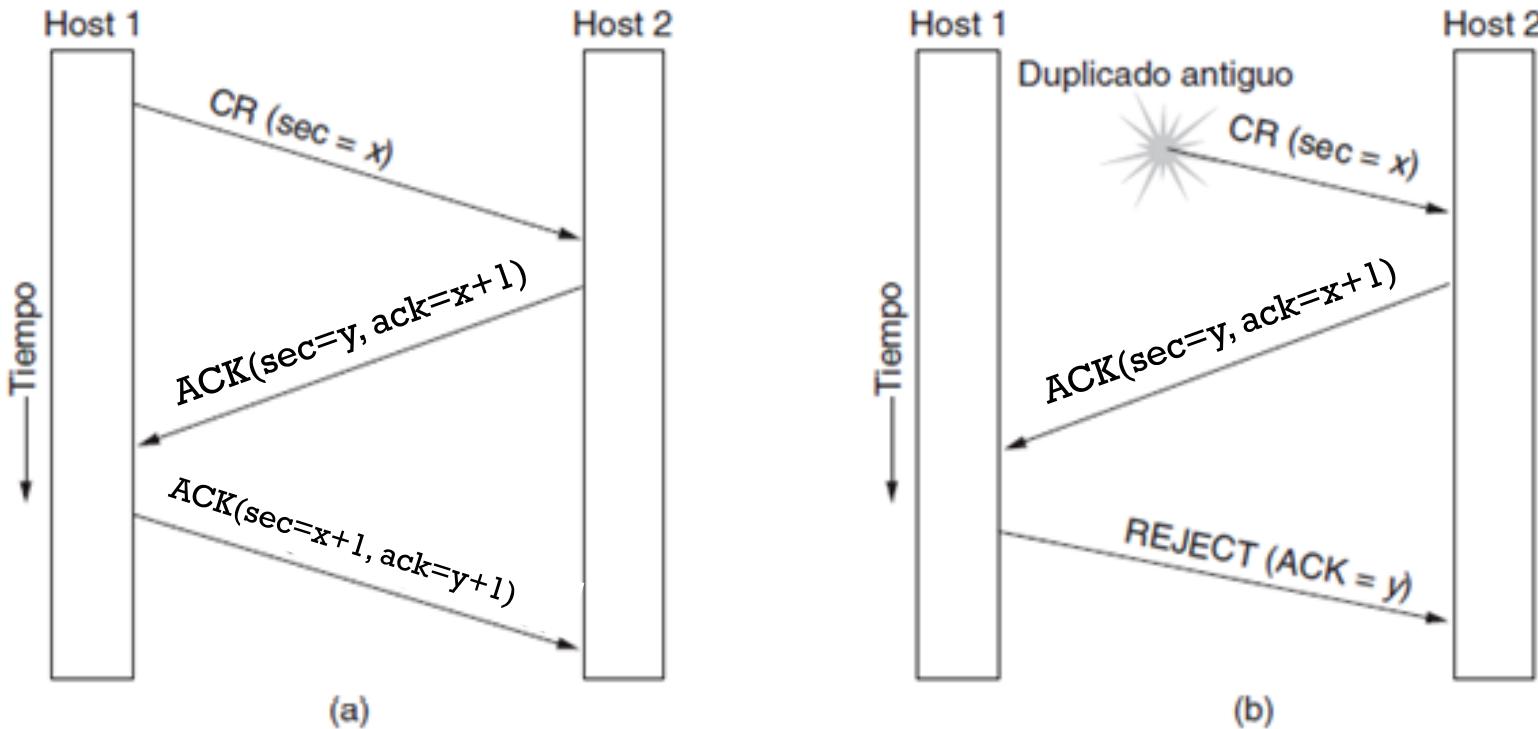
- El método que permite definir las direcciones de transporte en las que los procesos pueden escuchar solicitudes de conexión es a través de puntos terminales conocidos como **puertos**.
- **Acuerdo de tres vías (three-way handshake)**. Es un protocolo que implica que un igual (receptor) verifique que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

TCP Socket 基本流程圖
TCP Socket flow diagram



SEGMENTOS

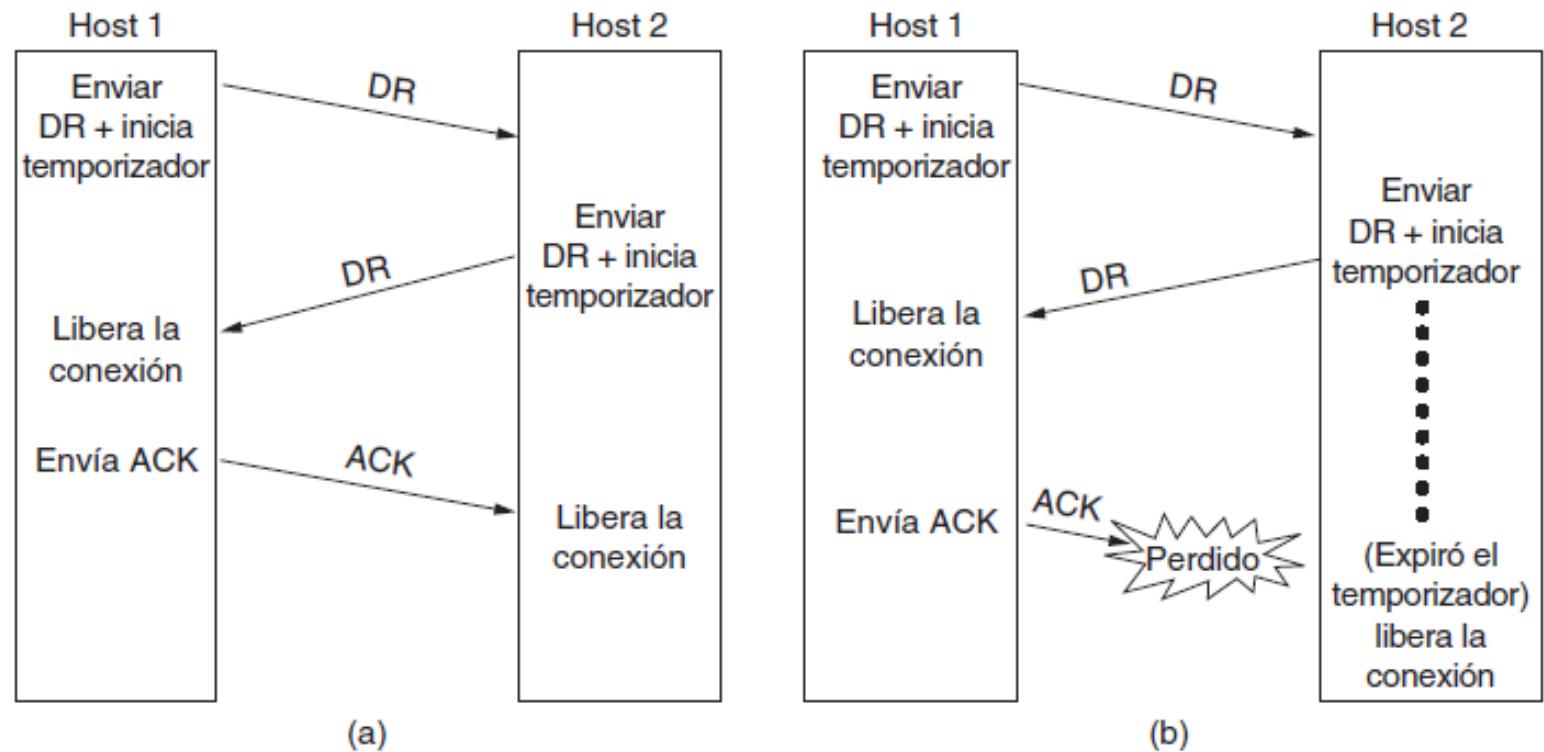
- CR (CONNECTION REQUEST)
- DR (DISCONNECTION REQUEST)
- ACK (acknowledgement), es un mensaje que el destino de la comunicación envía al origen de esta para confirmar la recepción del mensaje.



Establecimiento de una conexión

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

- *De ahora en adelante entendamos **la capa de transporte** como los vehículos de comunicación*
- *Mientras que **la capa de red** es la autopista en la que transitan esos vehículos.*

QUE TIPO VEHÍCULOS TENEMOS?

- Un vehículo UDP:
 - Al momento de transportar la información **no establece una conexión** de inicio a fin, no asegura llegar al destino.
- Un vehículo TCP:
 - **Establece una conexión** de inicio a fin, sabe en cada momento por que lugares debería pasar, asegura que si sigue esos lugares* **llegara al destino**.

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 transmite segmentos que consisten en un encabezado de 8 bytes seguido de la carga útil.

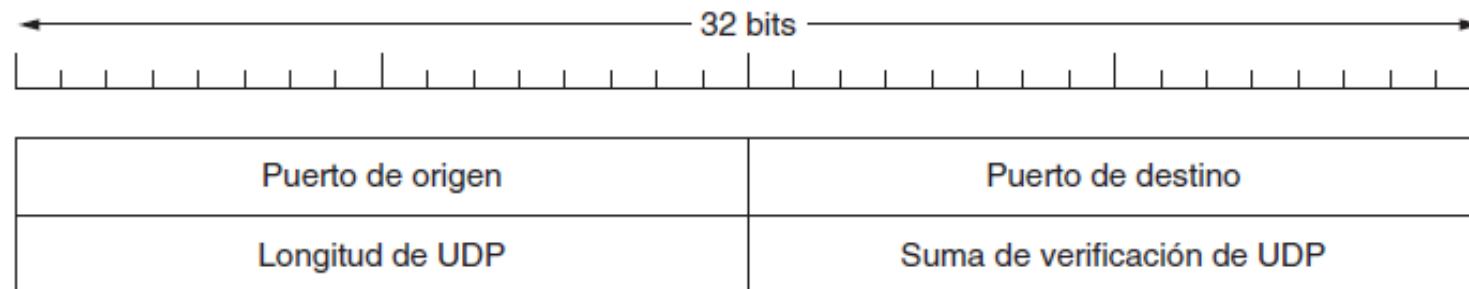


Figura 6-27. El encabezado UDP.

- La longitud incluye el encabezado de 8 bytes y los datos. La longitud mínima es de 8 bytes y la máxima es de 65535 bytes.

UDP (USER DATAGRAM PROTOCOL)

- Es orientado a las transacciones, no asegura entrega ni protección duplicada.
- ¿Es útil un vehículo que no asegura entregas, ni el correcto envío de la información?

UDP (USER DATAGRAM PROTOCOL)

- Es orientado a las transacciones, no asegura entrega ni protección duplicada.
- ¿Es útil un vehículo que no asegura entregas, ni el correcto envío de la información?
 - DNS
 - TFTP
 - Multimedia*
 - RTP (Realtime transport protocol)

LOS PROTOCOLOS DE TRANSPORTE DE INTERNET: TCP

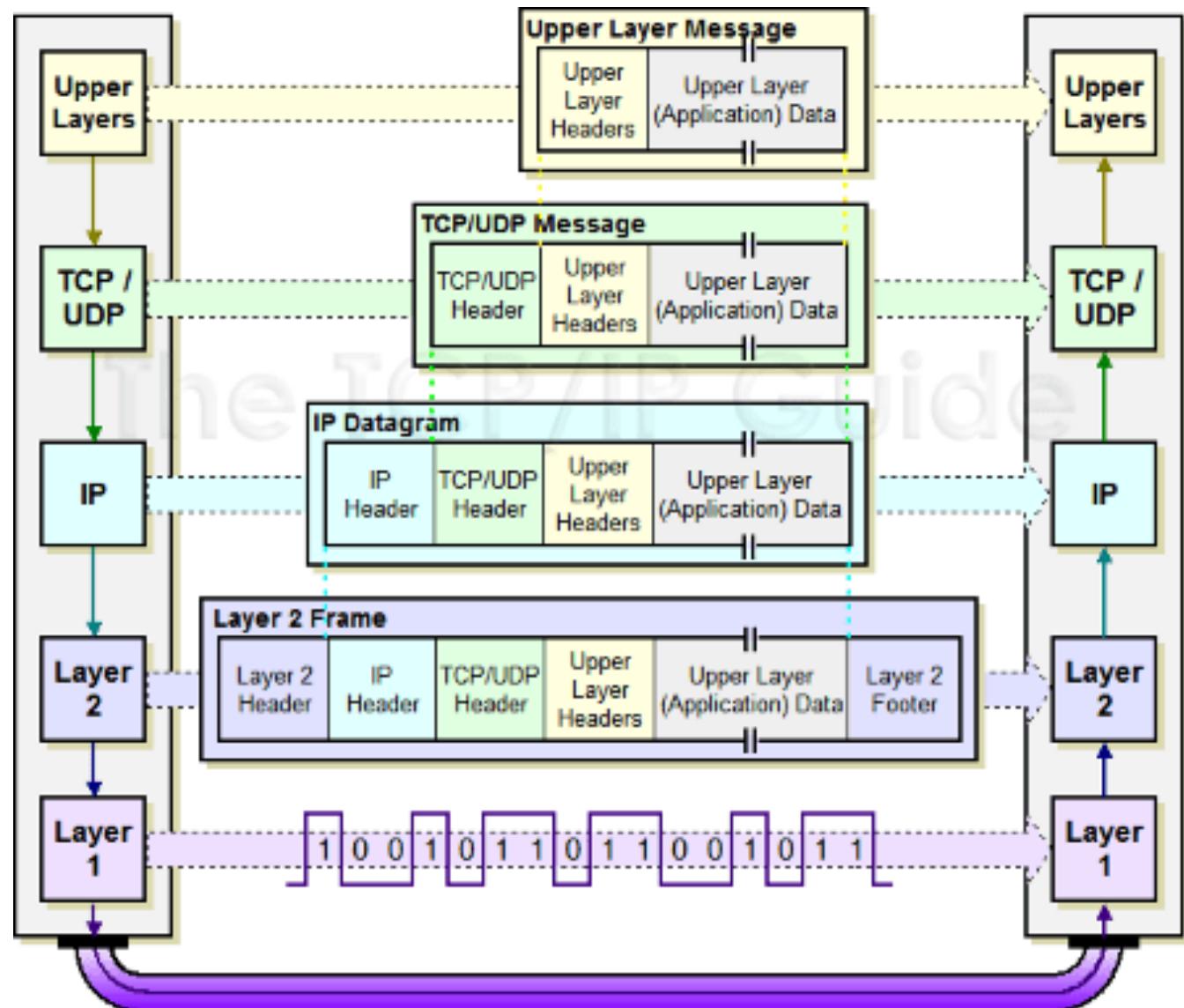
TCP (PROTOCOLO DE CONTROL DE TRANSMISIÓN)

- Se diseño con la finalidad de proporcionar un flujo de bytes confiable de extremo a extremo a través de una **internet 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 | SMTMP | 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.

Las capas inferiores no ofrecen 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.

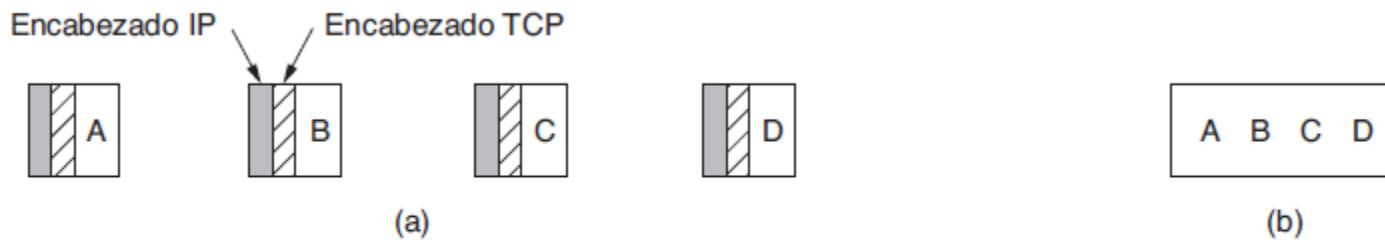


Figura 6-35. (a) Cuatro segmentos de 512 bytes que se envían como diagramas IP separados. (b) Los 2 048 bytes de datos que se entregan a la aplicación en una sola llamada READ.

- 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: 0xbff64 [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.

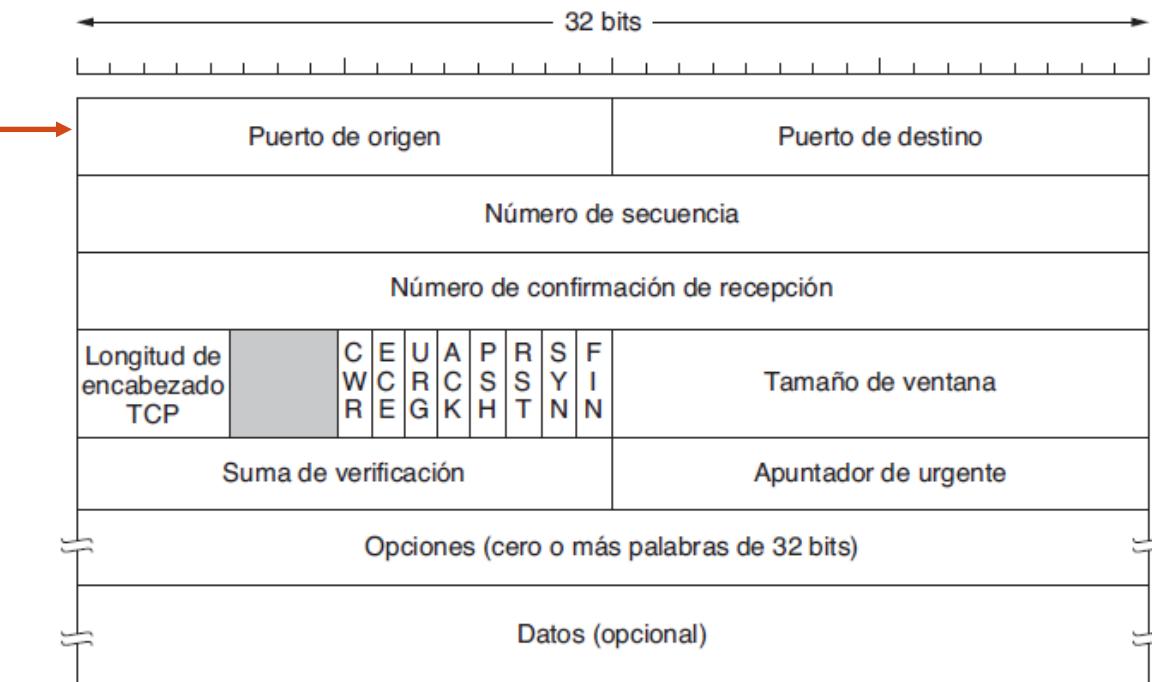


Figura 6-36. El encabezado TCP.

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 consumirlos 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.

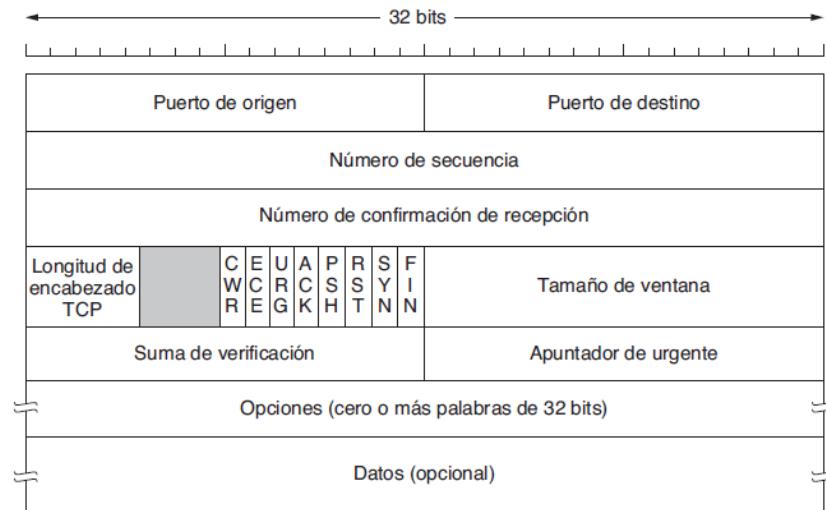


Figura 6-36. El encabezado TCP.

VULNERABILIDAD EN TCP:

¿Es claro el funcionamiento del mecanismo de acuerdo de 3 vias?

Usemos el:
“*¿y que tal si?*”

TCP SYN (STEALTH) SCAN (-SS)

- <https://nmap.org/book/synscan.html>

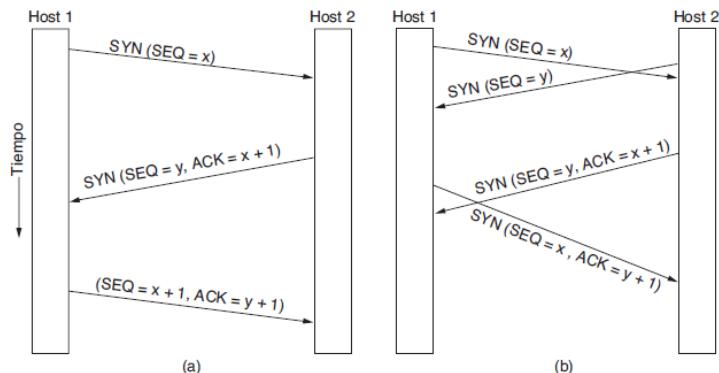
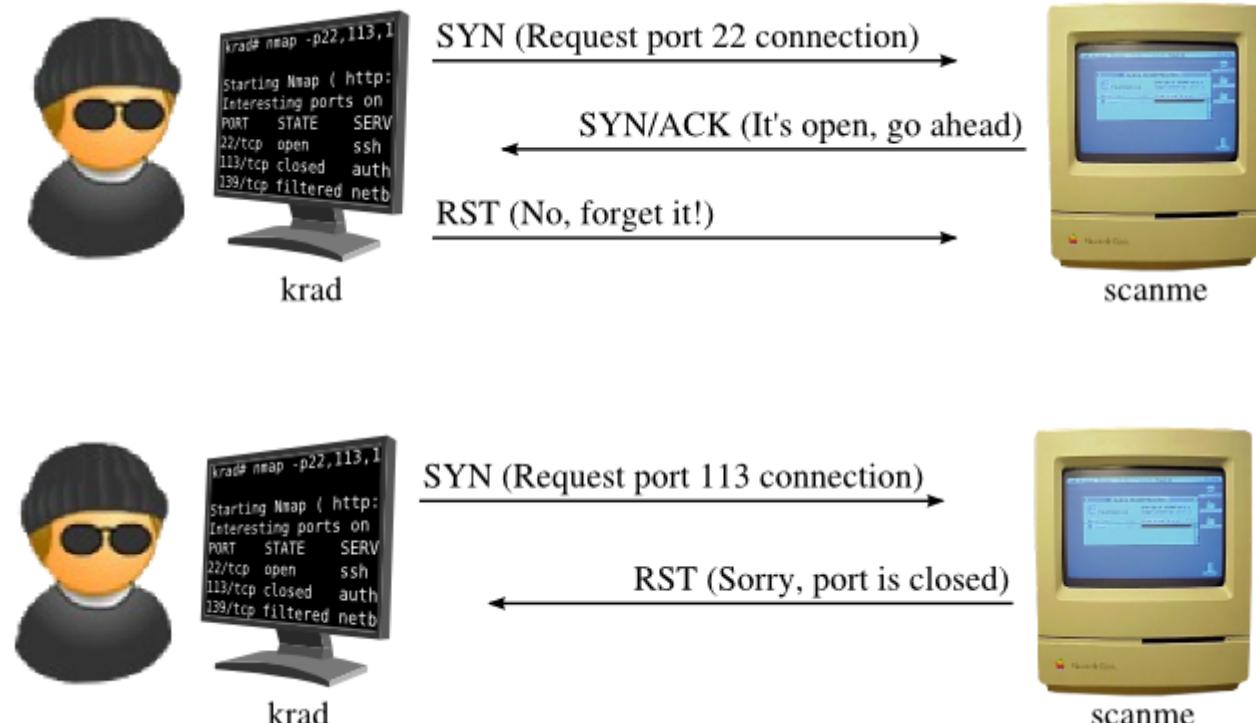


Figura 6-37. (a) Establecimiento de una conexión TCP en el caso normal. (b) Establecimiento de una conexión simultánea en ambos lados.



TCP SYN FLOOD

- Enviar repetidos paquetes SYN, buscando **bloquear** el servidor o dispositivo en modo escucha.
- Se llena los posibles **TCB (Transmission control block)** y se deja sin recursos para más posibles conexiones
- Como mitigamos esto?

TCP SYN FLOOD

- Enviar repetidos paquetes SYN, buscando **bloquear** el servidor o dispositivo en modo escucha.
- Como mitigamos esto?
 - **SYN Cookies** (hash en los paquetes con info del cliente, verificado por el server)
 - **RST Cookies** (Server responde con mal SYN/ACK y espera un RST)
 - **Stack Tweaking** (modifica el stack, tiempos de “resource allocation”, Timeout para bloquear conexiones entrantes)

CONTROL DE LA CONGESTIÓN Y CONTROL DE FLUJO

- El **control de flujo** se relaciona con el tráfico entre un emisor particular y un receptor particular. Su tarea es asegurar que un emisor rápido no pueda transmitir datos de manera continua a una velocidad mayor que la que puede absorber los paquetes el receptor.
- El **control de congestión** se ocupa de asegurar que la red sea capaz de transportar el tráfico ofrecido. Es un asunto global, en el que interviene el comportamiento de todos los hosts y enrutadores.

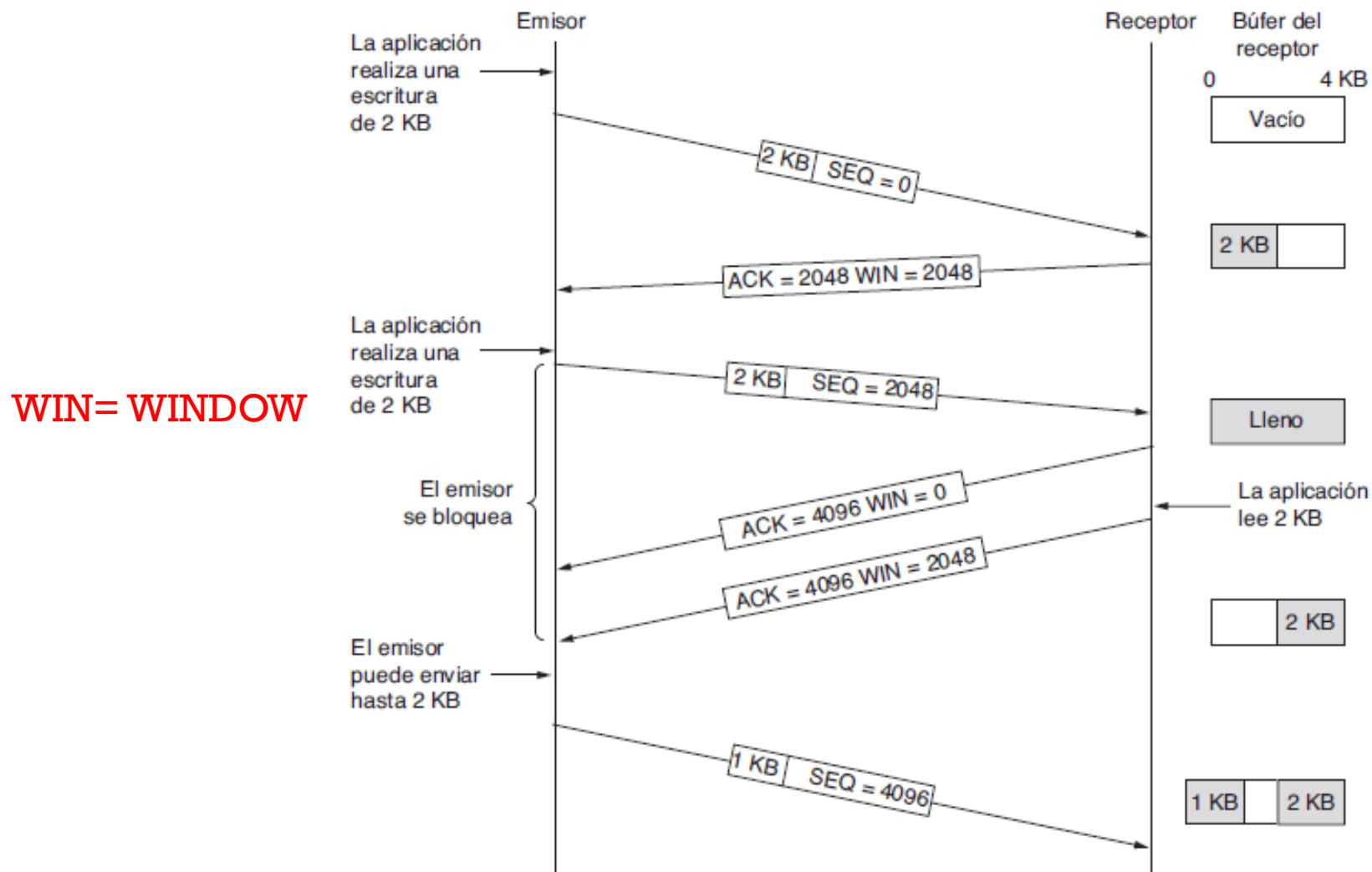


Figura 6-40. Administración de ventanas en TCP.

VENTANA DESLIZANTE DE TCP

- La ventana de 1 TCP separa los aspectos de la confirmación de la recepción correcta de los segmentos ($WIN=0$) y la asignación del búfer en el receptor ($WIN=...$).
- Cuando la ventana es 0, el emisor no puede enviar segmentos, excepto en dos situaciones:
 - Datos urgentes (URG) para finalizar el proceso en ejecución.
 - El emisor envía un segmento de 1 byte para que el receptor vuelva a anunciar el siguiente byte esperado y el tamaño de la ventana (**sonda de ventana**).

WIN= WINDOW

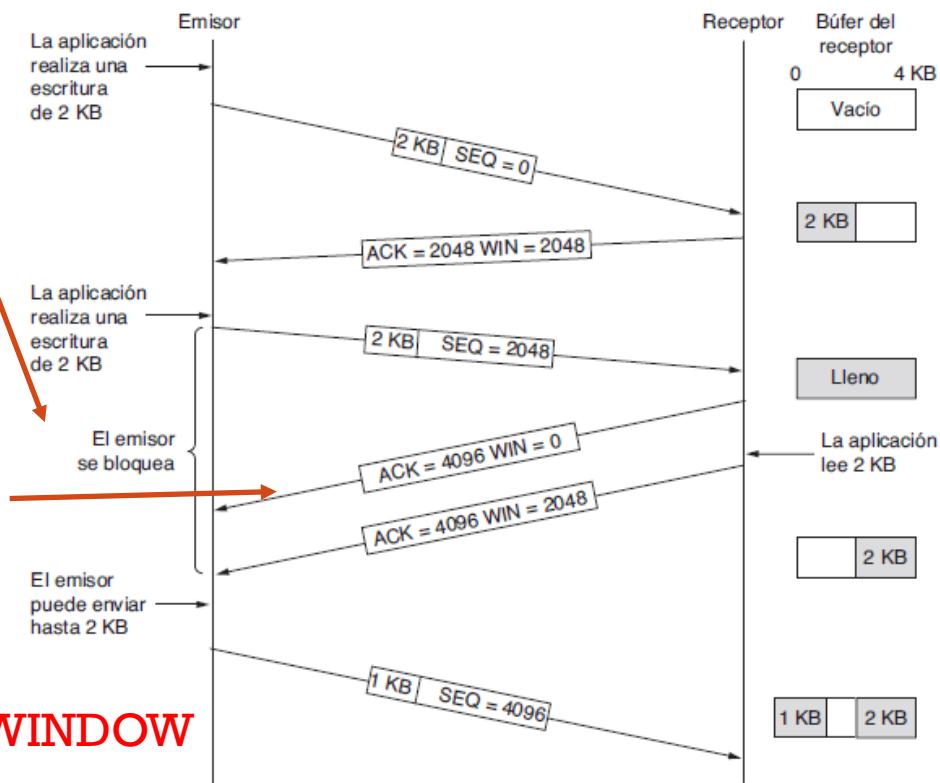
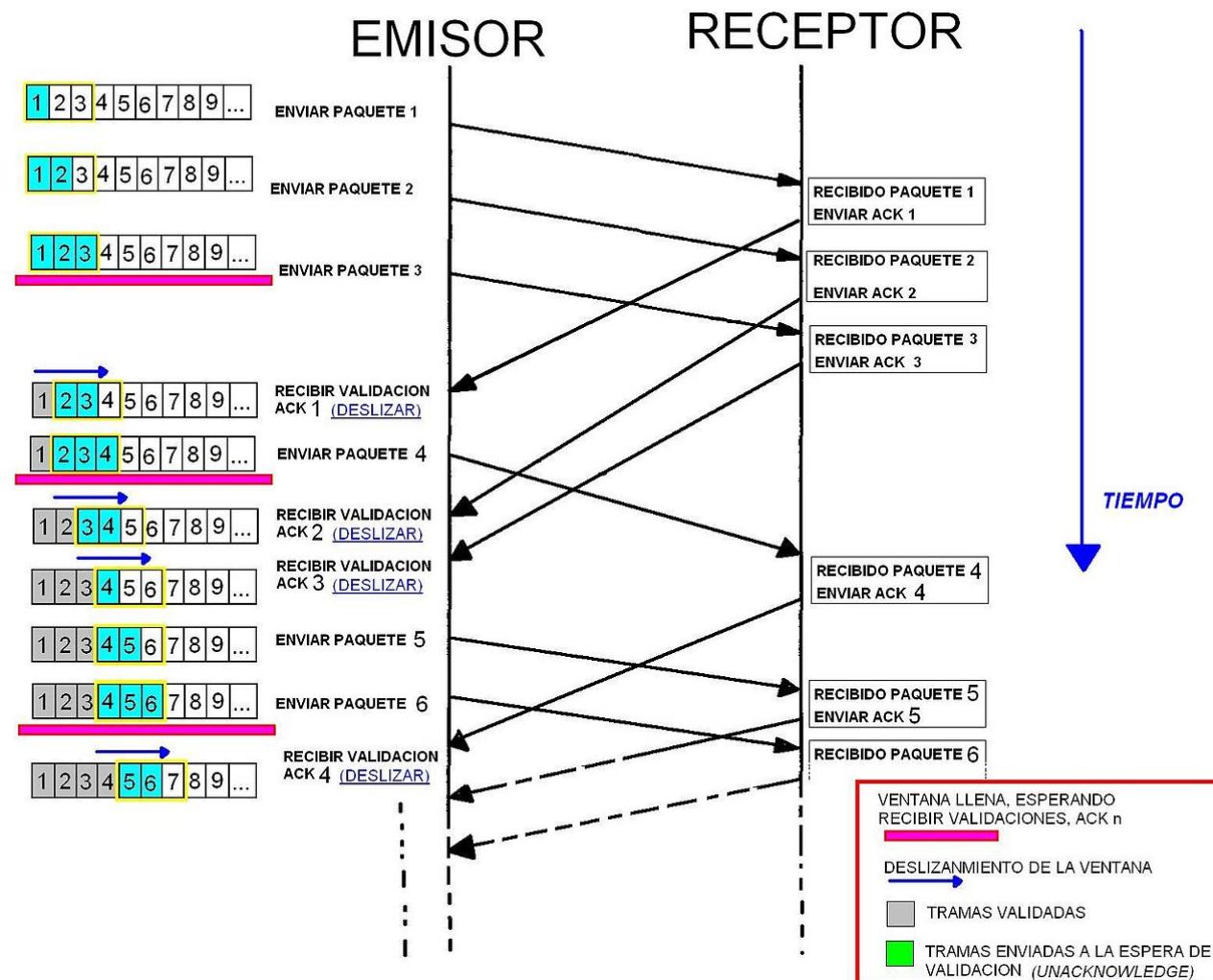


Figura 6-40. Administración de ventanas en TCP.

VENTANA DESLIZANTE DE TCP



VENTANA DESLIZANTE DE TCP

- Cuando el ancho de banda escasea y la aplicación que utiliza TCP aplica una suma de segmentos elevada (por ejemplo, Telnet y SSH), existe un mecanismo (**confirmaciones de recepción con retardo**) que permite retrasar las confirmaciones de recepción y las actualizaciones de ventana hasta 500 mseg, con el objetivo de reducir la carga impuesta en la red por el receptor. Pero este mecanismo es ineficiente para varios paquetes cortos (por ejemplo, datagramas de **21 bytes*** que contienen 1 byte de datos).

***21 bytes solo hablando del encabezado TCP y un byte de datos**

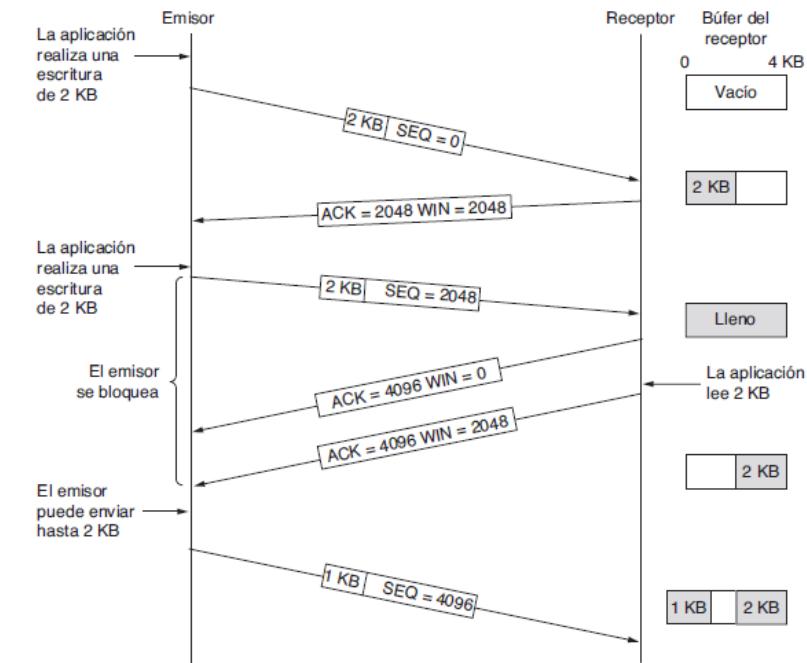
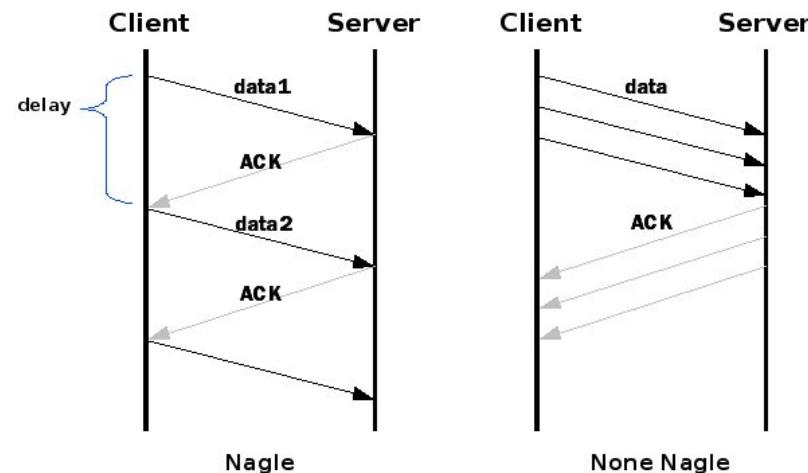


Figura 6-40. Administración de ventanas en TCP.

EL ALGORITMO DE NAGLE

- Cuando llegan datos en pequeñas piezas al emisor, sólo se envía la primera pieza y el resto se almacena en búfer hasta que se confirma la recepción del byte pendiente. Después se envían todos los datos del búfer en un segmento TCP y nuevamente comienzan a almacenarse en búfer los datos hasta que se haya confirmado la recepción del siguiente segmento. **Esto significa que sólo puede haber un paquete corto pendiente en cualquier momento dado.**
- **Es bueno deshabilitarlo cuando hay aplicaciones de gran flujo rápido de paquetes, por ejemplo, los juegos interactivos. Desactivar la opción TCP_NODELAY**

<https://youtu.be/fS3PW5yOttk>



SÍNDROME DE VENTANA TONTA

- Este síndrome ocurre cuando se pasan datos a la entidad TCP **emisora en bloques grandes**, pero una aplicación interactiva del lado **receptor lee datos sólo a razón de 1 byte a la vez**.
- La solución de Clark** es evitar que el receptor envíe una actualización de ventana para 1 byte. En cambio, se le obliga a esperar hasta tener disponible una cantidad decente* de espacio, y luego lo anuncia. El receptor no envía una actualización de ventana sino hasta que pueda manejar el tamaño máximo de segmento (MSS) que anuncio al establecerse la conexión.

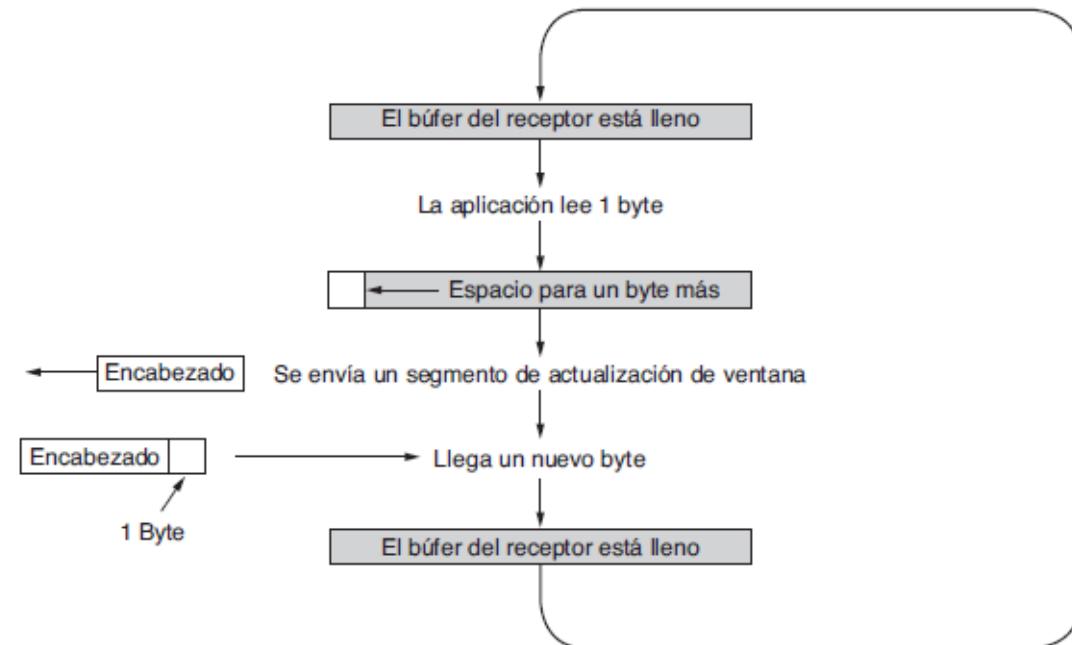


Figura 6-41. Síndrome de ventana tonta.

CLARK Y NAGLE

El algoritmo de Nagle y la solución de Clark al síndrome de ventana tonta son complementarios. **Nagle trataba de resolver el problema causado por la aplicación emisora que entregaba datos a TCP, 1 byte a la vez.** Clark trataba de resolver el problema de que la aplicación receptora tomara los datos de TCP, 1 byte a la vez. Ambas soluciones son válidas y pueden operar juntas. **El objetivo es que el emisor no envíe segmentos pequeños y que el receptor no los pida.**

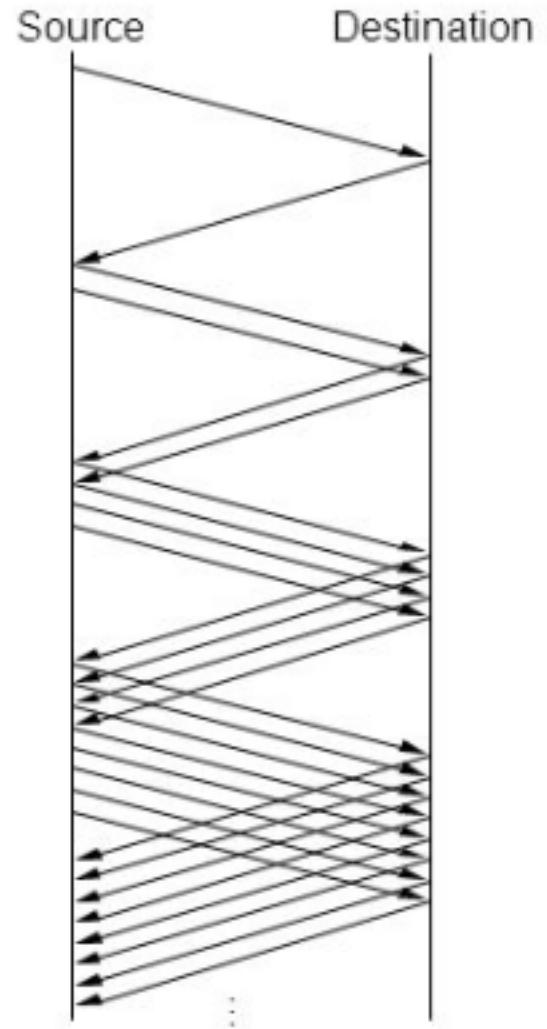
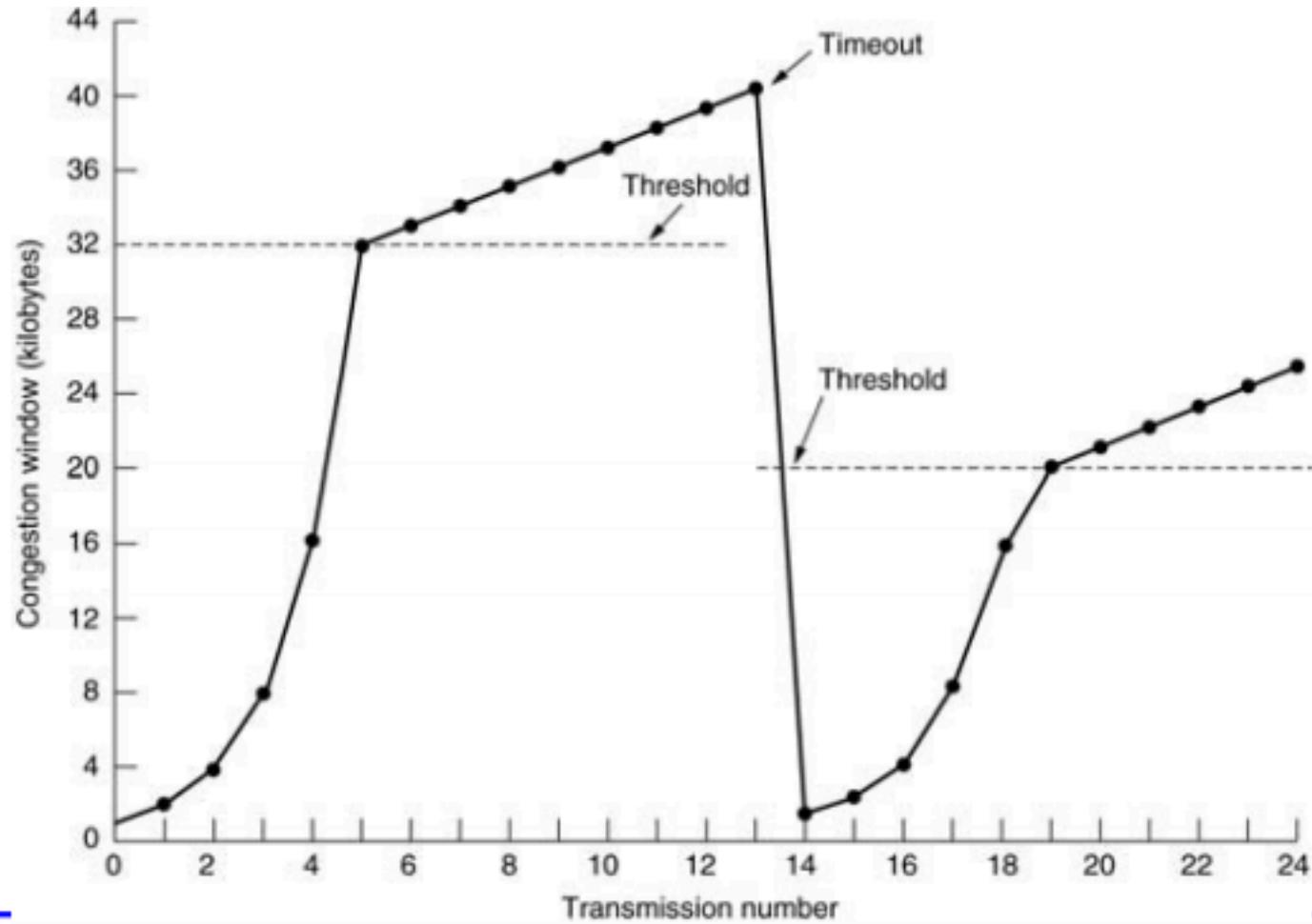
CONTROL DE CONGESTIÓN:

TCP Tahoe (Arranque lento):

- Crecimiento exponencial de la ventana de congestión
- Se recompensa cada ACK exitoso incrementando la ventana de congestión
- Si hay 3 ACK's duplicados, ventana de congestión baja a la mitad y luego la misma ventana crece linealmente

TCP Reno (Recuperación rápida):

- Si llegan ACK's duplicados (paquete perdido) significan que los segmentos enviados después al perdido también se perdieron
 - Se regula parcialmente el envío de información hasta que dejen de llegar ACK's repetidos.
-
- * Se suelen implementar juntos (**Arranque lento, retransmisión rápida y recuperación rápida**)



EL FUTURO DE TCP

- Existen distintas versiones de los algoritmos clásicos, en especial para el control de la congestión y la robustez ante los ataques informáticos. Dos aspectos a tener en cuenta:
 - TCP no proporciona la semántica de transporte que desean todas las aplicaciones, por ejemplo el control de los límites en los mensajes o registros.
 - SCTP (Protocolo de Control de Transmisión de Flujo, *Stream Control Transmission Protocol*). [RFC 4960](#)
 - SST (Transporte Estructurado de Flujo, *Structured Transmision Protocol*).
 - Control de la congestión. TCP se basa en las pérdidas de paquetes como señal de congestión.

DEBERES:

| | |
|---|--|
| <p>Proxima clase Miercoles 4 de Marzo</p> <p>Entrega 2 proyecto de desarrollo e investigación</p> | <p>Parcial 1</p> <p>Semana 8: viernes 13</p> |
|---|--|