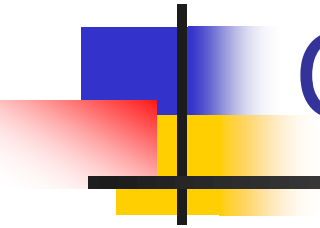
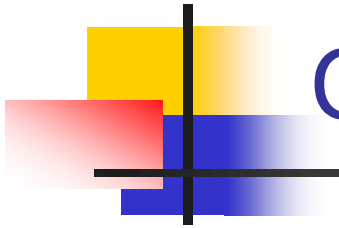


Capítulo 6

La capa de transporte CT



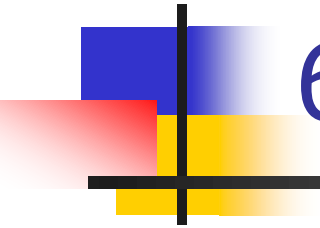
Contenido

6.1 El servicio de transporte

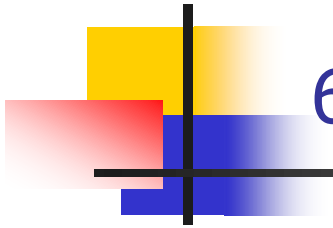
6.2 Elementos de los protocolos de transporte

6.3 El protocolo de Internet UDP

6.4 El protocolo de Internet TCP

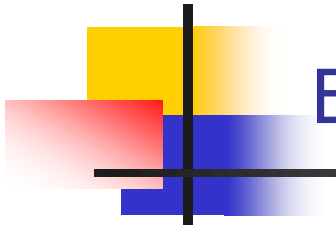


6.1 El servicio de transporte



6.1.1 Servicios proporcionados a la capa superior

- Los desarrolladores de aplicaciones de red necesitan conocer cómo funciona la CT
- La CT da un transporte de datos confiable **extremo a extremo** independientemente de las redes físicas
- Los usuarios de los servicios de la CT son los procesos de la capa de aplicación
- Para esto, la CT recibe los servicios de la capa de red CR



Entidad de transporte

- La CT se implementa en hardware y en software
- La CT puede estar en:
  - El kernel del sistema operativo
  - Un proceso de usuario independiente
  - Una librería de las aplicaciones de red, o
  - En la tarjeta de red

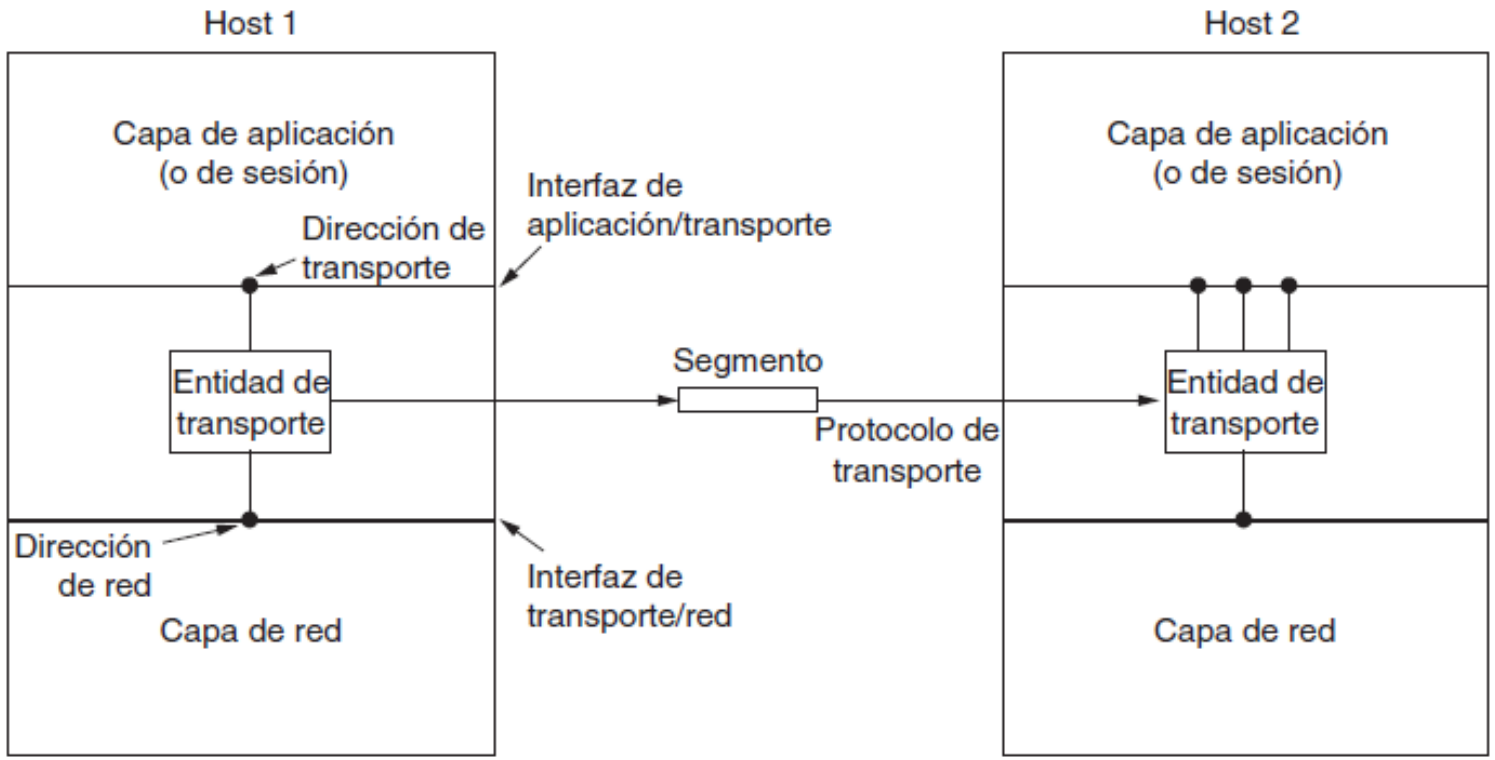


Figura 6-1. Las capas de red, transporte y aplicación.

Tipos de servicio de la CT

- Similar a la CR, hay dos tipos de servicio que se ofrecen a la capa de aplicación CA: orientado a la conexión, y sin conexión
- Recordar lo siguiente:
  - Las capas de Enlace de Datos, Capa de Red y Capa de Transporte ofrecen servicios sin conexión y orientado a la conexión
  - El servicio orientado a conexión no es una conexión física sino virtual porque no hay un cable dedicado entre los dos dispositivos. Fija antes un camino que es preferente, pero los paquetes luego, si es necesario, pueden seguir rutas diferentes

¿Dos capas que ofrecen los mismos servicios?

- La CR se ejecuta en los ruteadores de los operadores; la CT se ejecuta en los host de los usuarios finales
- Los usuarios finales no controlan la CR
- El proveedor a veces no cumple el compromiso de dar un servicio confiable: pierden paquetes, dañan paquetes, o la red se cae
- Se tiene que poner otra capa sobre la CR para asegurar la QoS
- Si accidentalmente se pierde una conexión de red, la CT establece automáticamente otra conexión de red con la CT remota

- La CT facilita a los usuarios crear aplicaciones de red sin tener que lidiar con la subred
- Subred: medios de transmisión y dispositivos de las empresas de telecomunicaciones
  - Empresas públicas de telecomunicaciones PSTN
  - ISPs

6.1.2 Primitivas de servicio de transporte

- Para que las aplicaciones puedan acceder al servicio de transporte, la CT facilita algunas primitivas de servicio
- Primitivas del servicio orientado a conexión

Primitiva	**Paquete enviado**	**Significado**
LISTEN	Ninguno	Se bloquea
CONNECT	Req	Intenta establecer una conexión
SEND	Datos	
RECEIVE	Ninguno	Se bloquea
DISCONNECT	Req	Intenta liberar la conexión

Diferencias entre los servicios de transporte y de red

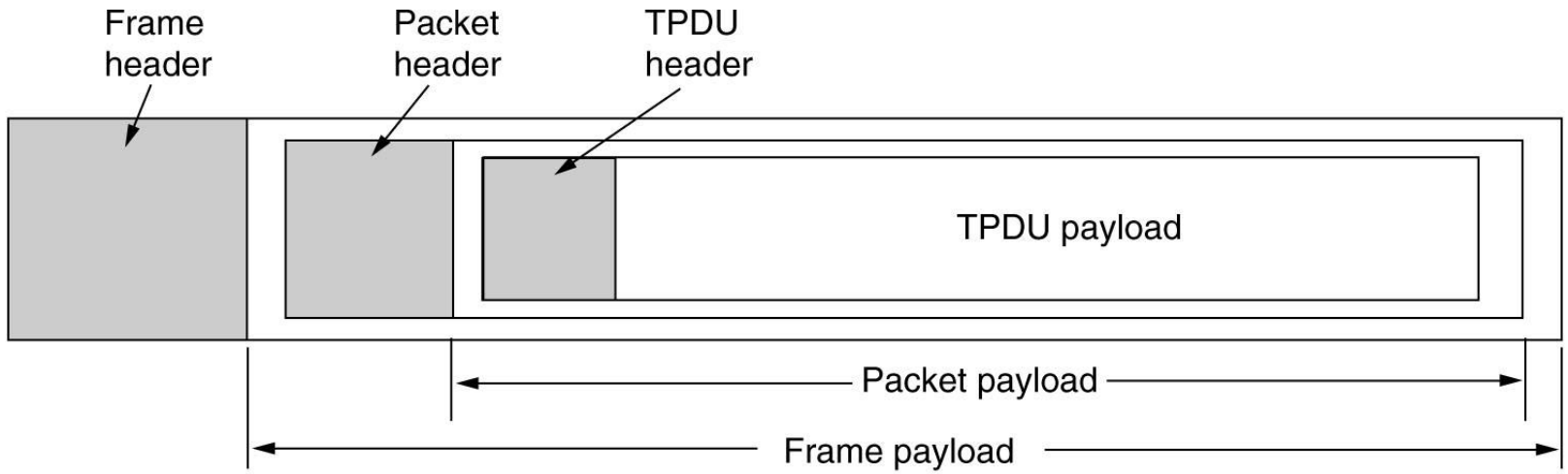
- **Primera diferencia**
- El servicio de la CR no es confiable porque depende de la red física que pierde paquetes, produce errores, o se cae
- El servicio de la CT orientado a conexión es confiable, trabajando sobre una red no confiable
- La CT oculta los defectos de la CR para que el flujo de bits esté libre de errores y pérdidas
- **Segunda diferencia**
- Los programadores tratan con las primitivas de transporte, que son fáciles de usar, no con las primitivas de red

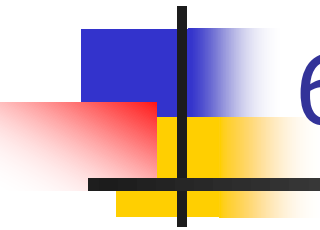
Funciones de la capa de transporte

- Son funciones transparentes para las aplicaciones

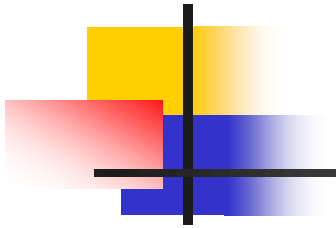
1. Controles de errores y pérdida de paquetes
2. Manejo de temporizadores
3. Confirmaciones ACKs
4. Retransmisiones

Anidamiento o encapsulamiento





6.2 Elementos de los protocolos de transporte



- El servicio de transporte se implementa mediante un protocolo de transporte
- Un protocolo de capa 4 se parece a un protocolo de capa 2 porque controla:
  - Errores
  - Secuenciación
  - Pérdida
  - Duplicación
  - Flujo

Diferencias entre los dos protocolos

- En la capa 2, dos sistemas se conectan P2P a través de un enlace físico
- En la capa 4 los dos sistemas se conectan *end-to-end* a través de una subred
- En la capa 2 el router no necesita especificar la dirección del otro enrutador
- En capa 4 hay que indicar la dirección destino
- En capa 2 es más sencillo establecer conexiones que en capa 4
- La subred almacena y reenvía paquetes. Esto produce complicaciones: retardos, pérdida, duplicación

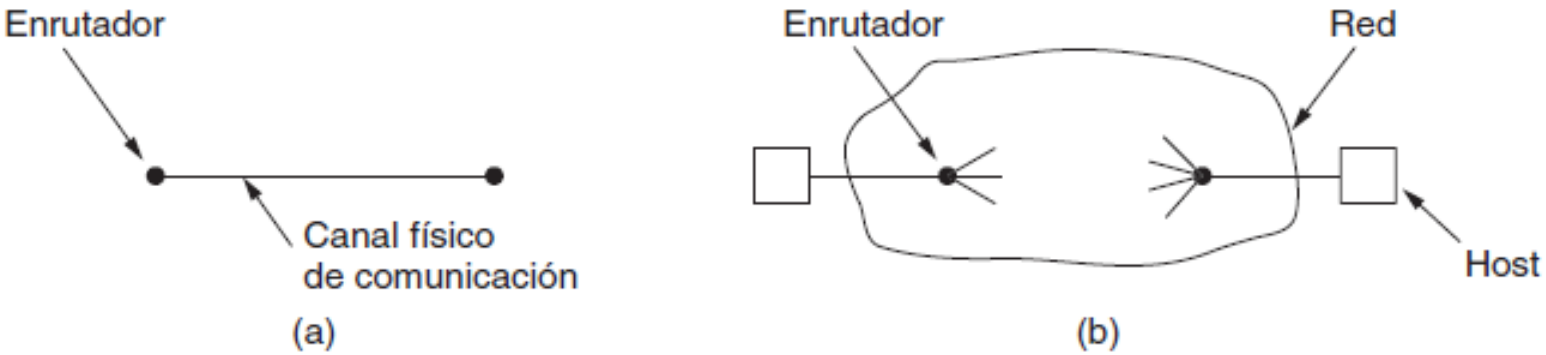
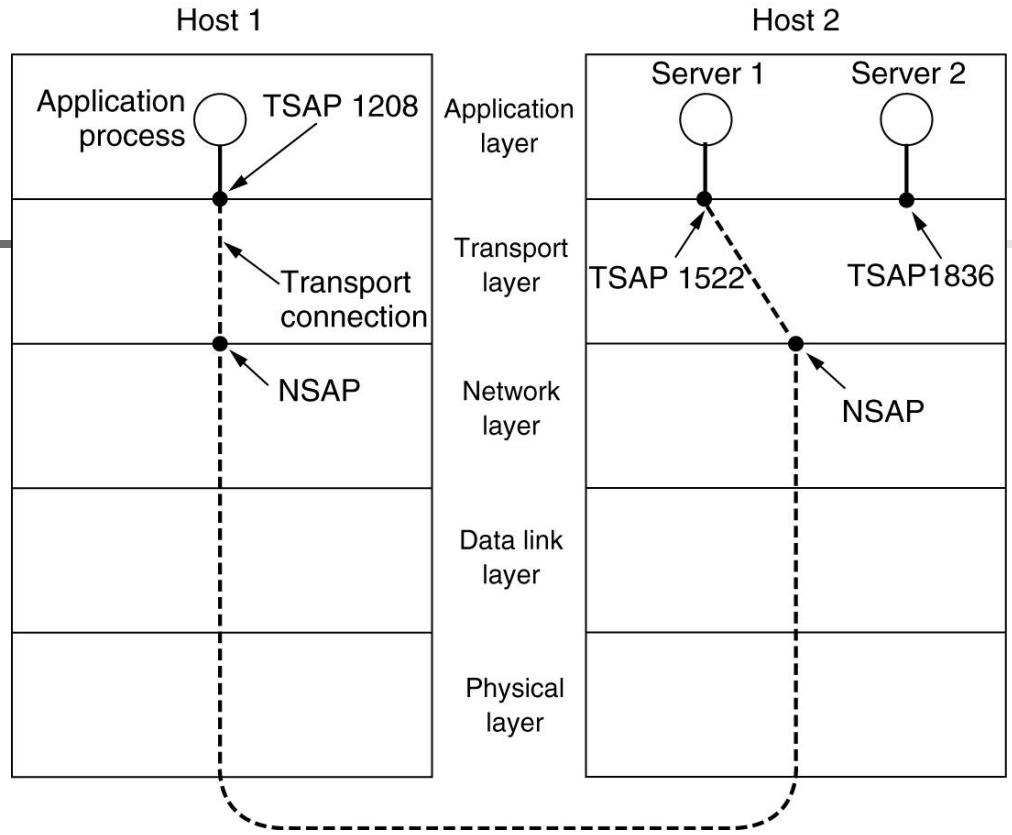


Figura 6-7. (a) Entorno de la capa de enlace de datos. (b) Entorno de la capa de transporte.

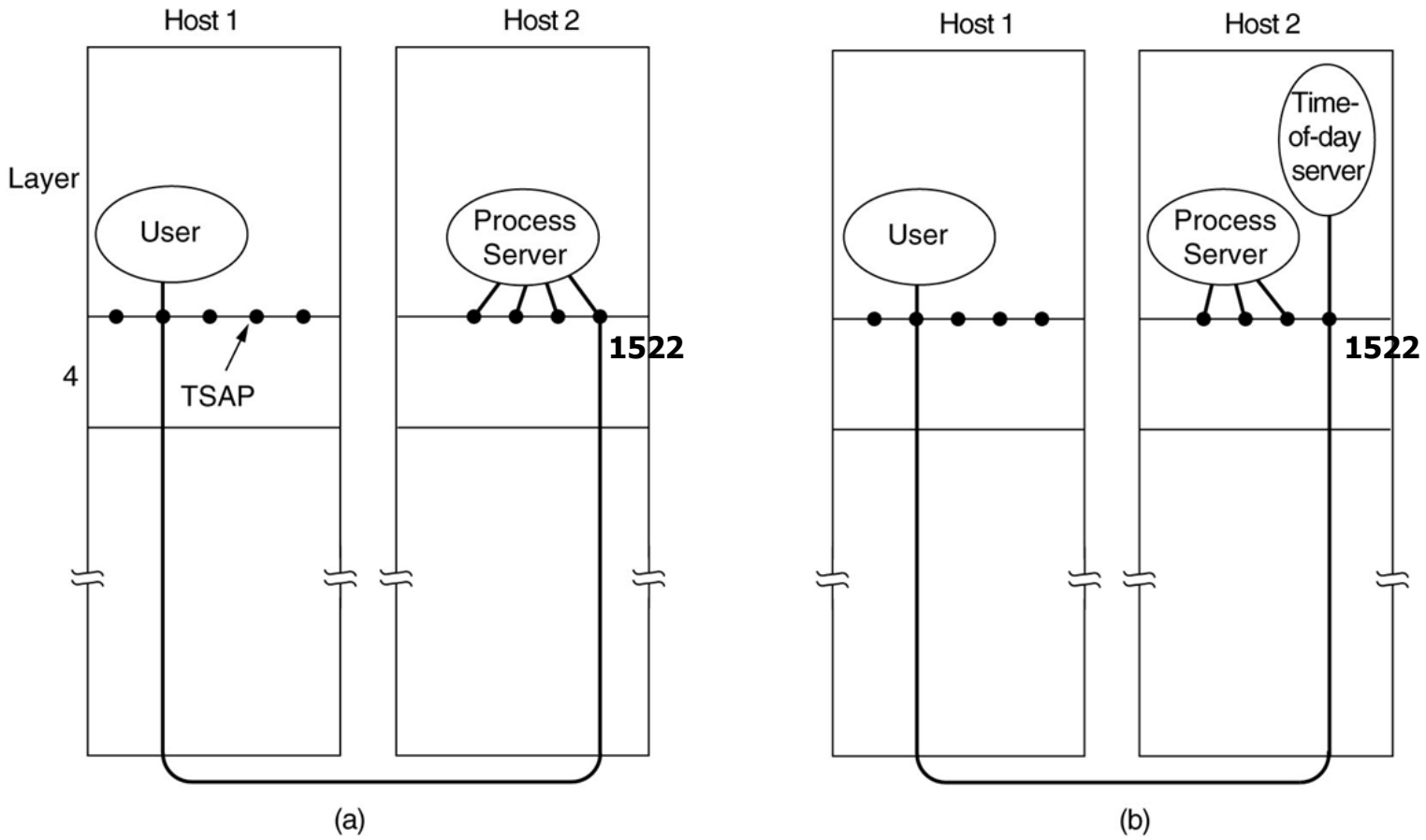
6.2.1 Direccionamiento

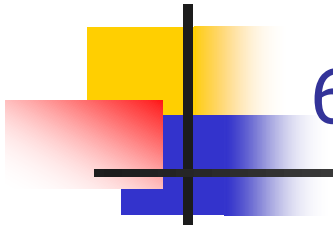
- A más de la dirección de red, los procesos, local y remoto que se comunican necesitan una dirección de transporte
- Los procesos pueden estar a la escucha de solicitudes de conexión (TCP), o directamente de solicitudes de información (UDP)
- En Internet estas direcciones se llaman puertos o TSAP



- El cliente sabe que el servidor de hora del día está conectado al TSAP 1522
- 1522 es un puerto *bien conocido* por los programadores
- Hasta 1023 son servicios bien conocidos /etc/services

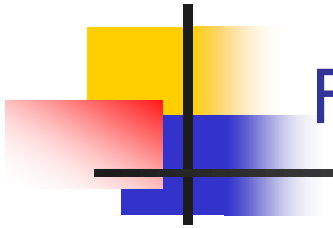
Servidor de procesos *inetd*
Apoderado o proxy de servidores de menor uso





6.2.2 Establecimiento de una conexión

- La red puede perder o duplicar paquetes, complicando las conexiones
- Un usuario establece una conexión con un banco para transferir dinero
- Si cada paquete de la transacción se duplica, se haría más de una transferencia de dinero



Forma de evitar la duplicación de paquetes

- Uso de direcciones de transporte desechables
  - Para cada transacción se genera un nueva dirección de transporte
  - Al cerrar la conexión se desecha la dirección y no se vuelve a usar
  - Si hay un paquete duplicado con una dirección anterior, éste se descarta

6.2.3 Liberación de una conexión

- Es más fácil liberar que establecer una conexión
- Hay 2 estilos de terminar una conexión:
  - Asimétrica
  - Simétrica

Liberación asimétrica

- Análogo al sistema telefónico
- Si una parte cuelga se interrumpe la conexión
- Es abrupta y puede haber pérdida de datos

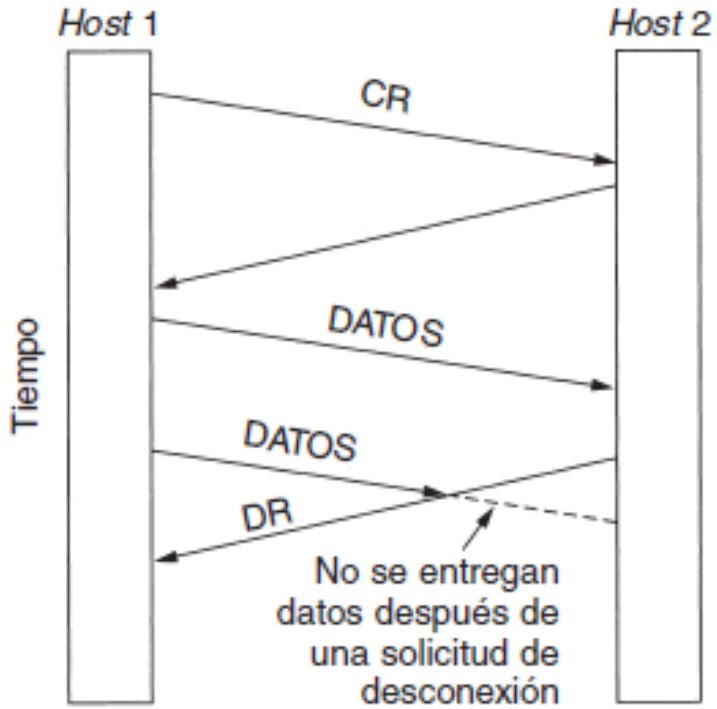
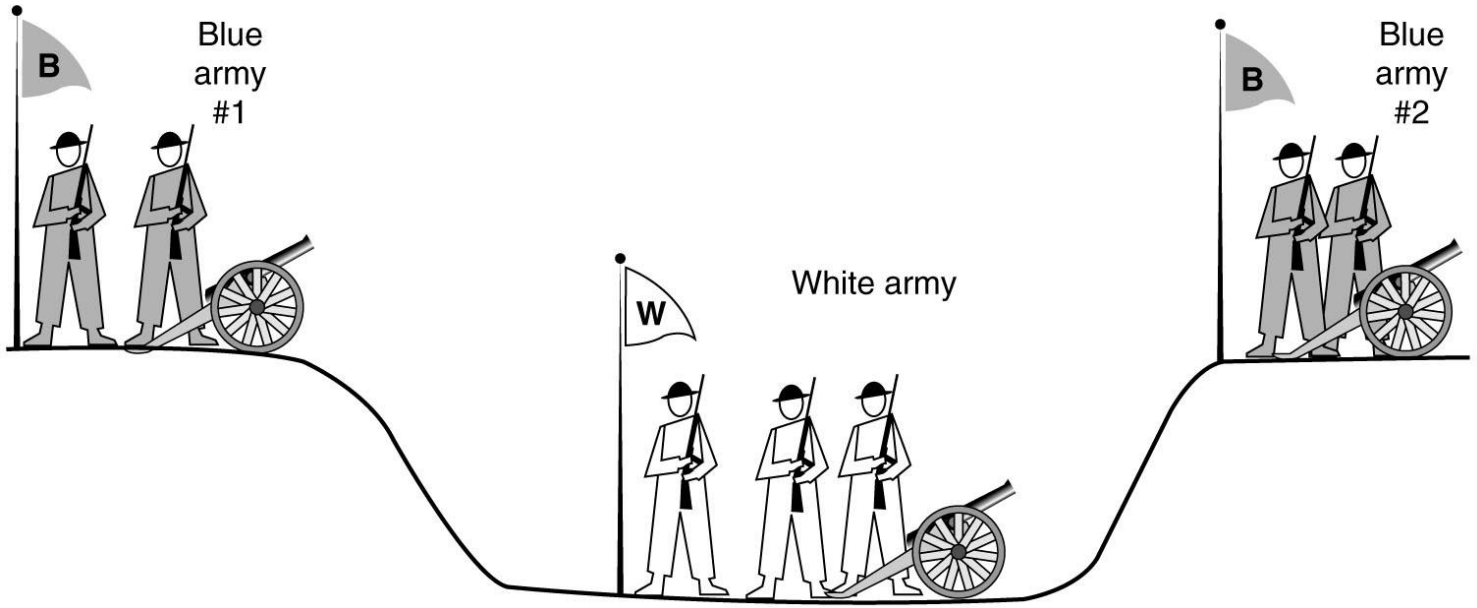


Figura 6-12. Desconexión abrupta con pérdida de datos.

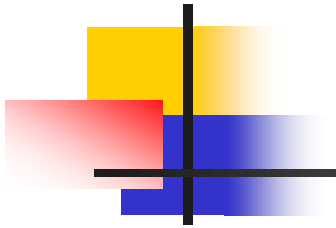
CR connection request; DR disconnection requiest

Liberación simétrica

- Protocolo más refinado
- Evita pérdida de datos
- Consta de 2 conexiones unidireccionales
- Cada dirección se libera por separado
- Un host puede recibir datos luego de enviar un DR



Los ejércitos grises deben sincronizarse para atacar simultáneamente al ejército blanco para poder vencer



- Problema: el emisor del mensaje final nunca sabrá que su mensaje llegó
- Si la una parte no sabe que la otra está lista para desconectarse, nunca ocurrirá la desconexión
- El protocolo puede fallar
- Un acuerdo de 3 vías por lo general funciona bien

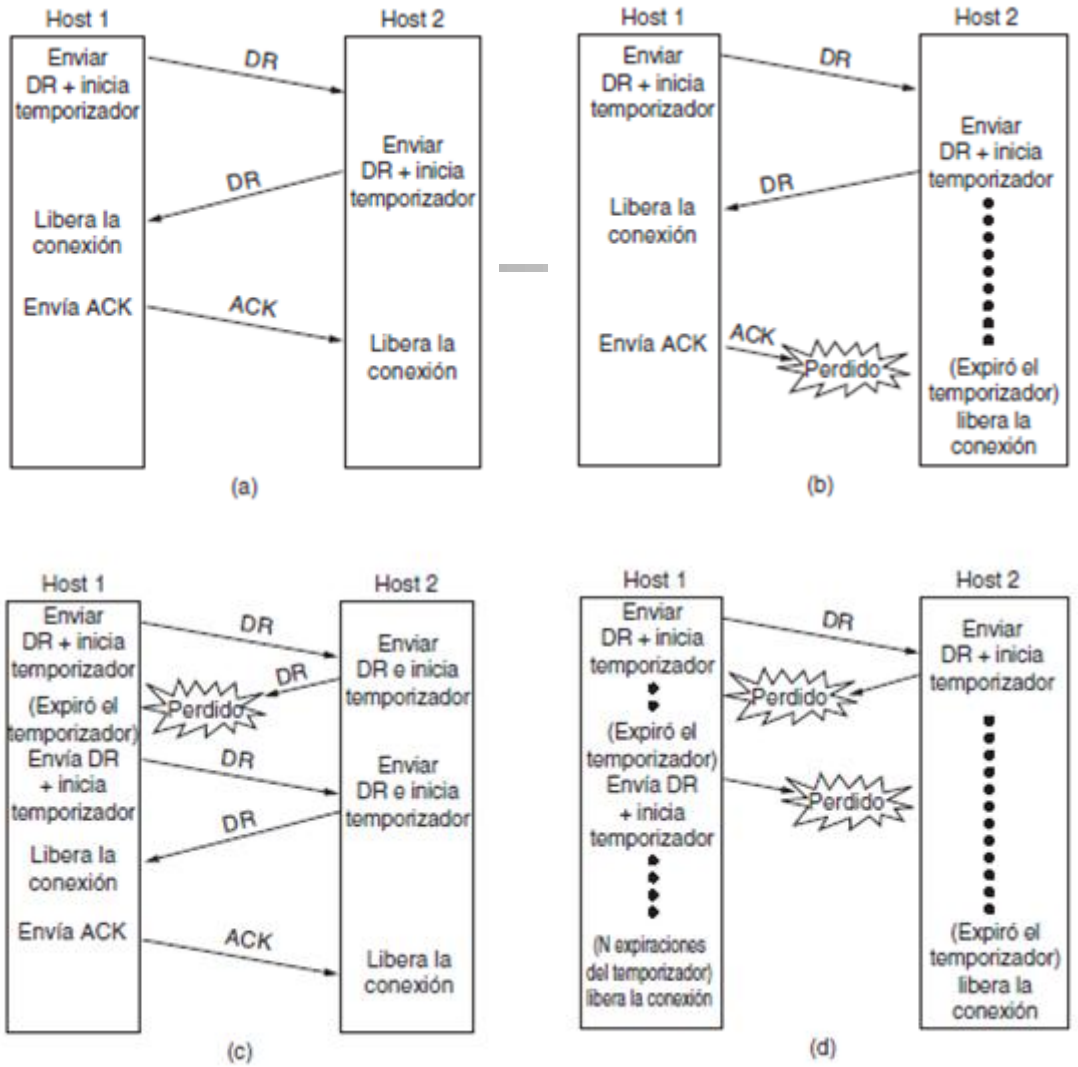


Figura 6-14. Cuatro escenarios de un protocolo para liberar una conexión. (a) Caso normal del acuerdo de tres vías. (b) Pérdida del último ACK. (c) Respuesta perdida. (d) Respuesta perdida y pérdida de los segmentos DR subsecuentes.

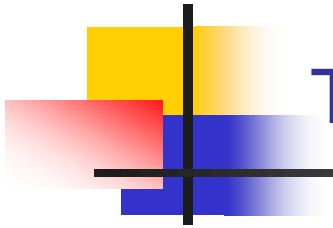
DR Disconnection Request

6.2.4 Control de flujo y almacenamiento en búfer en el manejo de las conexiones

- El control de flujo en la CT es similar al de la capa 2
- Las CTs tienen una mecanismo para no desbordar al receptor:
  - Parada espera: El emisor envía un paquete y espera un ACK
  - Ventana deslizante: El emisor envía varios paquetes y espera un ACK

- En la capa 2 existen buffers por línea
- En la CT también existen buffers
- El emisor tiene buffers de salida que almacenan las TPDU hasta recibir ACK del receptor
- El receptor tiene buffers de entrada para almacenar las TPDU mientras se procesa una TPDU

- Se puede manejar:
  - Un grupo de buffers para todas las conexiones de transporte, o
  - Un grupo de buffers para cada conexión
- La cuestión es el tamaño de los buffers



Tamaño de los buffers

- Se pueden asignar buffers **por conexión** de tres maneras:

1. Para TPDUs casi del mismo tamaño: Cadena de buffers de tamaño fijo
2. Para TPDUs de tamaño variable: Cadena de buffers de tamaño variable
3. Un solo buffer circular

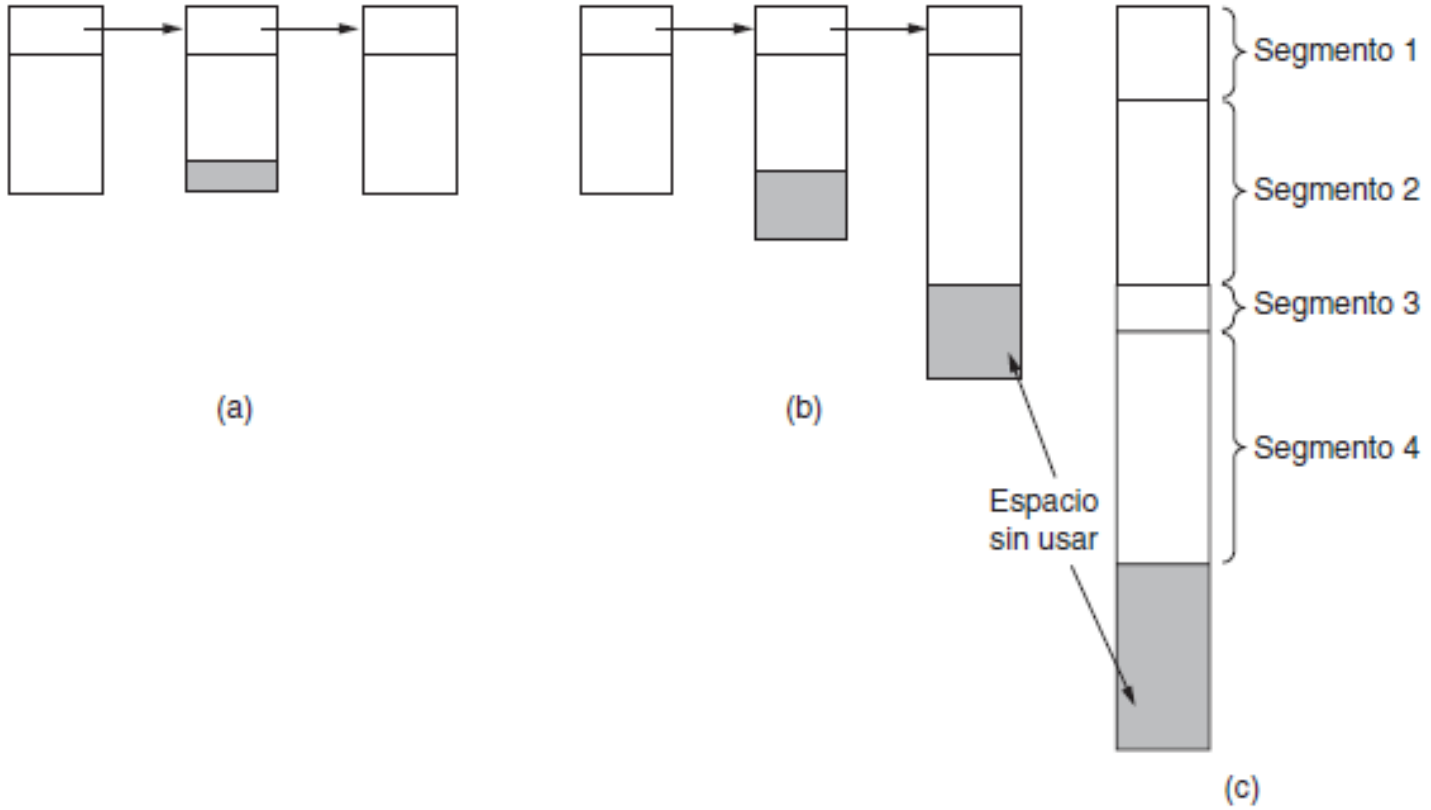
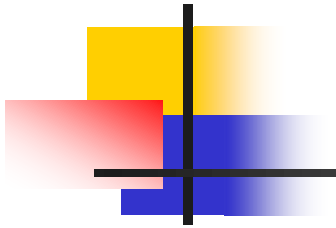
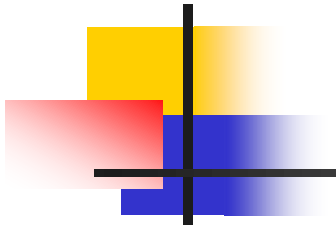


Figura 6-15. (a) Búferes encadenados de tamaño fijo. (b) Búferes encadenados de tamaño variable. (c) Un gran búfer circular por conexión.



- La CT permite que el emisor pida al receptor reservar suficientes buffers para que pueda recibir información
- Cuando se abren conexiones se asigna un número de buffers
- El número de buffers en origen y destino depende del tráfico cambiante
- Para tráfico con ráfagas no se asigna un número fijo de buffers, sino se lo asigna dinámicamente



- La RAM es económica. Los hosts tienen memoria suficiente. Los protocolos de CT no tienen este limitante
- El cuello de botella está la capacidad de las líneas

6.2.5 Multiplexación

- La multiplexión en la CT permite que múltiples aplicaciones de red en un dispositivo usen la red simultáneamente
- Así, los datos enviados se entregan a la aplicación correspondiente
- La multiplexión de varias conversaciones se da de tres maneras:

1. Un enlace físico
2. Un circuito virtual de capa 3 de OSI
3. Una conexión de capa 4

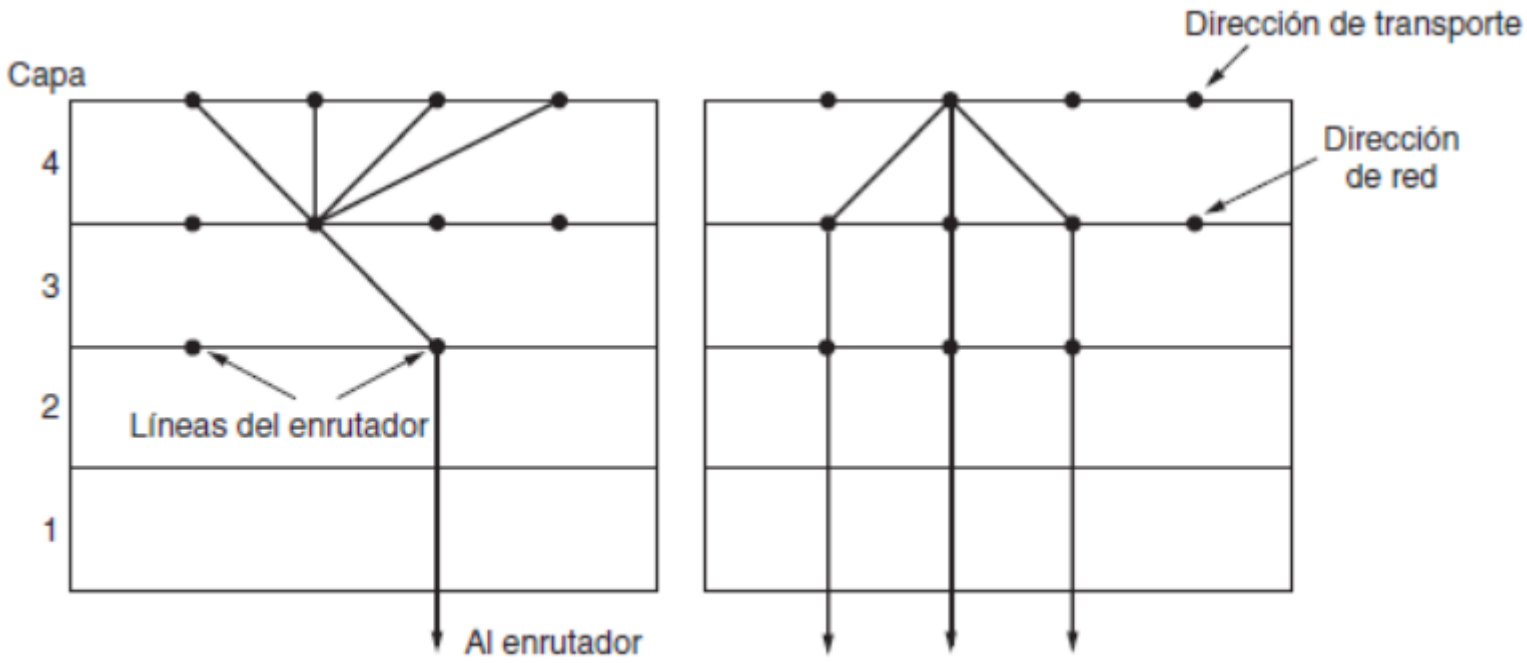
Multiplexión hacia arriba

- En el Modelo TCP/IP, la Capa de Internet no ofrece circuitos virtuales, solo datagramas (servicio sin conexión): no hay multiplexión
- En la CT es necesario multiplexar porque hay varios puertos y una sola dirección de red (IP en el modelo TCP/IP)
- Las conexiones de capa 4 comparten la dirección IP

Multiplexión hacia abajo

- En el Modelo OSI, La Capa de Red provee circuitos virtuales
- Si la conversación necesita más ancho de banda, se puede abrir más CVs y repartir el tráfico entre ellos
- Para ello, se necesitan varias direcciones de red
- En el Modelo TCP/IP los hosts solo tienen una dirección de red y no hay multiplexión hacia abajo

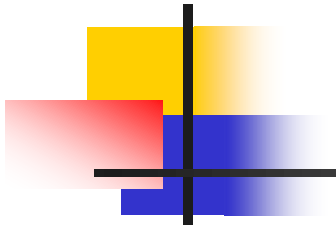
Modelo OSI



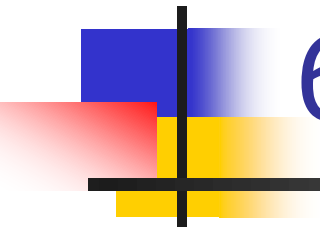
(a) Multiplexión hacia arriba. (b) Multiplexión hacia abajo.

6.2.6 Recuperación de caídas

- Servidores, clientes y routers están sujetos a caídas
- En la CR los datagramas o los circuitos virtuales se pueden perder
- Estos problemas los maneja la CT
- El host que se recupera de una caída puede continuar trabajando
- Mientras el servidor se recupera de una caída los clientes siguen trabajando

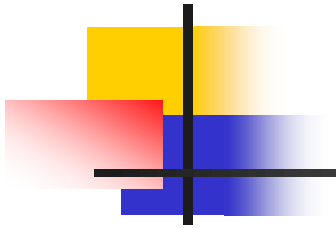


- Para recuperar su estado previo, el servidor difunde una *TPDU* a todos los hosts con los que hubo conexiones para:
  - Anunciar que acaba de reiniciarse y
  - Solicitar información del estado de todas las conexiones abiertas

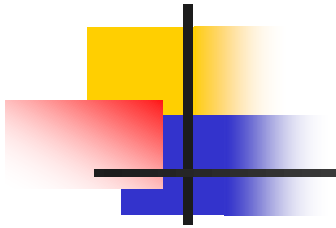


6.3 El protocolo de Internet UDP

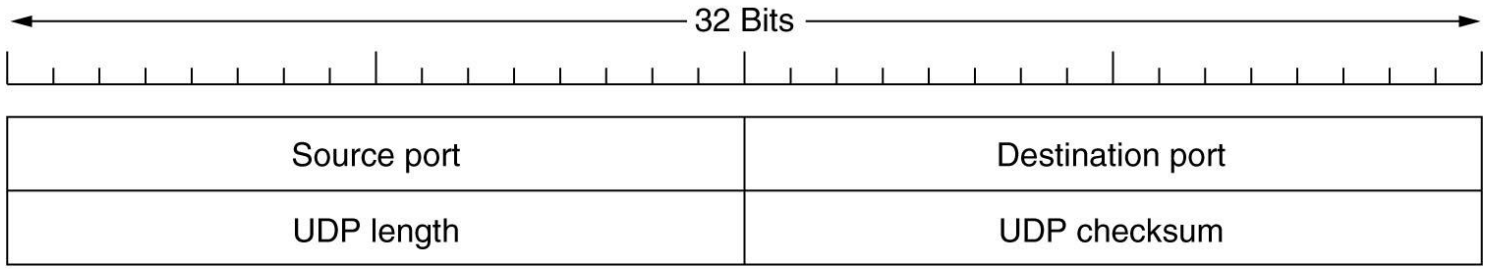
UDP User Datagram Protocol



- Internet tiene dos protocolos en la capa de transporte:
  - TCP que es orientado a la conexión
  - UDP que es sin conexión

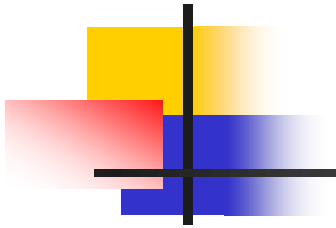


- Con UDP (1980) las aplicaciones funcionan sin conexión
- UDP divide los mensajes de las aplicaciones y los encapsula en segmentos
- Segmento: cabecera de 8 bytes + porción de mensaje (datos o carga útil)
- ¿Por qué no usar solo datagramas IP?
- Se usa UDP en lugar de solo IP para indicar los puertos origen y destino
- Sin los puertos, la CT no sabría a qué aplicación enviar el segmento
- El puerto origen se necesita cuando se debe enviar una respuesta al emisor



UDP tiene una cabecera (8 bytes)
EL chacksum incluye las direcciones IP

- UDP no controla:
  - flujo
  - errores
  - pérdida de segmentos
  - duplicación
  - retransmisión
- Esto lo realiza los procesos de usuario, si lo necesitara
- UDP ofrece a las aplicaciones una interfaz al protocolo IP
- Multiplexa varios procesos utilizando puertos



- UDP es útil en ciertas aplicaciones cliente-servidor
- El cliente envía una solicitud corta al servidor
- El cliente espera una respuesta corta
- Si se pierde la solicitud o respuesta, el cliente intenta de nuevo
- Ejemplos:
  - DNS utiliza UDP: Obt_direccion_IP(nombre_de_host)
  - Consultas rápidas y simples
  - Alertas y notificaciones
  - Juegos on-line

6.3.2 Llamada a procedimiento remoto RPC

- RPC *Remorte Procedure Call* se ideó en 1984 y se usa en sistemas distribuidos
- Es llamar a un proceso que reside en otro host para que ejecute una tarea y envíe una respuesta
- RPC hace que las aplicaciones de red sean más fáciles de programar y usar
- Así, los detalles de conexión se ocultan al programador de aplicaciones

Pasos para realizar un RPC

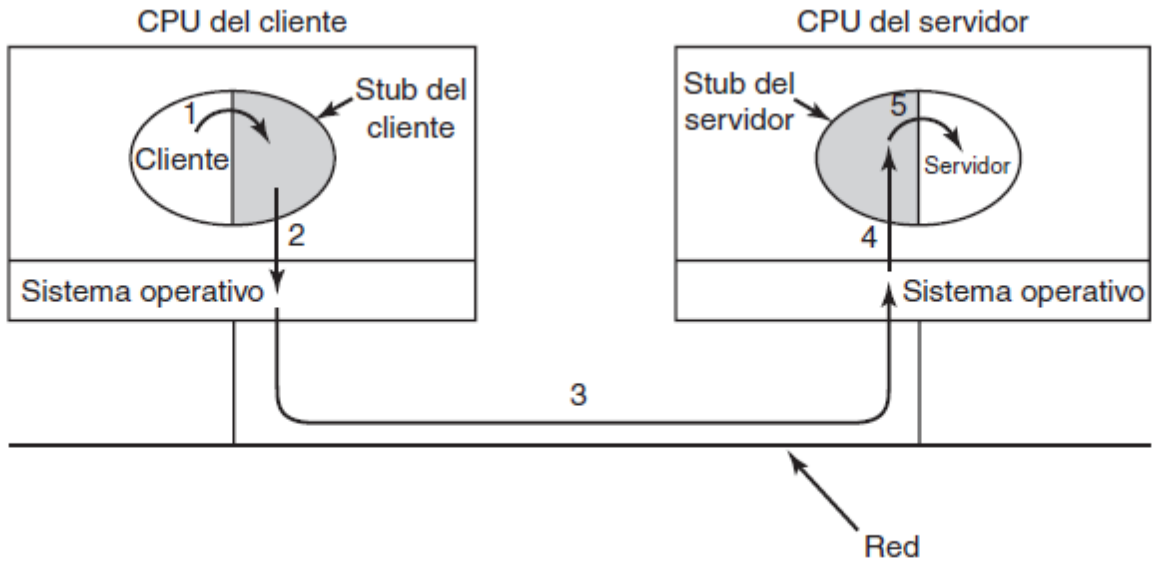


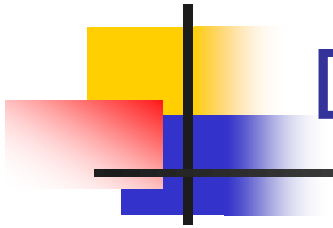
Figura 6-29. Pasos para realizar la llamada a un procedimiento remoto. Los stubs están sombreados.

Stub: interfaz con el sistema operativo

Pasos para realizar un RPC

1. El cliente llama al *stub* (subrutina de enlace) del cliente
2. *El stub del cliente* empaqueta el mensaje y llama al SO para enviarlo al servidor
3. El SO envía el mensaje al servidor
4. En el servidor, el SO pasa el paquete entrante al *stub del servidor*
5. El *stub del servidor* llama al procedimiento servidor con el mensaje desempacado

- La respuesta sigue la misma ruta en forma opuesta
- No se utilizan sockets

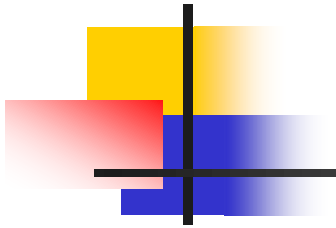


Desventajas de RPC

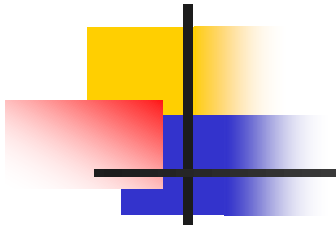
- Con RPC el paso de punteros es imposible porque los procesos residen en diferentes espacios de direcciones de memoria
- RPC se utiliza, con restricciones, muy ampliamente

6.3.3 Protocolo de transporte en tiempo real RTP

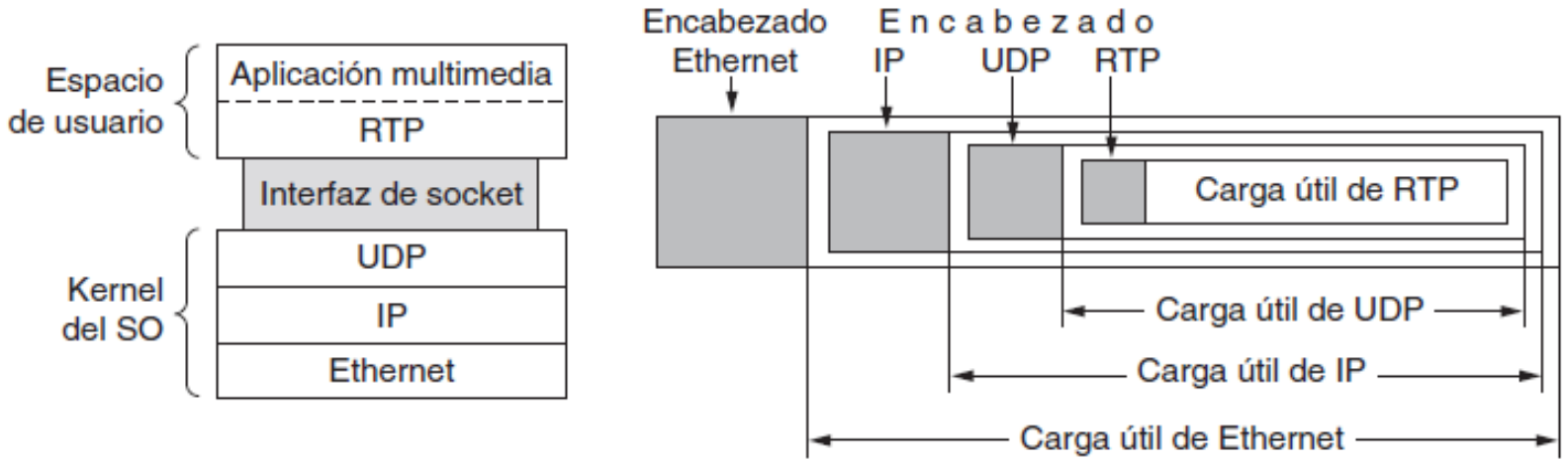
- *Tiempo real*: en vivo, momento actual, ahora
- *Información en tiempo real*: estar al tanto en forma inmediata del evento que ocurre al instante
- *Aplicación de tiempo real*: aplicación que informa de un evento con restricciones de retardo de transmisión y propagación
- *Tiempo de transmisión*: tiempo empleado por una estación para colocar todos los bits de una trama en el medio de transmisión
- *Tiempo de propagación*: Tiempo empleado por un bit en atravesar la red desde el origen al destino



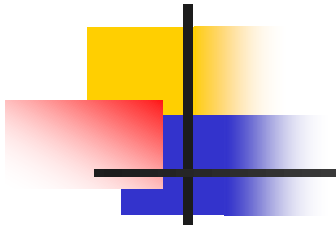
- UDP también se usa en aplicaciones en tiempo real
  - Sistemas de control de tráfico aéreo
  - Sistemas de monitoreo de signos vitales de un paciente
  - Aplicaciones para realizar operaciones financieras en la bolsa de valores
  - Sistemas de reserva de vuelos
  - Sistemas de control de procesos industriales: t^0, presión, rpm, . . .
  - Sistemas de control de generación distribución y consumo de energía eléc
  - Rastreo satelital para la industria del transporte
  - Videoconferencia



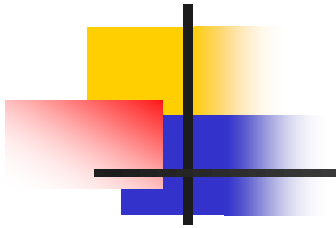
- Aplicaciones en tiempo real requieren un protocolo de transporte de tiempo real
- RTP Real-time Transport Protocol
- RTP se ejecuta sobre UDP



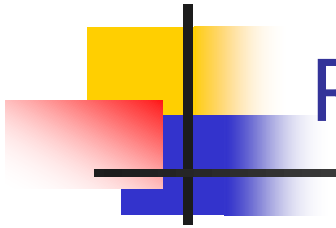
HDLC (High-Level Data Link Control) protocolo de capa de enlace punto a punto



- La aplicación de multimedia tiene **múltiples** flujos: audio, video y texto
- RTP multiplexa los flujos de tiempo real, función básica de RTP
- Luego se encapsula en paquetes RTP
- RTP es un protocolo independiente de la aplicación
- RTP se parece más a un protocolo de transporte
- RTP es un protocolo de transporte implementado en la capa de aplicación

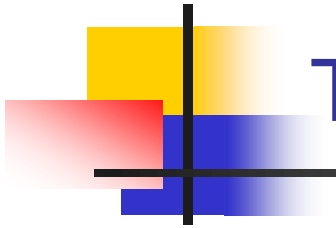


- Cada paquete enviado en el flujo RTP tiene un número secuencial
- Si falta uno, el receptor lo aproxima por interpolación
- La retransmisión no es práctica, pues llegaría muy tarde para ser útil
- No hay garantía de entrega



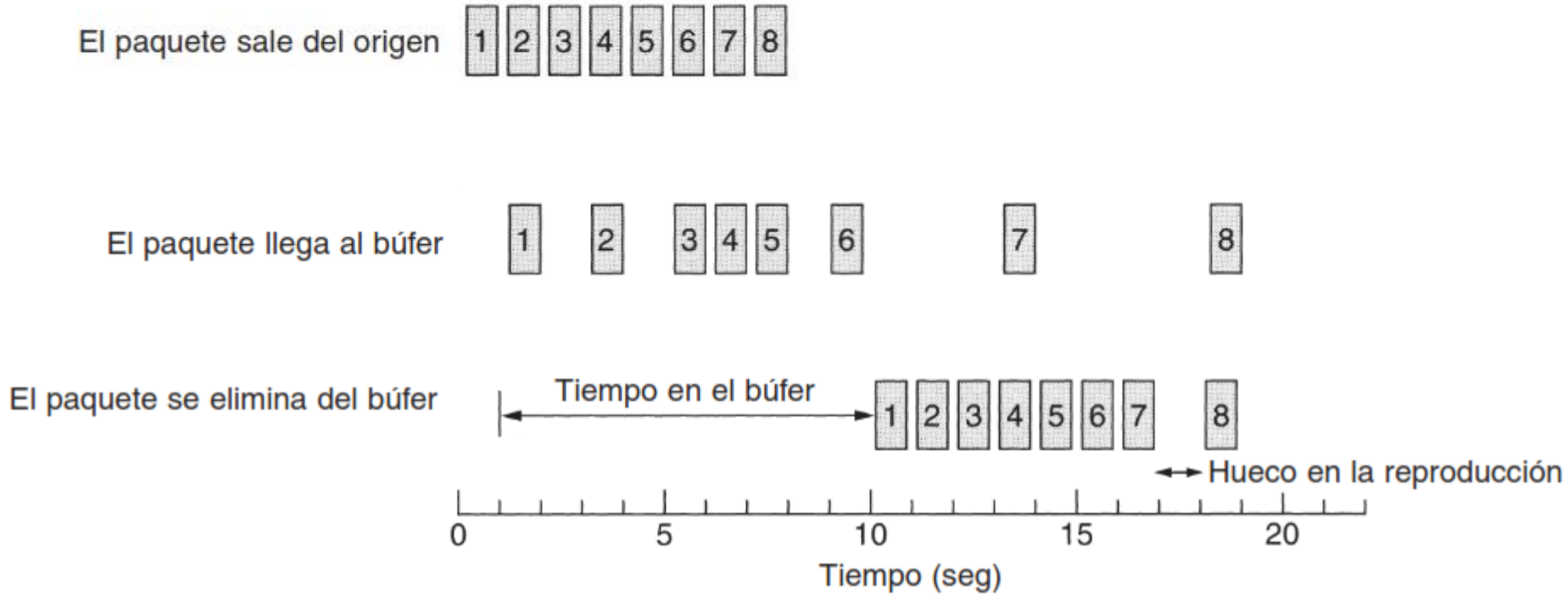
RTP no controla:

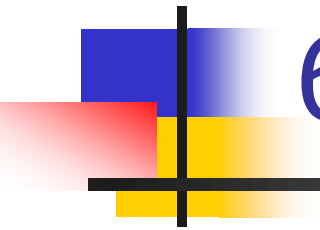
- flujo
- errores
- duplicación
- recepción
- Retransmisiones



Timestamping

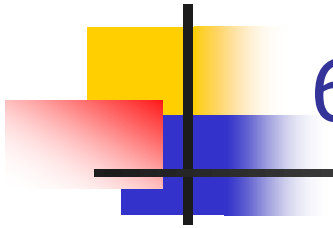
- Marcación del tiempo
- Es una característica de las aplicaciones en tiempo real
- El destino almacena una pequeña cantidad de paquetes y lo reproduce unos milisegundos después del inicio del flujo
- Reduce el efecto de la fluctuación
- Permite que la sincronización de múltiples flujos: video, voz, música
- Ejemplo: TV digital estéreo y doblada en varios idiomas


El paquete sale del origen
1 2 3 4 5 6 7 8
El paquete llega al búfer
1 2 3 4 5 6 7 8
El paquete se elimina del búfer
Tiempo en el búfer
1 2 3 4 5 6 7 8
Hueco en la reproducción
0 5 10 15 20
Tiempo (seg)



6.4 El protocolo de Internet TCP

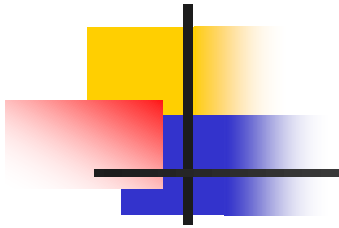
TCP Transport Control Protocol



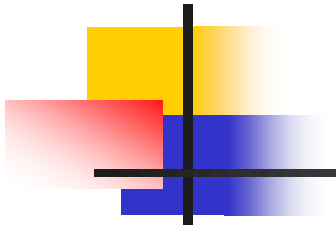
6.4.1 Introducción a TCP

- TCP (1981) da un flujo de bytes confiable de extremo a extremo a través de una subred no confiable
- La aplicación de usuario funciona sobre un servicio orientado a la conexión de TCP que, a su vez recibe un servicio sin conexión de IP

- La cabecera de UDP indica el tamaño del segmento
- La cabecera de TCP no indica el tamaño del segmento
- Este se calcula considerando el tamaño del datagrama IP menos el tamaño de la cabecera IP (que normalmente son 20 bytes)
- El tamaño del datagrama se indica en un campo de 16 bits
- El tamaño máximo del datagrama es $2^{16} - 1 = 65.535$ bytes
- El tamaño máximo del segmento es el tamaño máximo del datagrama menos el tamaño de la cabecera IP, esto es 65.535 – 20 bytes = 65.515 bytes



- En la práctica, un segmento TCP lleva 1460 bytes de datos + 20 bytes de cabecera TCP + 20 bytes de cabecera IP = 1500 bytes
- 1500 bytes es la capacidad máxima del campo de datos de la trama Ethernet



- Funciones de TCP:

1. Enviar los segmentos sin causar congestionamiento en la red
2. Controlar daño, y pérdida de segmentos
3. Realizar retransmisiones de ser necesario
4. Ordenar los fragmentos

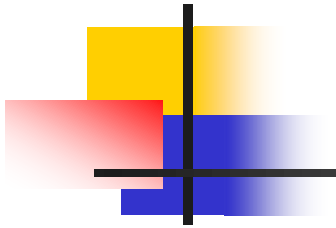
- Así, TCP da confiabilidad en la comunicación, que no da IP

6.4.2 El modelo del servicio TCP

- Emisor y receptor crean cada uno un punto terminal llamado socket
- Primero se crea un socket vacío (estructura de control en el SO)
- Luego se asigna al socket lo siguiente:
- Socket = dirección IP del host + puerto de 16 bits
- Un puerto es un TSAP
- Hay que establecer explícitamente una conexión entre sockets origen-destino

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.



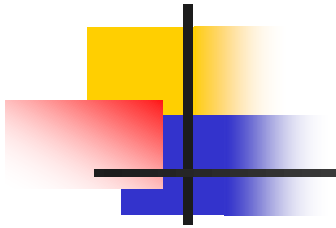
- Un mismo socket puede usarse para escucha de clientes (multiplexión), pero crea uno nuevo para responder a cada cliente
- Esto se hace en servidores que atienden a múltiples clientes a la vez
- El servidor distingue cada conexión con los identificadores de los dos sockets
- Los número de puerto menores a 1024 se llaman ***puertos bien conocidos*** y se reservan para servicios estándar

Algunos puertos bien conocidos

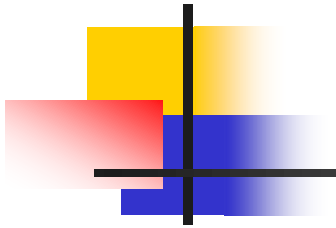
Puerto	Protocolo	Uso
20, 21	FTP	Transferencia de archivos.
22	SSH	Inicio de sesión remoto, reemplazo de Telnet.
25	SMTP	Correo electrónico.
80	HTTP	World Wide Web.
110	POP-3	Acceso remoto al correo electrónico.
143	IMAP	Acceso remoto al correo electrónico.
443	HTTPS	Acceso seguro a web (HTTP sobre SSL/TLS).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

Figura 6-34. Algunos puertos asignados.

(RTSP Real-Time Streaming Protocol; IPP Internet Printing Protocol)



- El demonio de cada servicio se conecta a su respectivo puerto; ej: el servidor ftp se conecta al puerto 21
- Hacerlo así podría llenar la memoria con demonios inactivos. Preferible *inetd*
- Conexiones TCP son full duplex
- TCP no soporta la multidifusión ni difusión, UDP sí



- Una conexión TCP es un flujo de bytes, no un flujo de mensajes
- TCP no reconoce mensajes de usuario
- El flujo de bytes es agrupado en diferentes segmentos TCP
- Si el usuario emite 4 mensajes de 512 bytes en un flujo TCP, estos mensajes se envían en cuatro, dos o un segmento de 512 bytes, 1024 bytes, o 2048 bytes
- La CT en el receptor no distingue si llegaron uno, dos o cuatro mensajes. TCP solo detectó que llegó un flujo de bytes a través de diferentes segmentos
- La CA se encarga de tomar el flujo de bytes y construir los mensajes

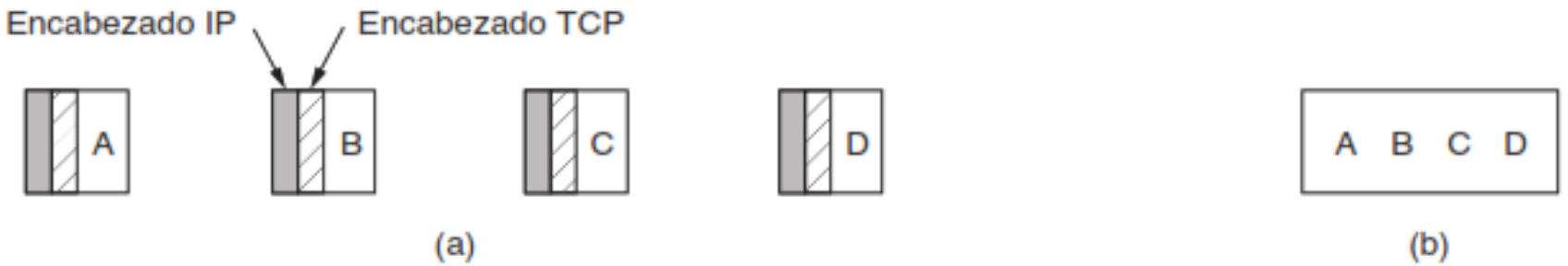
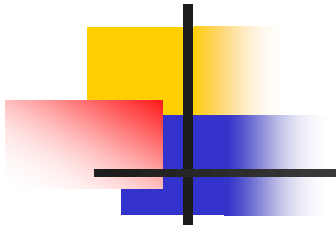


Figura 6-35. (a) Cuatro segmentos de 512 bytes que se envían como diagramas IP separados. (b) Los 2 048 bytes de datos que se entregan a la aplicación en una sola llamada READ.

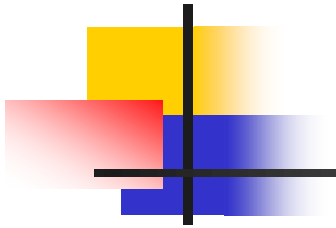
datagramas



- TCP envía al receptor los datos inmediatamente, o los almacena en un buffer para recolectar de la aplicación más datos y luego enviarlos
- Si la aplicación necesita que los datos se envíen inmediatamente activa el bit de la cabecera del segmento PUSH

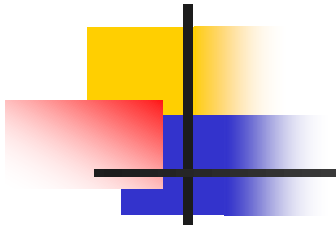
6.4.3 El protocolo TCP

- A cada byte de una **conexión** le corresponde un número de secuencia de 32 bits
- Antes, los enlaces entre enrutadores eran líneas telefónicas de $56Kbps$
- ¿A esa velocidad, en cuánto tiempo se agota la numeración?
- El número máximo es $2^{32} = bytes$
- $56Kbps = 56 \times 1024 \frac{bits}{s} = 57.344 \frac{bits}{s} = 7.168\ bytes/s$
- Al número máximo $2^{32} - 1$ se llega en $\frac{2^{32}\ bytes}{7.168\ bytes/s} \approx 166\ horas$
- Los números de secuencia se usan en el mecanismo de ventana deslizante



- Un segmento TCP tiene una cabecera de 20 bytes (+ una parte opcional) seguido de 0 o más bytes de datos
- TCP decide el tamaño de los segmentos en función de dos restricciones:

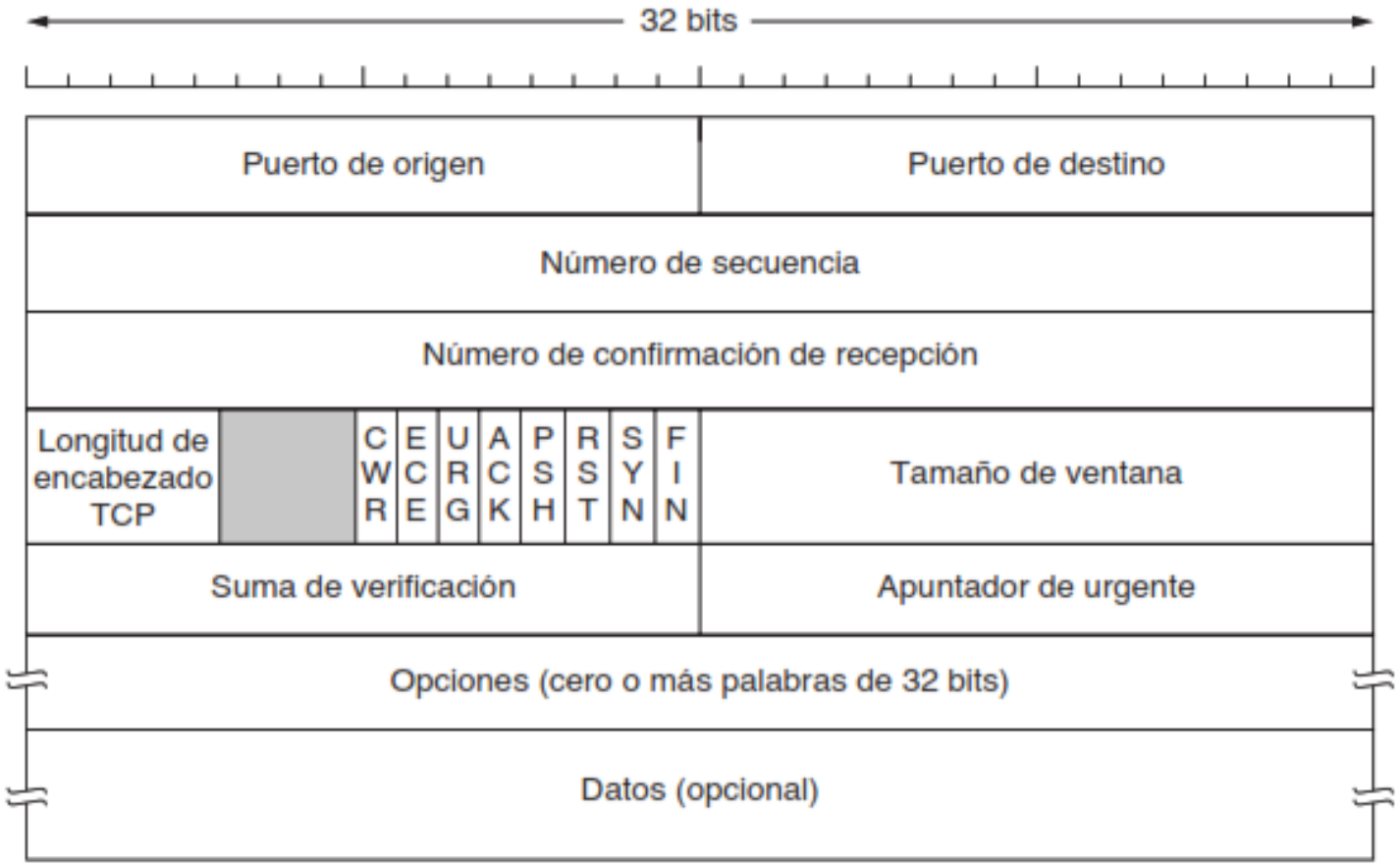
1. Que el segmento quepa en la carga útil de IP $2^{16} bytes = 65.515\ bytes$
2. Que quepa en la MTU de ruta (Path Maximum Transfer Unit) para evitar fragmentación
    1. Enthernet: MTU = 1500 bytes
    2. HDLC: MTU = 4096 bytes

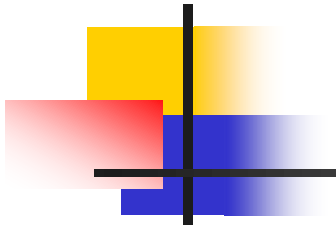


- TCP usa el protocolo de ventana deslizante:
  - Cuando se envía un segmento se inicia un temporizador de retransmisión
  - Cuando el segmento llega al destino, TCP envía un ACK con o sin datos, indicando el byte a recibir que tenga el siguiente número de secuencia
  - Si el temporizador expira, el emisor retransmite el segmento

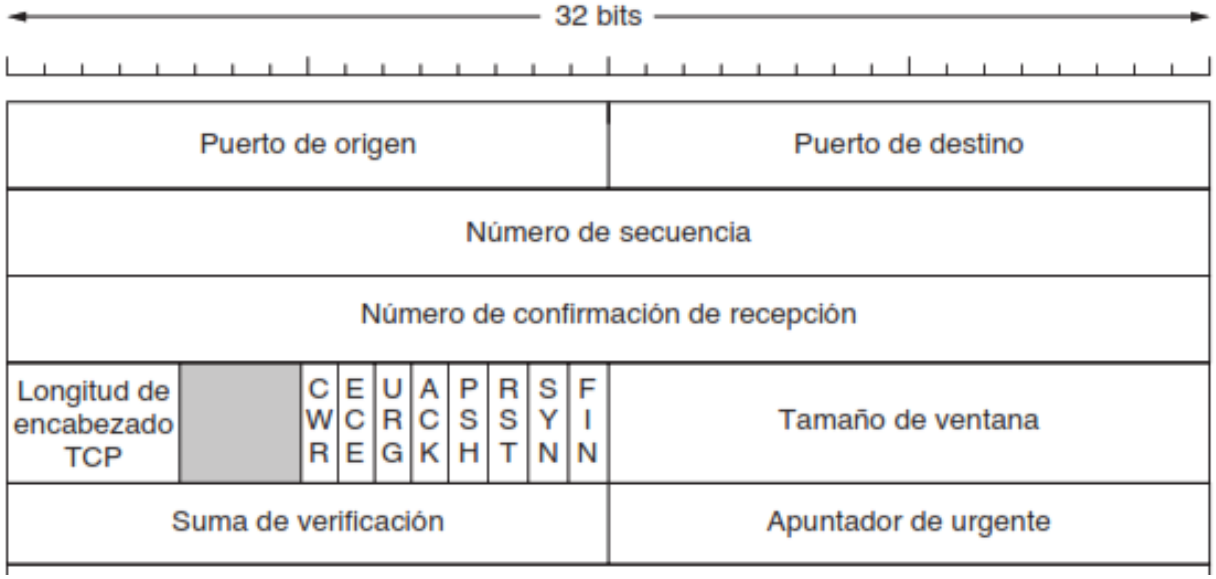
6.4.4 Encabezado del segmento TCP

20 bytes

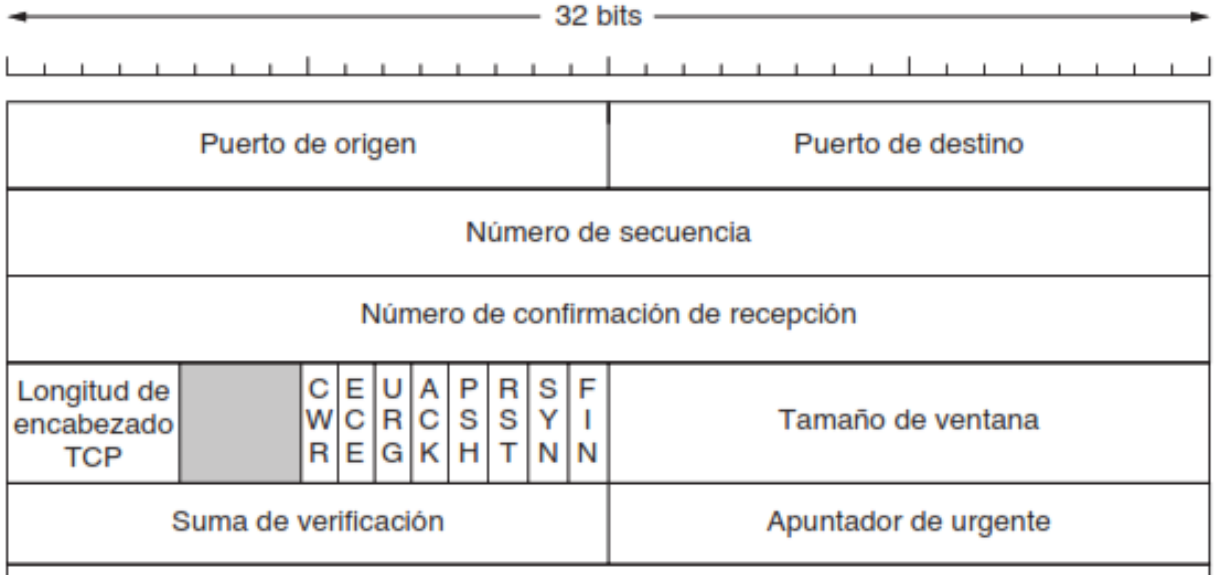




- Cada segmento comienza con un encabezado de formato fijo de 20 bytes
- El encabezado puede ir seguido de opciones
- Tras las opciones, si las hay, están los de datos de usuario
- Los segmentos sin datos se usan para ACK y mensajes de control

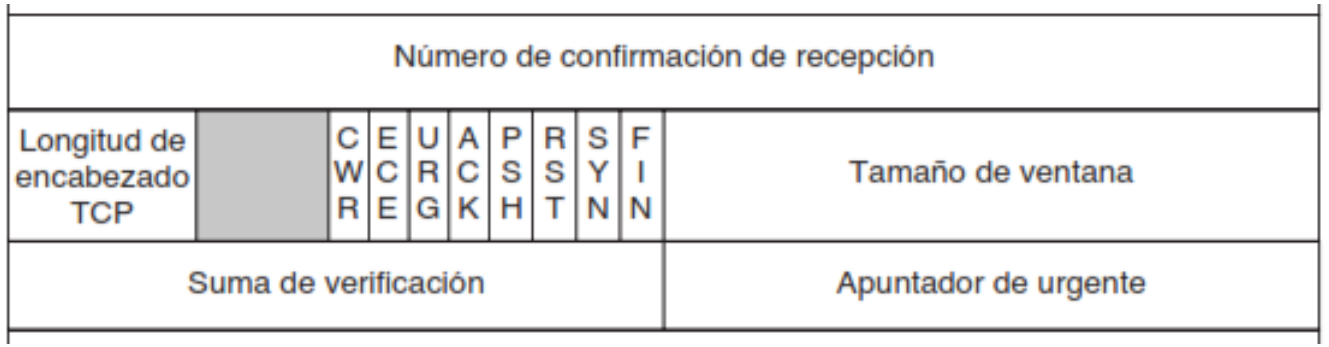


- *Puerto origen* y *puerto destino*
- Socket local de 48 bits: dirección IP + puerto
- El identificador de conexión se llama 5-tupla. (5 piezas de información): IP origen, puerto origen, IP destino, puerto destino, protocolo TCP
- *Número de secuencia*. Indica el primer byte del segmento



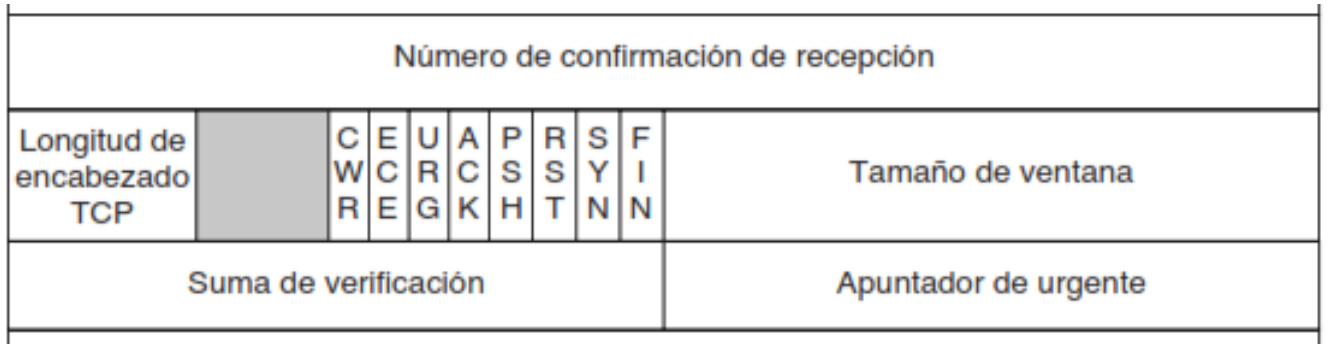
- *Número de confirmación de recepción* (ACK). Es el número del 1er. byte del siguiente segmento esperado, no es el número del siguiente segmento
- *Longitud del encabezado*. Con 4 bits expresa en palabras de 32 bits
- Longitud máxima: $(2^4 - 1)\frac{32}{8} = 60\ bytes$
- Luego hay 4 bits que no se utilizan

Ocho Indicadores de 1 bit

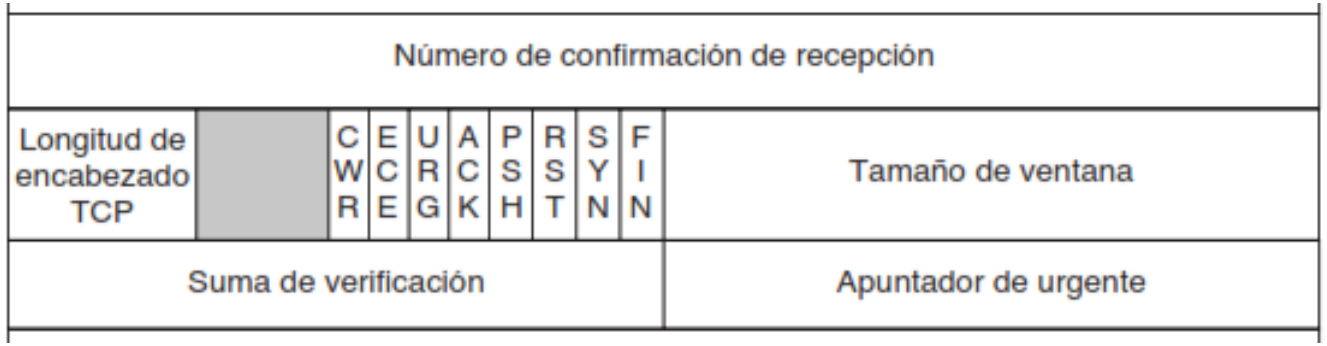


- CWR Congestion Window Reduced. El emisor avisa al receptor que ha reducido el tamaño de la ventana porque hay congestión en la red
- ECE Explicit Congestion notification-Echo. la red pide al emisor bajar la velocidad debido a que existe congestión
- Pero no lo hace directamente porque los routers no tienen CT, marcan con 11 el último subcampo ECN de *Servicios Diferenciados* del datagrama IP
- El datagrama llega al receptor y TCP marca el campo ECE en el segmento de regreso al emisor

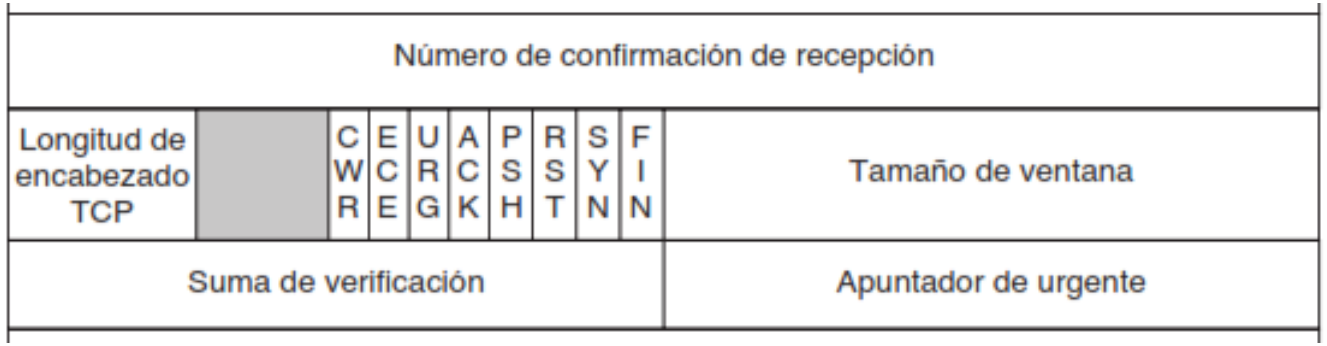
Ocho Indicadores de 1 bit



- URG Indica que hay datos que deben ser entregados con urgencia
- ACK Indica que el *Número de confirmación de recepción* es válido
- PSH La aplicación pide a TCP enviar el mensaje inmediatamente al receptor y en el receptor que entregue inmediatamente a la aplicación
- RST Restablece la conexión debido a la caída del host
- SYN Establece una conexión
  - CONNECTION REQUEST: SYN = 1, ACK = 0
  - CONNECTION ACCEPTED: SYN = 1, ACK = 1



- FIN Libera una conexión. El emisor puede seguir recibiendo datos
- Tamaño de ventana (Capítulo 3. Canalización) El control de flujo se maneja usando una ventana de tamaño variable. A menor tamaño menor flujo
- El tamaño máximo de ventana es $2^{16} = 64\ KB$ antes de recibir el primer ACK
- Si es 0, el receptor necesita de momento un descanso



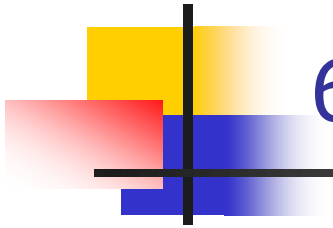
- Suma de verificación. Incluye el encabezado TCP, los datos y las direcciones IP
- Incluir direcciones IP viola la jerarquía de protocolos
- *Apuntador de urgente.* Trabaja con el bit *URG*. Indica el desplazamiento en bytes en el que están los datos urgentes a partir del campo *Número de secuencia* del primer byte del segmento actual

Campos opcionales

- Estos campos ofrecen flexibilidad y permiten adaptar TCP a diversas necesidades:
  - Nuevas funcionalidades
  - Optimización del rendimiento
  - Gestión de congestión
  - Seguridad: autenticación y el cifrado

6.4.5 Establecimiento de una conexión TCP

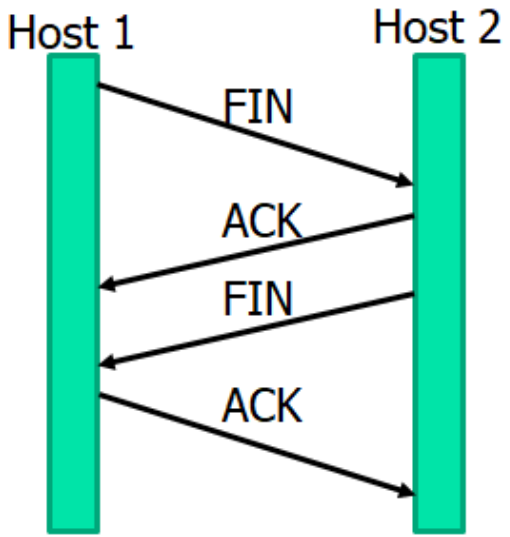
- El servidor espera una conexión
- El cliente solicita una conexión indicando dirección IP y puerto destino, y tamaño máximo del segmento TCP a aceptar
- El cliente espera una respuesta del servidor
- Si el puerto especificado es el correcto, el servidor acepta la conexión
- La conexión se realiza en 3 vías
  1. El cliente solicita conexión
  2. El servidor lo aceptación
  3. El cliente acusa recibo de la aceptación

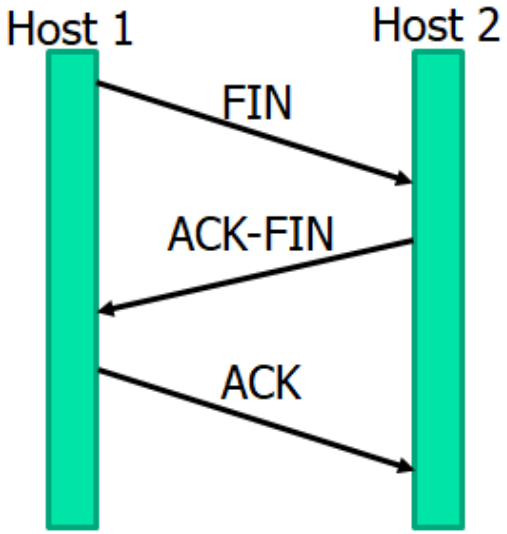


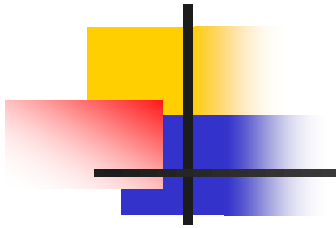
6.4.6 Liberación de una conexión TCP

- Las conexiones TCP son full dúplex, formadas con dos conexiones simplex
- Cada conexión simplex se libera independientemente
- Una de las partes envía un segmento TCP indicando que ya no va a seguir enviando datos, cerrándose ese sentido de la conexión
- Pero puede seguir recibiendo datos
- Cuando ambos sentidos se cierran se libera la conexión

- Normalmente se necesitan 4 segmentos TCP para liberar una conexión: un FIN y un ACK para cada sentido
- Es posible que el primer ACK y el segundo FIN estén en el mismo segmento, reduciéndose la cuenta total a 3

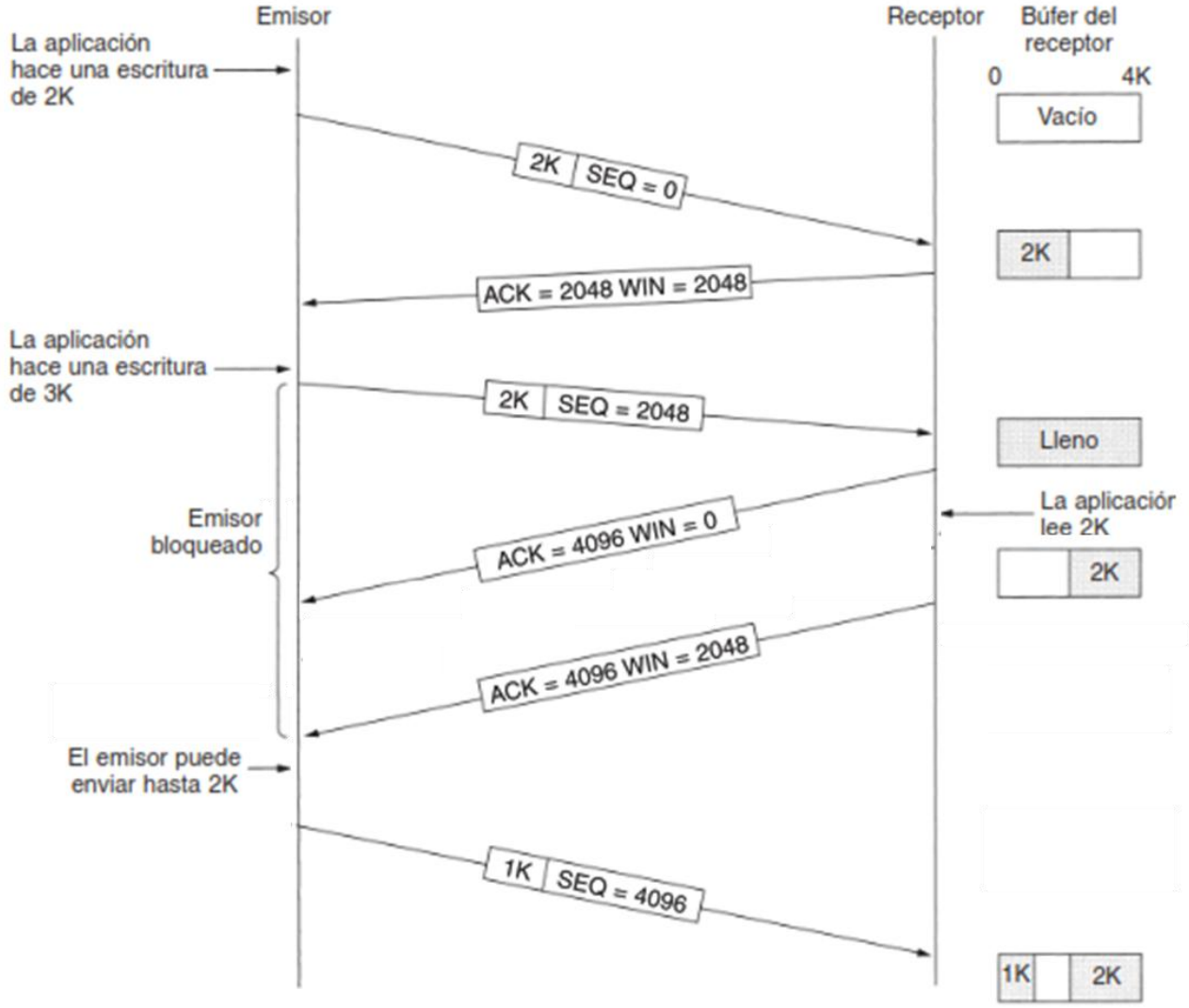






- Para evitar el problema de los 2 ejércitos, se usan temporizadores
- Si no llega la respuesta a un FIN en un máximo de dos tiempo de la vida de un paquete, el emisor de FIN libera la conexión
- Tarde o temprano el otro lado notará que nadie le está escuchando, también liberará la conexión

6.4.8 Política de transmisión de TCP

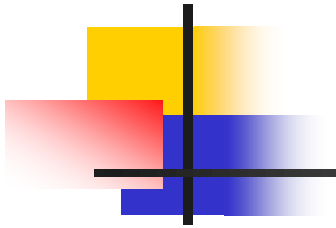


- No se requiere que el emisor envíe datos tan pronto como llegue de la aplicación
- Al inicio de la transmisión, el emisor podía esperar otros 2KB para transmitir un segmento con una carga útil de 4 KB
- Tampoco que se requiere que los receptores envíen confirmaciones de recepción tan pronto como sea posible
- El receptor podía esperar procesar todos los segmentos del buffer, y luego anunciar que tiene una ventana de 4 KB, y no sólo de 2KB

6.4.9 Control de congestión en TCP

- Si la carga inyectada a la red es mayor que la que puede manejar, se genera una congestión
- La capa IP intenta controlarla con los protocolos de enrutamiento
- Pero el trabajo pesado recae sobre TCP
- La solución a la congestión es disminuir la tasa de datos de los emisores
- TCP lo soluciona manipulando dinámicamente los tamaños de las ventanas

- El primer paso es detectar la congestión
- Se detecta observando las expiraciones de temporizador a través del campo TTL
- Expiración de temporizador implica la retransmisión de paquetes
- La retransmisión implica aumento de carga de la red
- La expiración de un temporizador por un paquete perdido, puede deberse a:
  - Ruido en la línea
  - Desbordamiento en un enrutador congestionado
- No hay pérdida de paquetes en líneas de fibra óptica En redes inalámbricas sí
- La mayoría de esas expiraciones en Internet se deben a congestión



- TCP toma acciones preventivas y acciones correctivas frente a la congestión
- Evitación o prevención de congestiones
  - Al establecer una conexión, emisor y receptor acuerdan un tamaño de ventana tal que no haya derrame en el receptor
- Pero aun puede ocurrir congestión en la red
  - El temporizador caduca y el emisor baja la velocidad

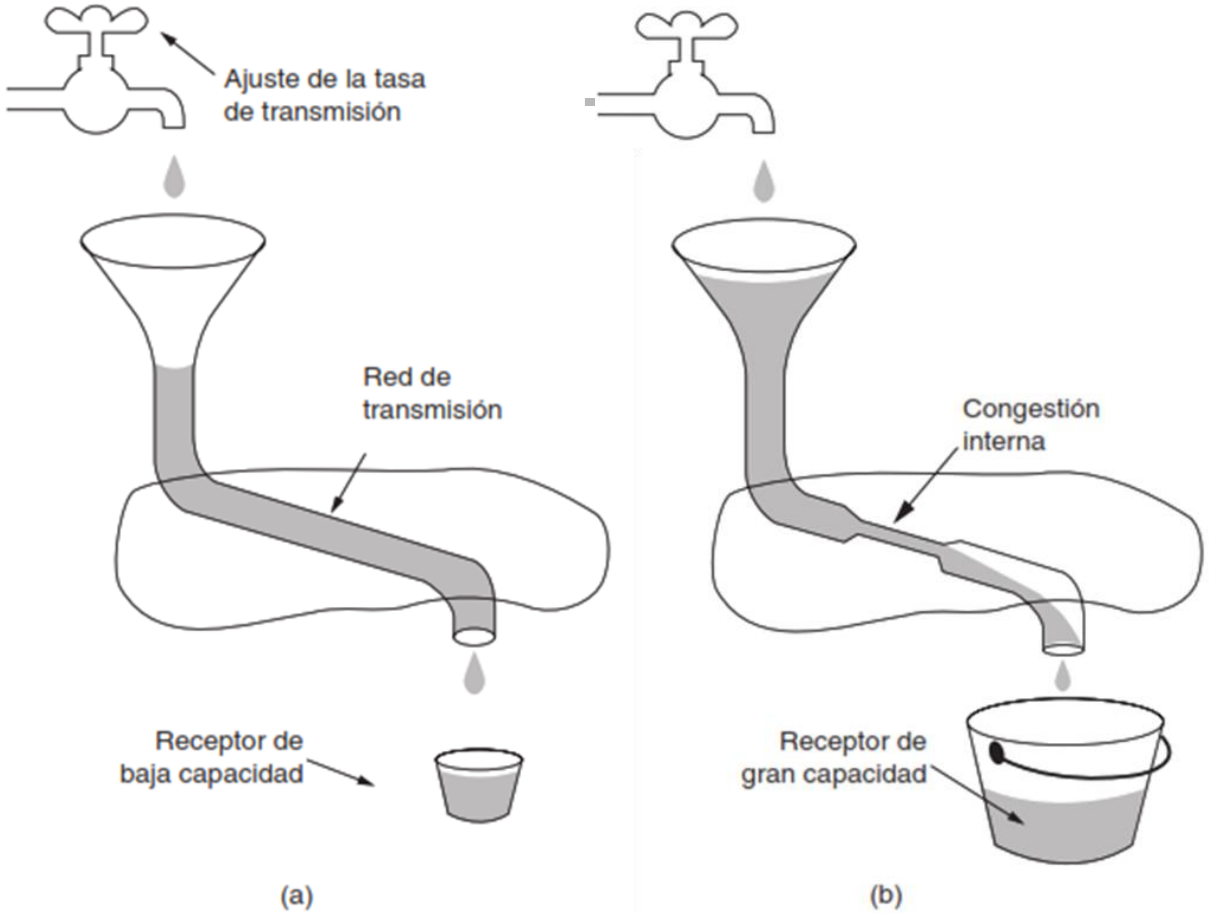
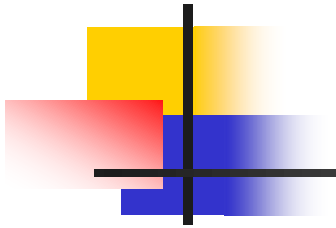
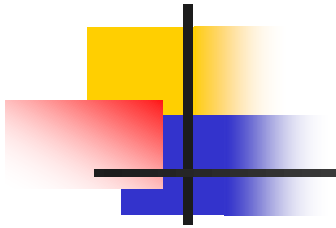


Figura 6-22. (a) Una red veloz que alimenta a un receptor de baja capacidad. (b) Una red lenta que alimenta a un receptor de alta capacidad.



- En Internet hay dos problemas potenciales
  - Capacidad de la red y
  - Capacidad del receptor
- La estrategia es manejarlos por separado
- Para ello, cada emisor mantiene dos ventanas:
  - La ventana de recepción que ha otorgado el receptor, y
  - La ventana de congestión
- Cada una indica la cantidad de bytes que puede enviar el emisor
- La cantidad que puede enviar el emisor es la menor de las dos ventanas



- Hay dos algoritmos que permiten determinar la cantidad de bytes que puede enviar el emisor sin producir congestión ni en el receptor ni en la red
  - Arranque lento
  - Algoritmo de Internet

Algoritmo de arranque lento

Tamaño de la ventana de congestión del emisor

Segmento confirmado

Tamaño de la ventana **luego** de la

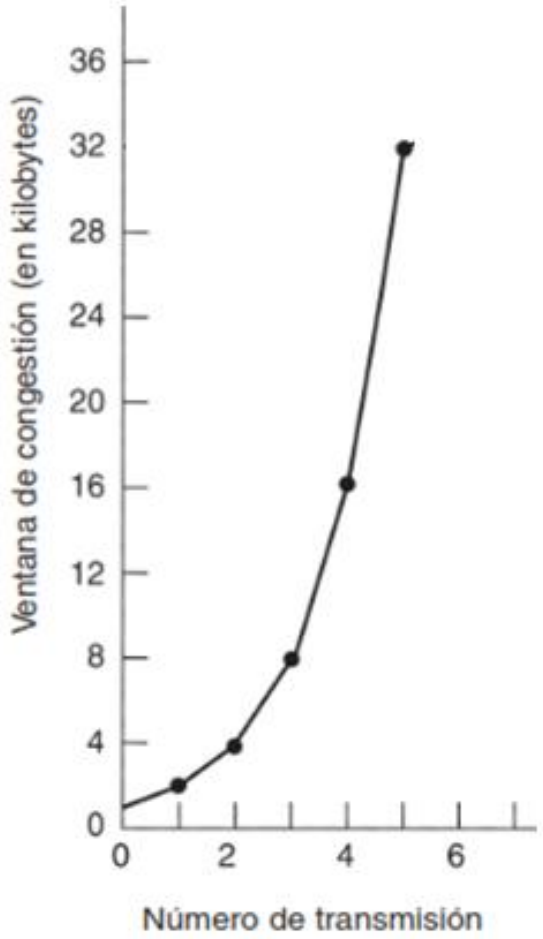
1ra. transmisión de la ventana

2da. transmisión de la ventana

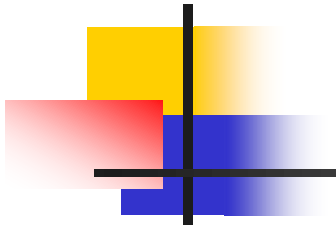
3ra. transmisión de la ventana

Algoritmo de arranque lento

- Al establecer una conexión, el emisor asigna a la ventana de congestión el tamaño de segmento máximo, entonces envía un segmento
- Si se recibe la confirmación de recepción de este segmento, el emisor aumenta el tamaño de la ventana en un segmento máximo y envía dos segmentos de tamaño
- A medida que se confirma cada segmento, se aumenta el tamaño de la ventana en un segmento
- Cada ráfaga confirmada duplica el tamaño de la ventana



Ejemplo del algoritmo de Arranque Lento



- La ventana de congestión sigue creciendo exponencialmente hasta:
  - Ocurrir una expiración del temporizador, o
  - Alcanzar el tamaño de la ventana receptora

Algoritmo de Internet

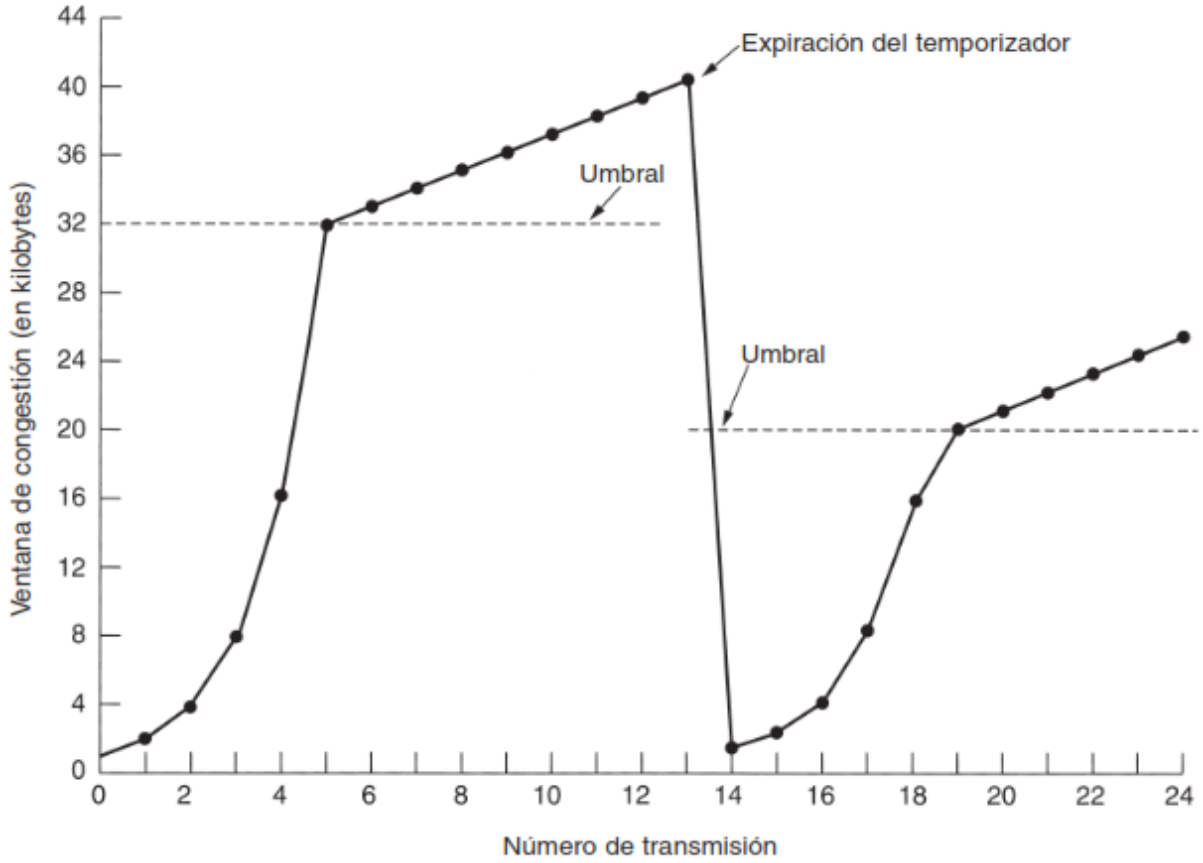
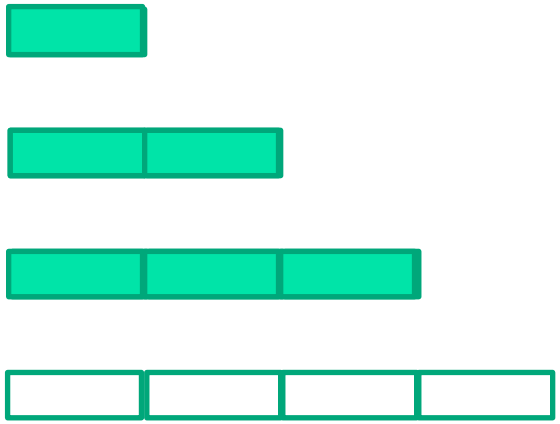


Figura 6-37. Ejemplo del algoritmo de congestión de Internet.

Algoritmo de Internet

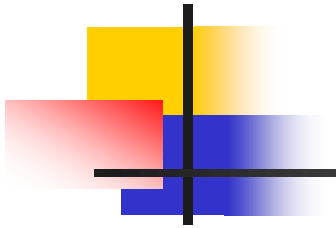
Tamaño de la ventana de congestión del emisor

Segmento confirmado



Algoritmo de Internet

- Se usa el parámetro umbral, inicialmente de 32 KB, además de la ventana del receptor y de la ventana de congestión
- Si expira el temporizador, se fija el umbral a la mitad de la ventana de congestión actual, y la ventana de congestión se restablece a un segmento
- Luego se usa el *algoritmo de arranque lento*
- Pero, el crecimiento exponencial termina al alcanzar el umbral
- Luego, las transmisiones exitosas aumentan linealmente la ventana de congestión; o sea, aumenta en un segmento por ráfaga, no en un segmento por segmento

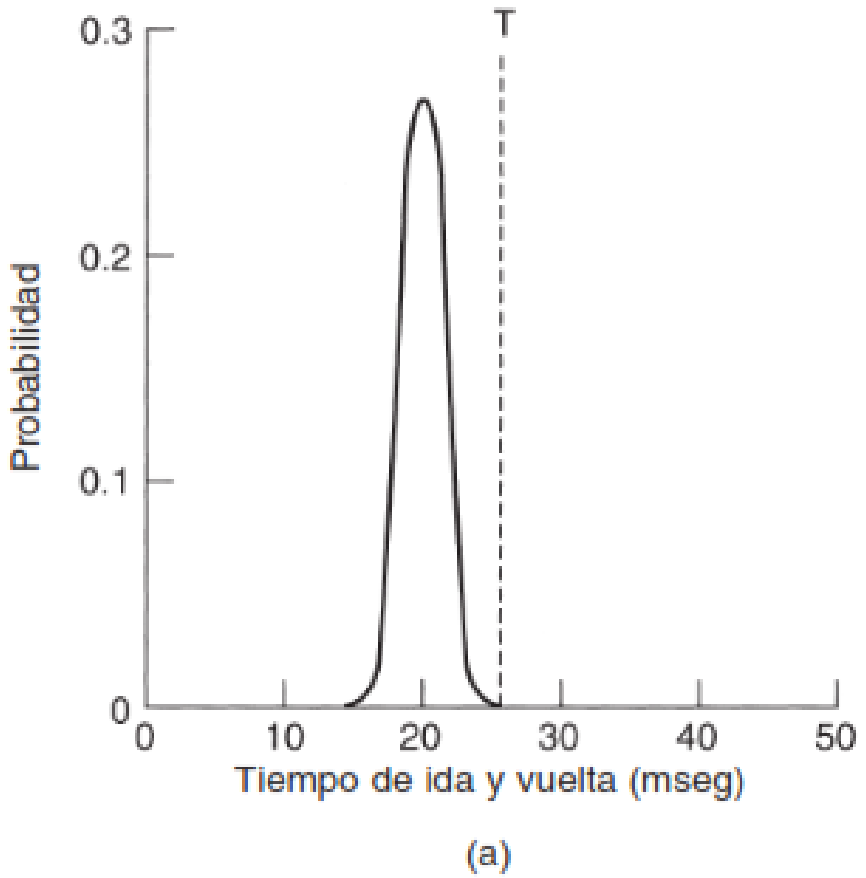


- Si no ocurren más expiraciones del temporizador, la ventana de congestión continuará creciendo hasta el tamaño de la ventana del receptor
- En este punto, dejará de crecer y permanecerá constante mientras no ocurran más expiraciones del temporizador

6.4.10 Administración de temporizadores del TCP

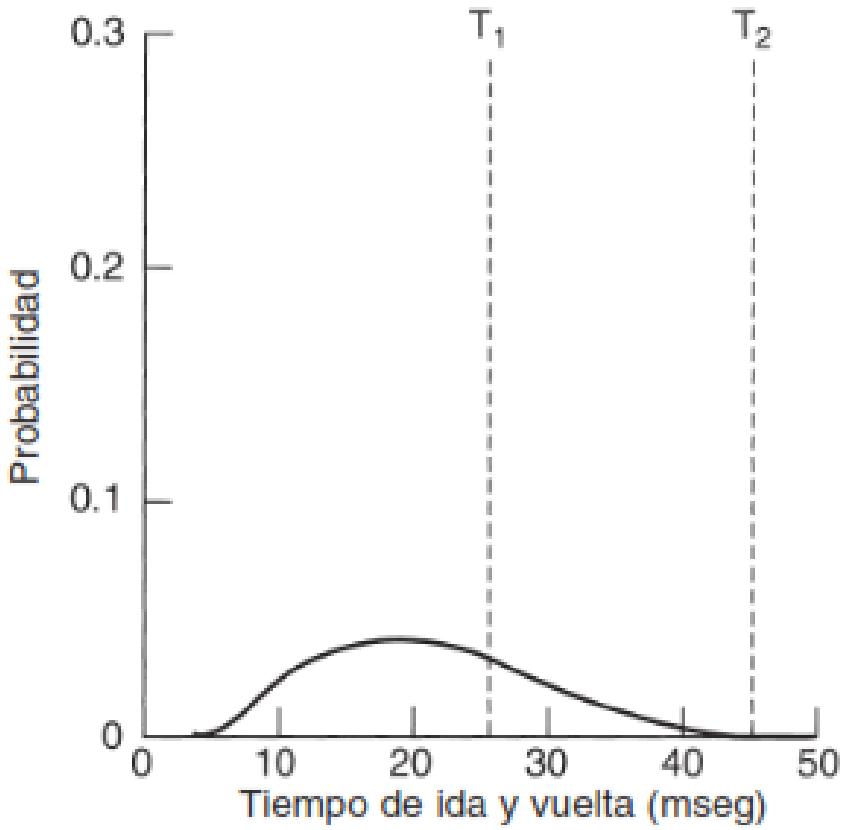
- Al enviarse un segmento se inicia el temporizador de retransmisiones
- Si la confirmación de recepción del segmento llega antes de expirar el temporizador, este se detiene
- Si el temporizador termina antes de llegar la confirmación de recepción, se retransmite el segmento y se reinicia el temporizador
- ¿Qué tanto debe durar el temporizador?
- Esta duración es mucho más predecible en el capa 2 que en la capa 4. Fig. 6.38(a)

Probabilidad del tiempo de llegada de la confirmación en la capa 2 vs el temporizador T

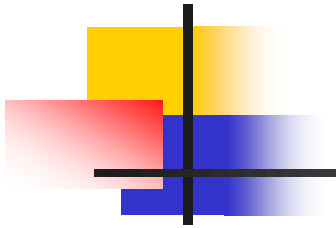


(a)

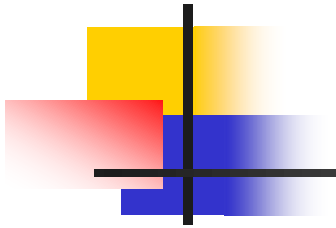
Probabilidad del tiempo de llegada de la confirmación en la capa 4 vs el el temporizador T1 o T2



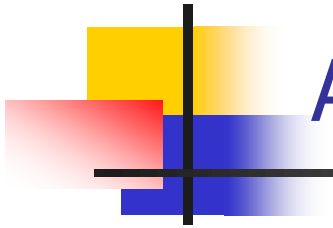
(b)



- En la capa 2 los ACKs pocas veces se retrasan
- La ausencia de ACKs se debe a la pérdida de esta o de la trama
- En TCP, es difícil la determinación del tiempo de ida y vuelta
- Si el intervalo de expiración es corto (T1) ocurren retransmisiones innecesarias, sobrecargando la red innecesariamente
- Si es largo (T2), el desempeño disminuye debido al mayor retardo de retransmisión de cada paquete perdido
- Por otro lado, el tiempo de ida y vuelta puede variar muy rápidamente



- La solución es usar un algoritmo que ajuste dinámicamente el intervalo de expiración del temporizador
- Esto se logra con base en mediciones continuas del desempeño de la red
- El algoritmo que usa TCP es de Jacobson (1988)

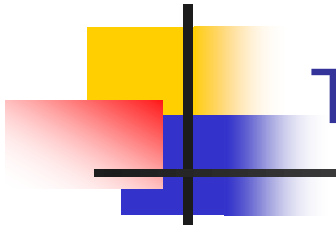


Algoritmo de Jacobson

- TCP utiliza cuatro temporizadores diferentes:
  - De retransmisión
  - De persistencia
  - De seguir con vida
  - De espera cronometrada antes del cierre de una conexión

Temporizador de retransmisión

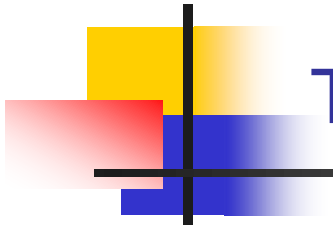
- Este temporizador se fija con base en $2RTT$ (Round-Trip Time)
- Cada conexión maneja la variable RTT, que es la mejor estimación actual del tiempo de ida del segmento y vuelta al origen de ACK
- Cada vez que se envía un segmento se inicia un temporizador de retransmisión
- Si ACK tarda un tiempo M, antes de expirar el temporizador, TCP actualiza RTT
- $RTT = \alpha RTT + (1 - \alpha)M$
- $\alpha = 7/8$ es el peso que se da al RTT anterior
- $1 - \alpha = 1/8$. Se da más peso a RTT anterior que a M
- Si el temporizador expira, este se duplica y RTT no se actualiza



Temporizador de persistencia

- Este es diseñado para evitar el bloqueo
- El receptor envía un ACK, indicando al emisor que no transmita (ventana 0). Esta indicación llega al emisor, y este inicia el temporizador de persistencia
- Luego el receptor actualiza la ventana, pero se pierde esta información
- Ahora el emisor y receptor esperan que el otro haga algo
- Cuando termina el temporizador de persistencia el emisor envía un sondeo al receptor

- El receptor responde al sondeo con el tamaño de la ventana
- Si la ventana sigue en 0, se inicia otra vez el temporizador de persistencia
- Si la ventana es diferente de cero el emisor puede enviar datos



Temporizador de seguir con vida

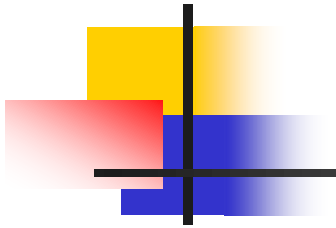
- Si una conexión está inactiva por mucho tiempo, este temporizador expira
- El un lado de la conexión sondea que el otro lado aun esté allí
- Si no recibe respuesta, se termina la conexión

Temporizador de espera cronometrada

- Este temporizador opera en el cierre de una conexión
- Tiene una duración de 2TTL Time-To-Live del paquete
- Así se asegura que todos los paquetes lleguen antes del cierre

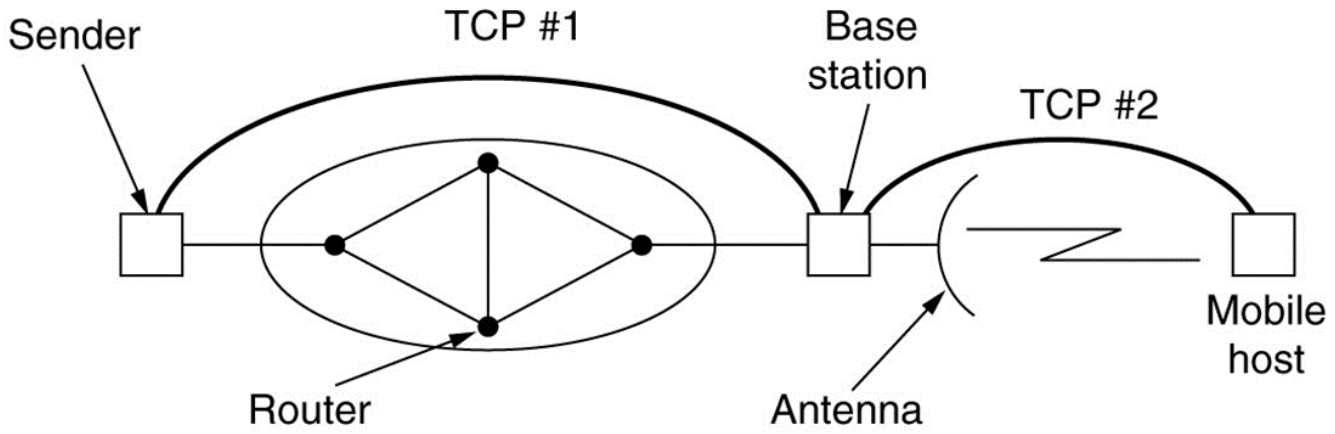
6.4.11 TCP y UDP inalámbricos

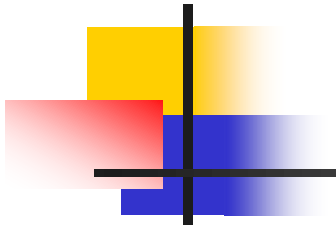
- Los protocolos de transporte deberían ser independientes de la capa física
- TCP no debería preocuparse si se opera sobre cobre, fibra o radio
- Pero en la práctica sí importa, ya que TCP es optimizada para redes cableadas, no para redes inalámbricas
- El problema es el algoritmo de control de congestión
- TCP supone que las expiraciones del temporizador de retransmisión ocurren por congestión de la red, no por paquetes perdidos
- Por tanto, TCP disminuye su velocidad (arranque lento) para reducir la carga a la red y aliviar la congestión



- Pero los enlaces inalámbricos pierden paquetes, no por congestionamiento
- En este caso los paquetes perdidos se deben reenviar inmediatamente
- Frente a un paquete perdido por congestionamiento en una red cableada, el emisor debe reducir la velocidad
- En redes inalámbricas la reducción de la velocidad disminuye su desempeño. El emisor debe al menos mantener la velocidad, no reducirla
- Si el emisor no sabe si la red es cableada o inalámbrica, no se puede tomar la decisión correcta

- Con frecuencia, el camino desde el emisor al receptor no es homogéneo
- La solución es dividir la conexión TCP en conexiones distintas





- La ventaja de este esquema es que ahora ambas conexiones son homogéneas
- Las expiraciones del temporizador en la red 1 reducen la velocidad del emisor
- Las expiraciones del temporizador en la red 2 aceleran la velocidad
- La desventaja: viola la semántica de TCP: conexión extremo a extremo