Tema 3: Nivel de transporte

3.1 Introducción

Servicios

- Proporciona una comunicación lógica entre procesos de aplicación que se ejecutan en hosts diferentes
- Proporciona un servicio de transferencia de datos (comunicación de procesos) extremo a extremo que aísla a las capas superiores de los detalles de la red.
- Responsable de la entrega de un mensaje entero entre procesos.
- Asegura que el mensaje completo llega intacto y en orden.
- Está presente en los sistemas finales pero no en los dispositivos de interconexión
  - ✓ Lado emisor: divide los mensajes de aplicación en segmentos y se pasan al nivel de red
  - ✓ Lado receptor: reensambla los segmentos en mensajes y se pasan al nivel de aplicación

Existen dos tipos de **protocolos de transporte**:

UDP, trata cada segmento de forma independiente

TCP, usa números de secuencia para relacionar los segmentos

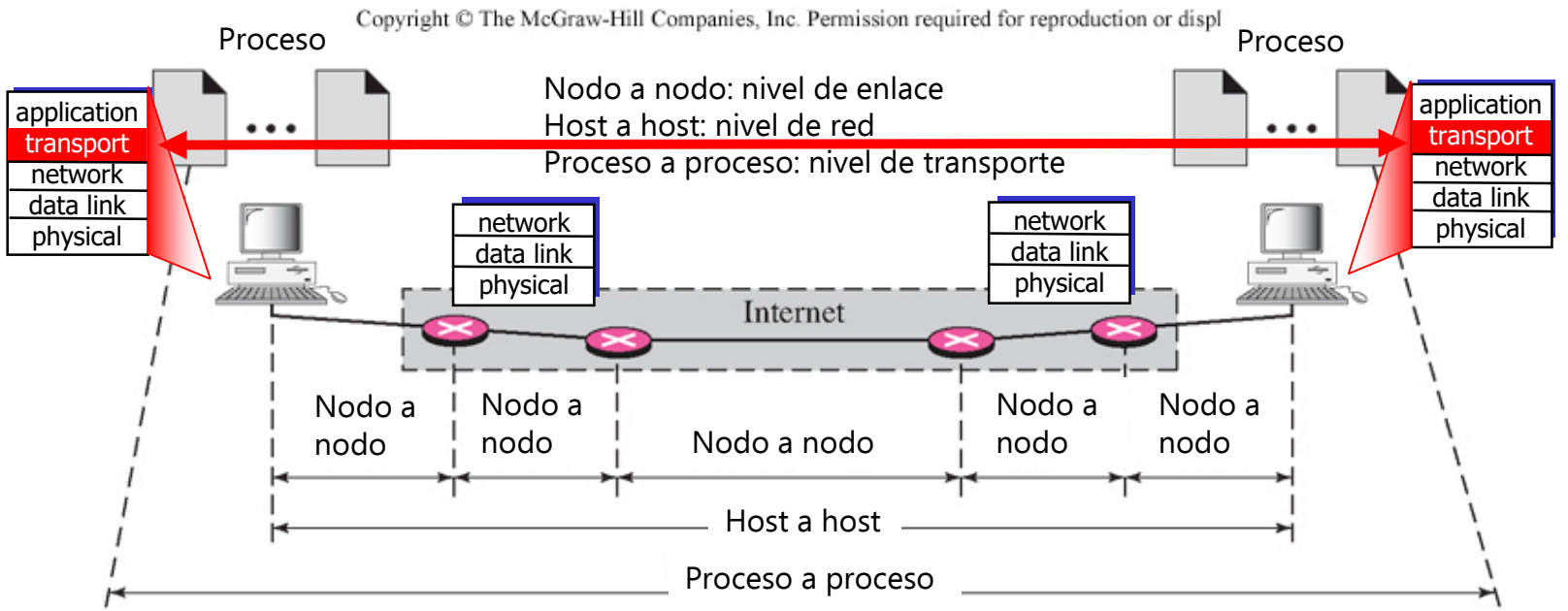
Funciones

- Las principales **funciones** que se realizan a nivel de transporte son:
  - ✓ **transporte de la información** desde el origen al destino de forma segura o no segura con independencia de la red física
  - ✓ **oferta de servicios a la capa de aplicación**: tanto orientados a conexión como no orientados a conexión
  - ✓ **recuperación de los fallos** de los niveles inferiores e informe de los mismos a las capas superiores: Calidad de Servicio
  - ✓ se comporta como un **interface** entre un proveedor de servicios seguros y el usuario
  - ✓ **complementa al nivel de red**: comprobación integridad datos (incluye campo comprobación errores en la cabecera)
    - Protocolo IP proporciona servicio no fiable
      - ➢ Servicio best-effort (mejor entrega), pero no la garantiza (ni la entrega de segmentos, ni en orden, ni la integridad

- TCP
  - ✓ Fiable, entrega en orden
    - Técnicas de control de flujo, números secuencia, mensajes confirmación y temporizadores
  - ✓ Control de congestión (servicio que se presta a la red)
  - ✓ Orientado a conexión
- UDP
  - ✓ No fiable, no entrega en orden
  - ✓ No orientado a conexión

3.1. Introducción

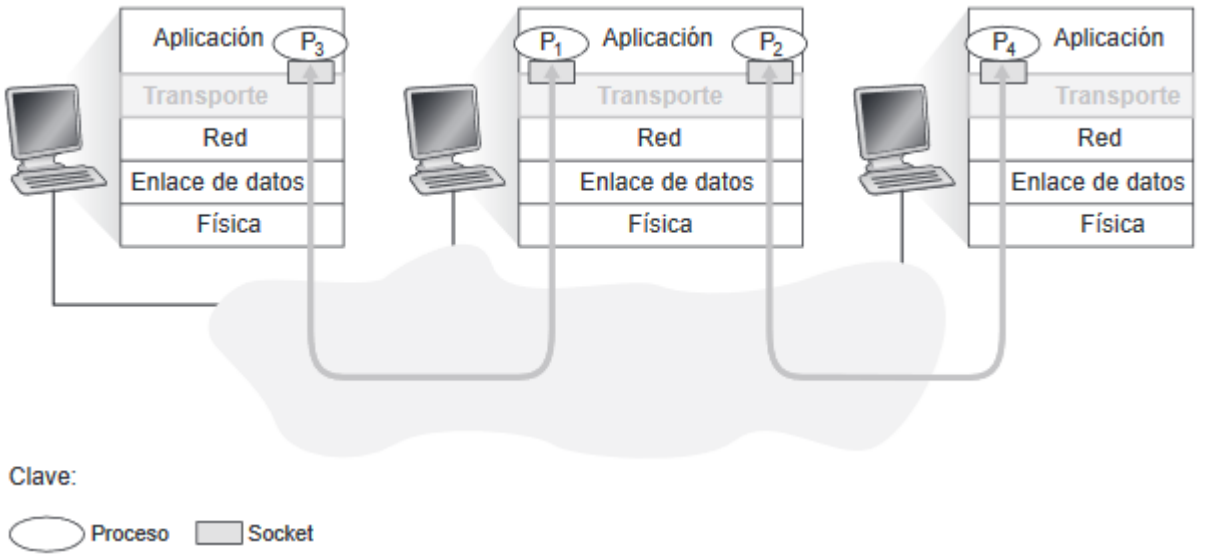
- Nivel red: comunicación lógica entre hosts
- Nivel de transporte: comunicación lógica entre procesos



- Extender la entrega entre host a una entrega entre procesos -> multiplexación y demultiplexación

3.2 Comunicación entre procesos

- La capa de transporte debe entregar los datos contenidos en los segmentos recibidos desde la capa de red a la aplicación correspondiente
- Un proceso de aplicación -> uno o varios sockets -> cada socket identificador único



(Fuente: Redes de computadoras. Un enfoque descendente 7ED. James F. Kurose, Keith W. Rose. 2017)

Multiplexación / Demultiplexación

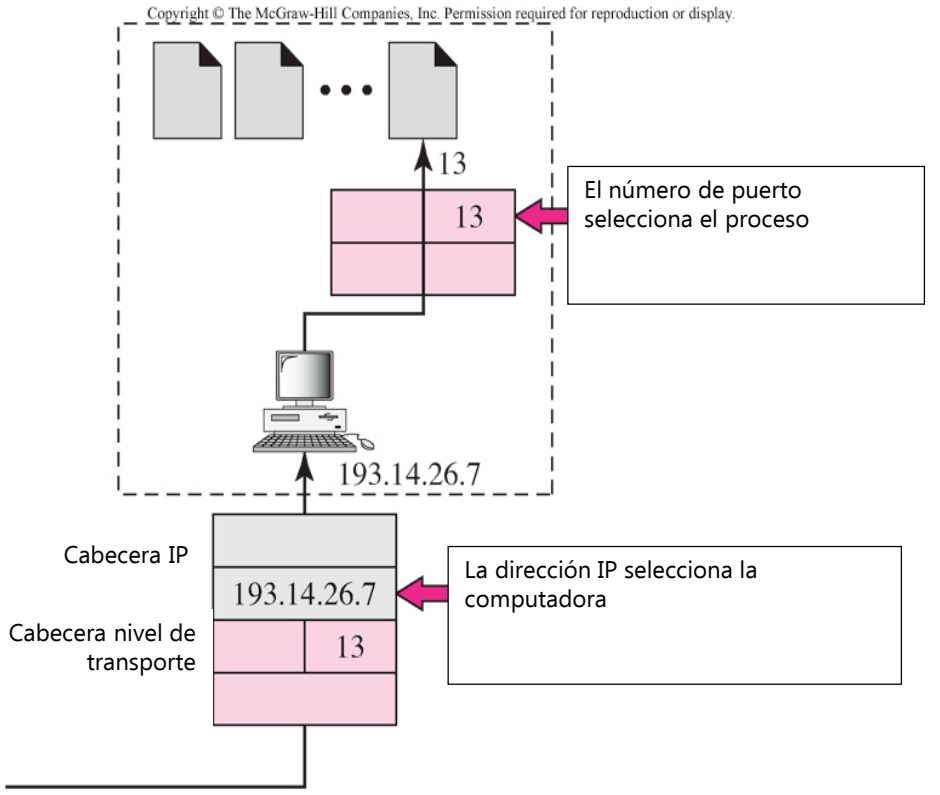
- Multiplexación: (en el emisor)
  - ✓ Varios procesos necesitan enviar segmentos, pero hay un único nivel de transporte (relación muchos a uno)
  - ✓ cada fragmento de datos se encapsula con la información de cabecera para crear los segmentos y pasarlos a la capa de red
- Demultiplexación: (en el receptor)
  - ✓ el nivel de transporte entrega los datos contenidos en un segmento de la capa de transporte al proceso apropiado

La multiplexación en un host requiere:

1. Que los sockets tengan identificadores únicos
2. Que cada segmento tenga campos especiales (puerto origen y puerto destino) que indiquen el socket al que tiene que entregarse el segmento

Direccionamiento

- Número de puerto → 16 bits (0-65535)
  - ✓ Rangos:
    - Puertos bien conocidos (servidor) : 0-1023, uso restringido (80 HTTP)
    - Puertos registrados: 1024-49151, reservados para aplicaciones concretas (3306 MySQL)
    - Puertos privados: 49152-65535 no controlados ni registrados (efímeros)
- Socket UDP (sin conexión)
  - ✓ IP destino + puerto destino
  - ✓ Dos segmentos con distinto origen se entregan al mismo socket destino
- Socket TCP (con conexión)
  - ✓ IP origen/puerto origen + IP destino/puerto destino
  - ✓ Dos segmentos con distinto origen se entregan a dos sockets distintos

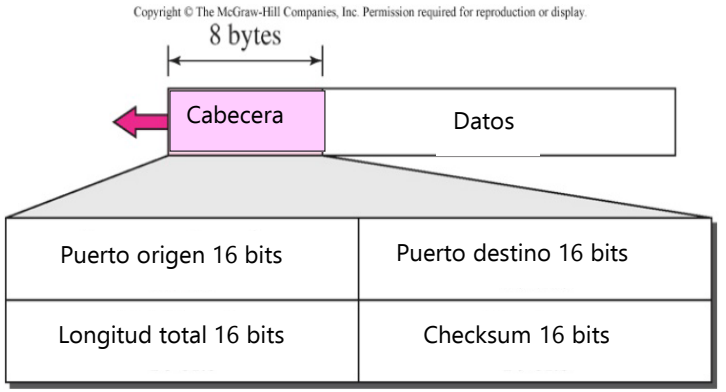


3.3. Protocolo UDP

- **User Datagram Protocol**
- Proporciona un servicio sin conexión (cada segmento se gestiona independientemente) y no fiable a los procesos de la capa de aplicación
- Simple y poco sofisticado
  - ✓ Multiplexación/demultiplexación
  - ✓ el mecanismo de control de error es la suma de comprobación
    - ➢ no detecta mensajes perdidos ni duplicados
    - ➢ si el receptor detecta un error descarta el datagrama
- Protocolo *no seguro:*
  - ✓ no garantiza entregas
  - ✓ no controla los duplicados
  - ✓ no realiza control de flujo
- **Ventaja**: reduce considerablemente la información suplementaria de la red
- **Segmento UDP**: 8 bytes de cabecera más datos

3.3. Protocolo UDP

- **Cabecera UDP**
  - ✓ **número de puerto**: identifican el proceso en la máquina origen y en destino
  - ✓ **longitud:** incluye cabecera (8 bytes) y datos
  - ✓ **suma de comprobación** (opcional): detecta errores en el datagrama de usuario
    - si no se usa: todos los bits a 0
    - si se usa, emplea el mismo algoritmo que TCP e IP



3.3. Protocolo UDP

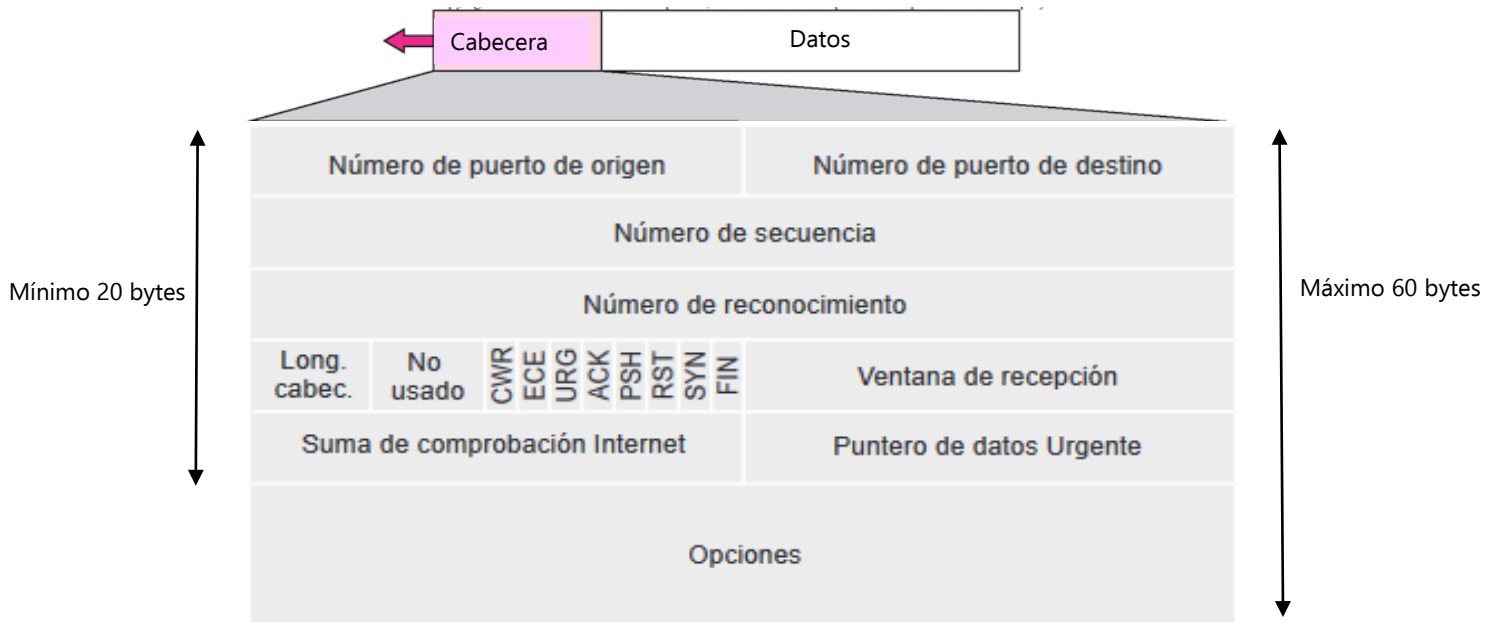
- **Uso UDP**
  - ✓ comunicación petición-respuesta sencilla, por ejemplo DNS
  - ✓ procesos con mecanismos internos de control de flujo y error, por ejemplo TFTP
  - ✓ procesos de gestión, por ejemplo SNMP
  - ✓ para algunos protocolos de actualización de ruta, por ejemplo RIP
  - ✓ Aplicaciones streaming multimedia
- *¿Porqué usar UDP?*
  - ✓ mejor control en el nivel de aplicación sobre qué datos se envían y cuando
  - ✓ sin establecimiento de conexión
  - ✓ Sin información de estado de conexión
  - ✓ Baja sobrecarga de control

3.4. Protocolo TCP

- **Transmission Control Protocol**
- Proporciona un servicio
  - ✓ orientado a conexión, antes de enviar los datos se establece la conexión lógica
  - ✓ fiable, usa mecanismos de confirmación para comprobar que los datos han llegado completamente y seguros
  - ✓ punto a punto, un emisor y un receptor (no acepta ni broadcast ni multicast)
  - ✓ full-dúplex, flujo de datos bidireccional en la misma conexión
  - ✓ control de flujo, el emisor no sobrecargue al receptor
  - ✓ transmisión de flujos (pipelined)
    - protocolo orientado al flujo
    - crea un entorno en el cual el proceso emisor y receptor parecen estar conectados por un “tubo” (pipe) imaginario que transporta sus datos a través de Internet.
    - se necesitan dos buffers de almacenamiento temporal
    - TCP agrupa los bytes en un paquete denominado segmento para entregarlo al nivel IP para su transmisión.

3.4. Protocolo TCP

Estructura del segmento



- **puertos** (16 bits): identifican los puntos terminales de la conexión
- **número de secuencia** (32 bits): número asignado al primer byte de datos que contiene el segmento
- **número de confirmación** (32 bits): número de byte que el receptor del segmento espera recibir
- **longitud cabecera** (4 bits): número de palabras de 32 bits incluidas en la cabecera
- **no usado** (4 bits): para usos futuros

Estructura del segmento

- **Indicadores** (8 bits): uno o más de estos bits pueden activarse al mismo tiempo; permiten el control de flujo, control de congestión, establecer y terminar la conexión, etc.
  - ✓ CWR: indica ventana de congestión reducida
  - ✓ ECE: indica congestión
  - ✓ URG: indica que a continuación vienen datos urgentes
  - ✓ ACK: indica reconocimiento válido
  - ✓ PSH: indica el uso de la función PUSH, entrega inmediata (función de carga)
  - ✓ RST: resetear la conexión, empezar de nuevo
  - ✓ SYN: sincronizar números de secuencia para establecer conexiones
  - ✓ FIN: liberar una conexión
- **Ventana de recepción** (16 bits): para control de flujo: bits que se pueden recibir a partir del último acuse de recibo enviado
- **suma de comprobación** (16 bits): igual que en el protocolo UDP y también el protocolo IP
- **puntero de datos Urgente** (16 bits): válido cuando el puntero urgente está activo, apunta al comienzo de los datos urgentes en el segmento
- **opciones** (variable): hasta 40 bytes de información opcional en la cabecera

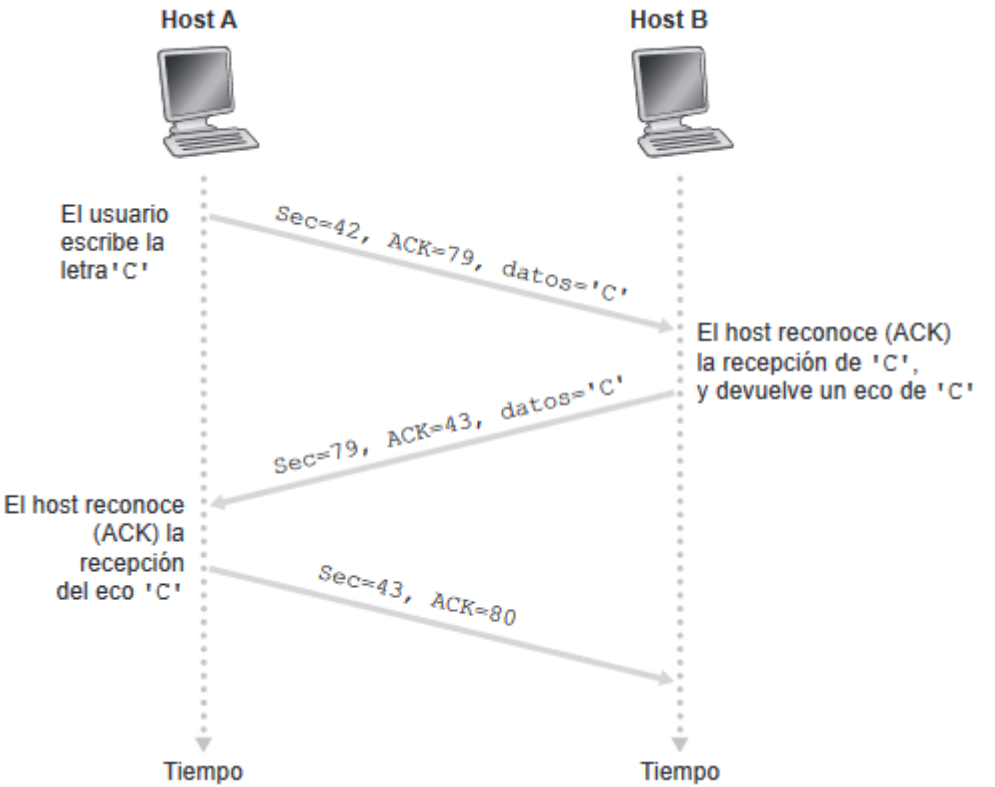
Estructura del segmento

Números de secuencia y ACKs, escenario sencillo:

El Host A envía una letra al Host B que este devuelve como eco (longitud de datos = 1 byte).

La transmisión ya está iniciada y en este momento el **Host A** envía un segmento TCP con número de sec = 42 y número de confirmación = 79. Esto significa que en ese segmento se está confirmando un segmento anterior enviado por el Host B cuyo número de secuencia era 78 y además indica que el siguiente que espera recibir es el 79.

A continuación el **Host B** envía un segmento con el número de secuencia = 79 y el número de confirmación del segmento enviado por Host A = 43 (confirma el segmento 42 y espera recibir el 43).



(Fuente: Redes de computadoras. Un enfoque descendente 7ED. James F. Kurose, Keith W. Rose. 2017)

Control de la conexión

Para el control de la conexión, el intercambio de información tiene tres fases:

- ✓ *Establecimiento de conexión*
  - Sincronizar número de secuencia y reservar recursos
  - Negociación en tres pasos:
    - ➢ El programa servidor le dice a su TCP que está listo para aceptar una conexión (*apertura pasiva*)
    - ➢ El programa cliente le dice a su TCP que quiere conectarse a un servidor en particular (*apertura activa*)
    - ➢ TCP comienza la negociación en tres pasos:
      - el cliente envía el primer segmento, con SYN = 1, para sincronizar los números de secuencia
      - el servidor envía un segundo segmento, con SYN=1 y ACK=1, para sincronizar los números de secuencia en dirección contraria y confirmar el SYN anterior
      - el cliente envía un tercer segmento, ACK=1, confirmando la recepción del segundo segmento.

Control de la conexión

✓ *Transferencia de datos*

- TCP transmite los datos en modo full-duplex
- Tanto emisor como receptor pueden enviar datos y confirmaciones.
- La confirmación se incluye con los datos.
- Entrega inmediata de datos, PSH=1, el emisor no espera a rellenar su ventana, el segmento se envía inmediatamente.
- Datos urgentes, URG=1, cuando TCP crea un segmento inserta los datos urgentes al principio del segmento. El resto del segmento puede contener datos normales. El campo puntero urgente de la cabecera define el final de los datos urgentes y el comienzo de los datos normales

Control de la conexión

✓ *Fin de la conexión*

- Cualquiera de los dos procesos puede cerrar la conexión (normalmente el cliente)
- Negociación en tres pasos:
  - el TCP cliente envía el primer segmento, FIN=1 (puede incluir la última porción de datos enviados por el cliente o puede ser únicamente de control)
  - el TCP servidor después de recibir el segmento FIN, informa a su proceso y envía un segundo segmento, FIN=1 y ACK=1, confirmando el anterior
  - el cliente TCP envía el último segmento, ACK=1, confirmando la recepción del segundo
- Semicierre:
  - un extremo deja de enviar datos mientras sigue recibiendo

Control de la conexión

- Recuperación ante Caídas/Fallos:
  - ✓ ante un fallo en una conexión es necesario reactivar dicha conexión
  - ✓ puede ocurrir que al fallar una conexión, uno de los extremos permanezca activo
  - ✓ *soluciones:*
    - temporizador de renuncia: tiempo máximo de espera de mensaje
    - segmento RST
  - ✓ aún así persiste el *problema de la sincronización*, las entidades de transporte pueden no saber:
    - cual es el último segmento aceptado
    - cual es la última confirmación recibida

Transferencia de datos fiable. Control de errores

Garantiza que el flujo de datos que un proceso extrae de su buffer de recepción TCP no está corrupto, no contiene huecos, ni duplicados y está en orden

Tres mecanismos de detección y corrección

- ✓ *sumas de comprobación (detección)*
  - cada segmento incluye un campo de suma de comprobación que sirve para comprobar si el segmento está corrupto → lo descarta el TCP destino
- ✓ *confirmación*
  - usa confirmaciones (ACK) para indicar la recepción de segmentos de datos
- ✓ *retransmisión*
  - cuando un segmento está corrupto, se pierde o se retrasa, se retransmite
  - en las implementaciones modernas un segmento se retransmite cuando:
    - ➢ expira un temporizador de retransmisión
    - ➢ cuando el emisor recibe tres ACK duplicados

Transferencia de datos fiable. Control de errores

TCP crea un servicio de transferencia de datos fiable sobre un servicio no fiable de IP

- ✓ segmentos transmitidos por un “tubo” (conexión lógica)
- ✓ ACKs acumulativos
- ✓ temporizador de retransmisión individual

Campos involucrados:

- ✓ Secuencia: offset (en bytes) dentro del mensaje que indica el primer octeto del campo de datos. Permite la entrega ordenada
- ✓ Confirmación: número de byte esperado en el receptor. Tiene carácter acumulativo
- ✓ Bit ACK del campo de control. Indica la validez o no de la confirmación de acuse
- ✓ Suma de comprobación (cheksum) de todo el segmento

Transferencia de datos fiable. Control de errores

Durante el proceso de transmisión/retransmisión en el emisor TCP, gestiona

- Aceptación de los datos de procesos de usuario (aplicación):
  - ✓ los *fragmenta* en segmentos menores de 64k (MTU estándar 1500 bytes)
    - crea segmento con un número de secuencia (seq#)
    - seq# es el número de flujo de bytes del primer byte de datos en el segmento
  - ✓ los *envía* como datagramas IP
  - ✓ inicia temporizador (timeout) si aún no se está ejecutando (iniciado por otro segmento)
- Fin temporización
  - ✓ Retransmisión del segmento
  - ✓ Reinicio del temporizador
- Recepción ACK
  - ✓ TCP utiliza ACKs acumulativos (confirma todos los segmentos anteriores no confirmados)

Estimación timeout:

- ✓ Mayor que el tiempo de ida y vuelta
- ✓ demasiado pequeño -> timeout prematuro
- ✓ Demasiado grande -> Reacción lenta ante pérdida de segmentos
- ✓ Solución adaptable: según el emisor estime la velocidad de la red

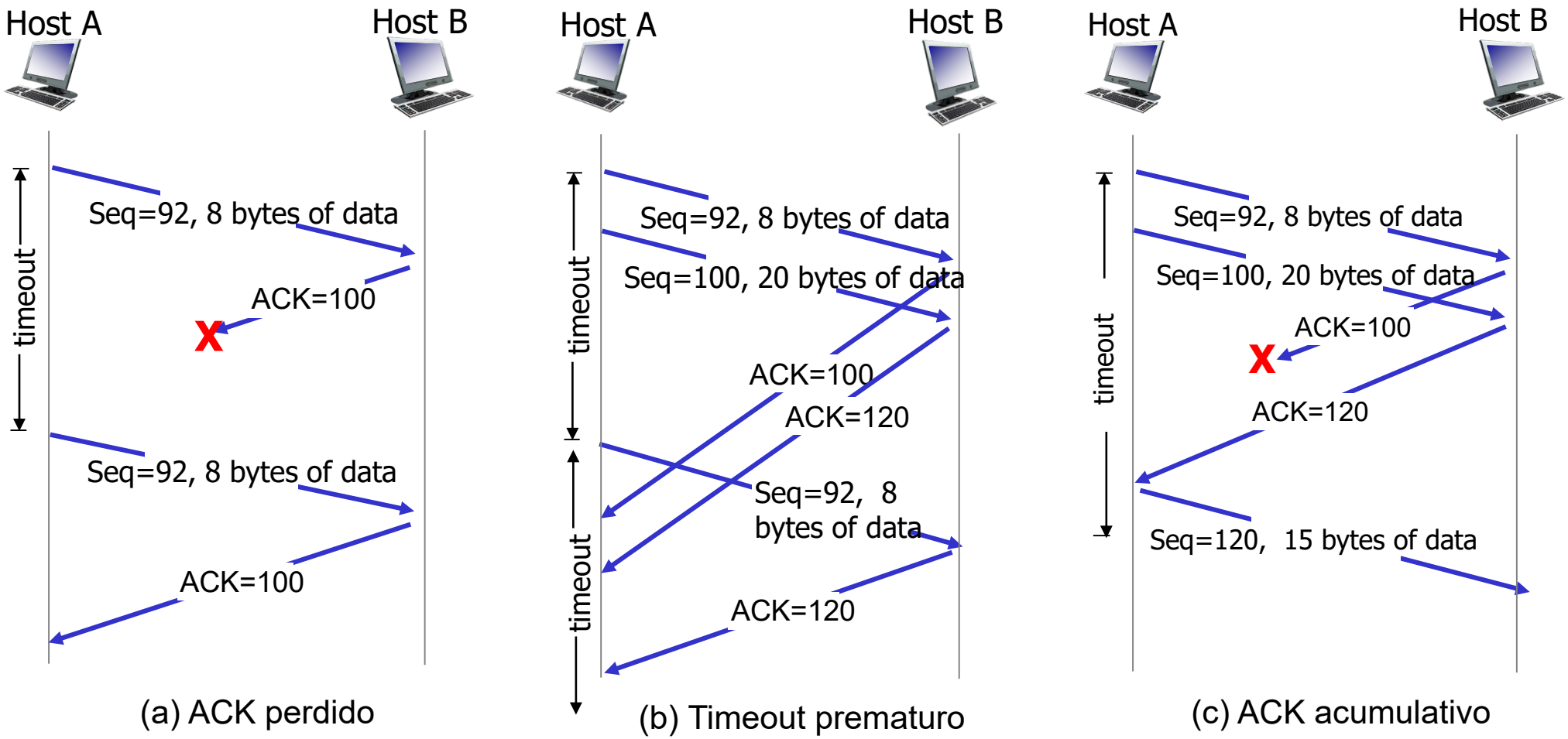
Si expira el temporizador:

- ✓ retransmisión del segmento que causó el timeout
- ✓ reiniciar temporizador

Confirmación recepción:

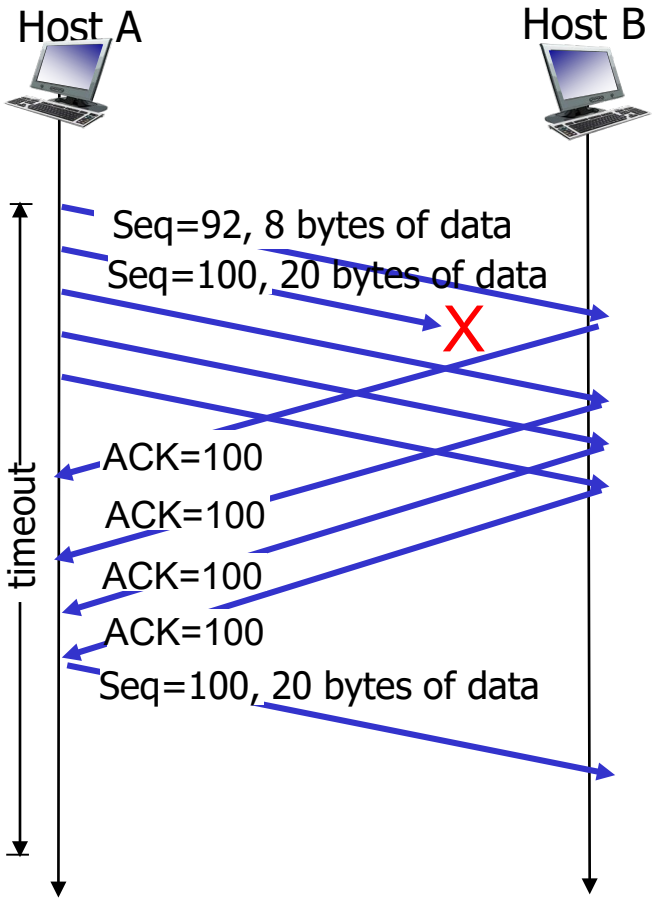
- ✓ si el ACK confirma segmentos no confirmados previamente
  - actualizar lo que ha sido confirmado
  - iniciar temporizador si todavía hay segmentos sin confirmar

Escenarios de retransmisión



Retransmisión rápida después que el emisor recibe tres ACK duplicados

- Período timeout puede ser relativamente largo:
  - ✓ Provoca grandes retrasos antes de reenviar el paquete perdido
- Solución: detectar segmentos perdidos a través de ACK duplicados.
  - ✓ el remitente, normalmente, envía muchos segmentos seguidos
  - ✓ Si se pierde un segmento, genera un ACK duplicado.
- Retransmisión rápida TCP
  - ✓ si el remitente recibe 3 ACK por los mismos datos
  - ✓ reenvíe el segmento no confirmado con el número de secuencia más pequeño
  - ✓ es probable que se descarte el segmento no confirmado, así que no espere el tiempo de espera



**En la gráfica, B recibe un segmento con un número de secuencia mayor que el esperado -> detecta falta de un segmento -> Se genera un ACK duplicado. Cuando se envían muchos segmentos seguidos, se generaran muchos ACK duplicados. Si recibe 3 para los mismos datos se hace retransmisión rápida*

Control de flujo

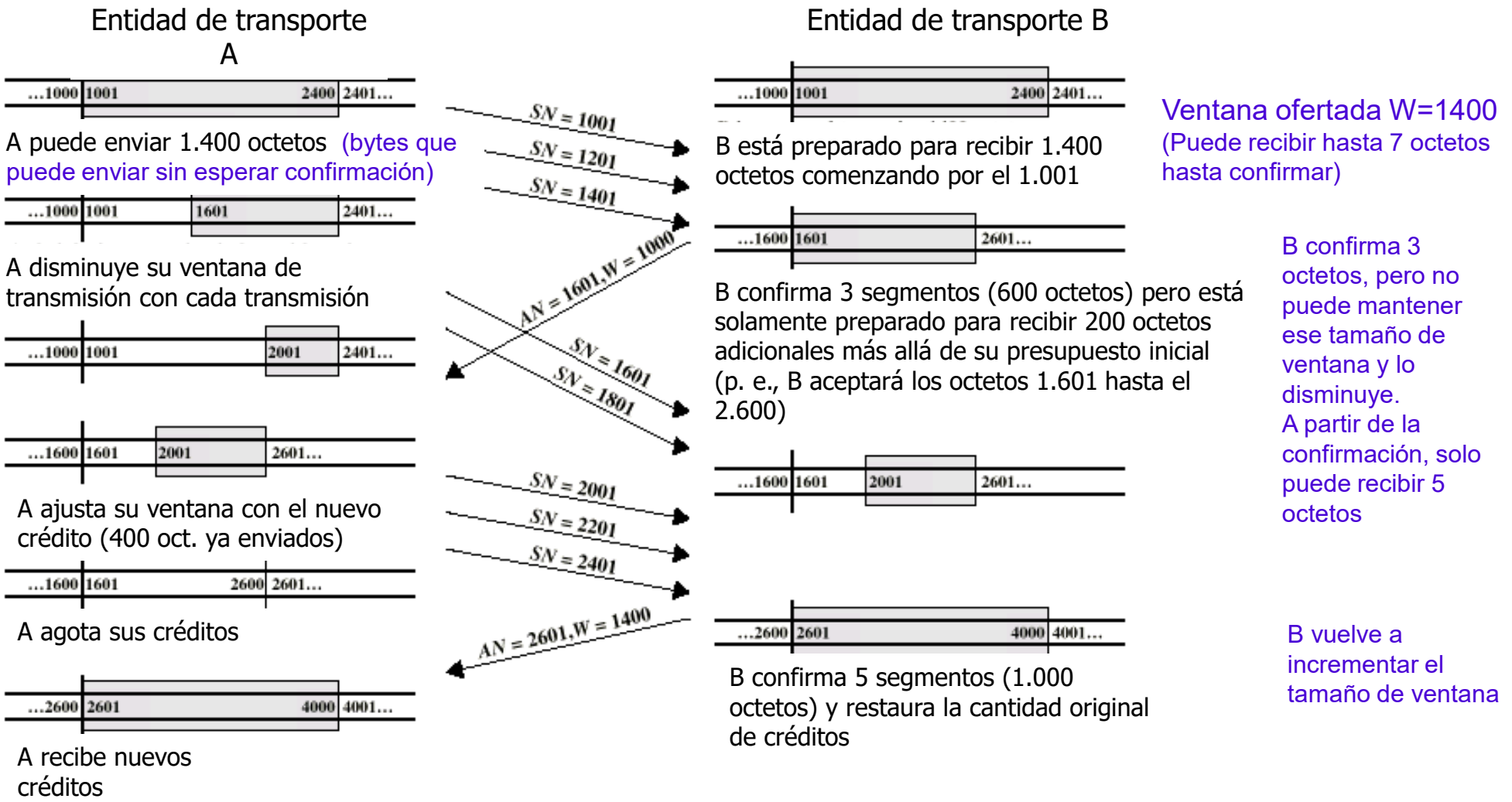
- Evitar que el emisor sature al receptor
- Posibles **soluciones**:
  - ✓ **descartar** segmentos ante una congestión
  - ✓ mecanismos de **ventana deslizante**: envío de n segmentos sin esperar confirmación
    - sistema de créditos**:** cada mensaje de confirmación indica el número del byte a recibir así como el umbral de la nueva ventana (créditos)
- TCP proporciona un servicio de control de flujo mediante un sistema de créditos por el que el receptor avisa al emisor del espacio disponible en su buffer (ventana de recepción), lo que puede aceptar
  - ✓ Utiliza el campo “ventana” para establecer la ventana ofertada (en bytes)
    - El receptor especifica al emisor un tamaño de ventana máximo autorizado para transmitir (ventana ofertada)
    - El emisor transmite los datos de acuerdo con lo que se denomina ventana útil (ventana ofertada – bytes en tránsito)
  - ✓ Una ventana de recepción en cada lado (full-dúplex)
  - ✓ La ventana de recepción es una variable dinámica

3.4. Protocolo TCP

Control de flujo

Ventana útil

A medida que envía bytes quedan menos antes de que los confirme B



Control de congestión

- Un aspecto importante de una red de conmutación de paquetes es la congestión. Esta puede ocurrir si la carga de la red es mayor que su capacidad
- El control de congestión se refiere a los mecanismos y técnicas que permiten controlar la congestión y mantener la carga por debajo de esa capacidad.
- La congestión se produce en el router (un buffer para cada enlace) y tiene costes:
  - ✓ Buffer infinito -> grandes retardos cuando la tasa de llegada de los paquetes se aproxima a la capacidad del enlace
  - ✓ Buffer finito -> el emisor realiza retransmisiones para compensar los paquetes descartados por desbordamiento del buffer o innecesarias si está en la cola pero su temporizador finaliza
  - ✓ Descarte de paquete a lo largo de la ruta -> se desperdicia a capacidad de transmisión empleada en cada uno de los enlaces anteriores para encaminar dicho paquete hasta el punto de la red donde se descarta

Control de congestión

- Mecanismos de control de congestión
  - ✓ Terminal a terminal
    - La capa de red no proporciona soporte explícito a la capa de transporte
    - La congestión la deducen los sistemas terminales en función de paquetes perdidos y retardos
      - ➢ Aplicado por TCP: pérdida de segmentos TCP -> congestión -> TCP reduce ventana
  - ✓ Asistido por la red -> Notificación explicita de congestión
    - La capa de red (routers) proporciona información explícita a los sistemas terminales del estado de conexión de la red a través de TCP (extensiones en encabezados IP y TCP)
    - La información de congestión se realimenta de la red al emisor
      - ➢ El router marca un campo (campo ToS) en el encabezado IP para indicar congestión al receptor.
      - ➢ El receptor activa los bits de congestión en el encabezado TCP para notificar al emisor de la congestión

Control de congestión

- Mecanismo de control de congestión que usa TCP
  - ✓ Cada emisor limita la velocidad de transmisión en función de la congestión de red percibida.
  - ✓ ¿Cómo se limita la velocidad? Ventana de congestión
  - ✓ Detección de la congestión: fin temporizador o 3 ACKs duplicados
  - ✓ Si no existe congestión: TCP usa los paquetes ACK para incrementar el tamaño de ventana
    - Consecuencia: si todos los emisores hacen lo mismo, la red colapsa
  - ✓ Para determinar el tamaño de la ventana de congestión, sin colapsar la red, TCP se basa en los siguientes principios:
    - Un segmento perdido implica congestión -> reducir velocidad del emisor TCP
    - Un segmento reconocido -> segmento entregado, incrementar velocidad transmisión en ACK
    - Tanteo del ancho de banda -> incrementar la velocidad hasta que hay una pérdida, entonces reducir velocidad de transmisión

Control de congestión

- Algoritmo de control de congestión que usa TCP, tres fases
  - ✓ arranque lento: inicio conexión tamaño de ventana de congestión pequeño que se incrementa exponencialmente hasta que alcanza un umbral (se detecta congestión)
  - ✓ evitación de la congestión: el tamaño de ventana de congestión se incrementa aditivamente (desde la mitad del valor que tenía cuando se detectó congestión) hasta que se detecta la congestión
    - Por fin de temporización: el tamaño vuelve al inicio y el umbral es la mitad del tamaño de la ventana antes de la congestión
    - Por pérdida de paquete: el tamaño de la ventana de congestión se divide a la mitad y el umbral es el tamaño de la ventana antes de la congestión
  - ✓ recuperación rápida: se va incrementando el tamaño de la ventana como en la evitación hasta que llega el ACK para el segmento que falta, entra de nuevo en estado de evitación