TEMA 3
CAPA DE TRANSPORTE EN INTERNET

Fundamentos de Redes

2016/2017

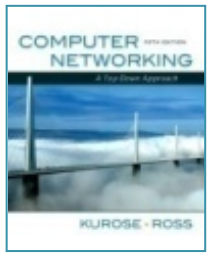
GRUPO A

Sandra Sendra Compte (ssendra@ugr.es)

- Bibliografía Básica:

Capítulo 10, Pedro García Teodoro, Jesús Díaz Verdejo y Juan Manuel López Soler. ***TRANSMISIÓN DE DATOS Y REDES DE COMPUTADORES***, Ed. Pearson, 2ª Ed. ,Pearson, 2014, ISBN: 978-0-273-76896-8

- Para saber más...



Capítulo 3 James F. Kurose y Keith W. Ross. ***COMPUTER NETWORKING. A TOP-DOWN APPROACH***, 5ª Edición, Addison-Wesley, 2010, ISBN: 9780136079675.

- Agradecimientos:

Transparencias originales de **Juan Manuel López Soler, Pedro García Teodoro, Jorge Navarro Ortiz**, Departamento TSTC, UGR.

- Para saber más...

Capítulo 17: William Stalling. ***COMUNICACIONES Y REDES DE COMPUTADORES***, 6ª Edición, PEARSON EDUCACION, 2000, ISBN: 9788420529868.

Tema 3. CAPA DE TRANSPORTE EN INTERNET

1. **Introducción a los protocolos de Capa de Transporte**
2. Protocolo de datagrama de usuario (UDP).
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
4. Control de congestión.
4. Extensiones TCP.
5. Ejercicios.

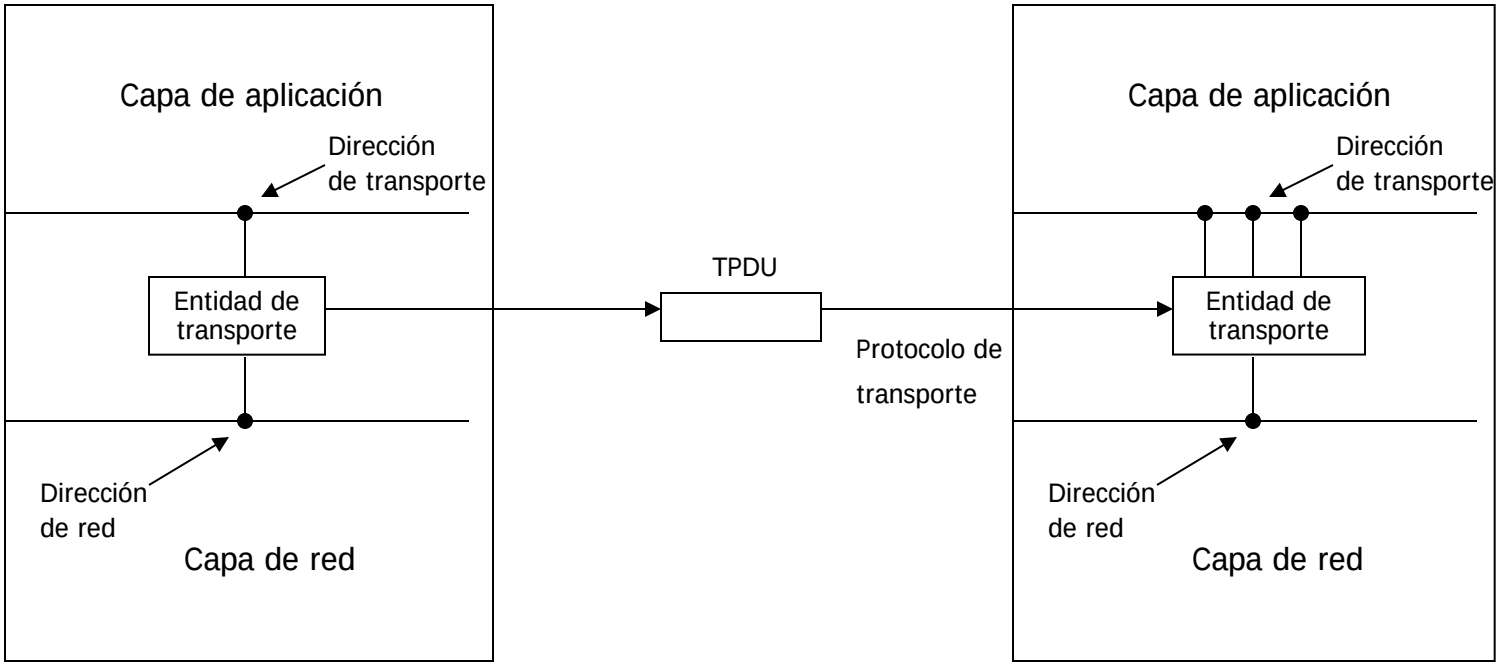
Capa de transporte

- Las redes de datos e Internet nos dan soporte para establecer una comunicación continua y confiable entre los equipos.
- En un único dispositivo, las personas pueden utilizar varios servicios (correo electrónico, acceso Web y mensajería instantánea, etc).
- Los datos de cada una de estas aplicaciones se empaqueta, se transporta y se entrega al servidor adecuado o aplicación en el dispositivo de destino.

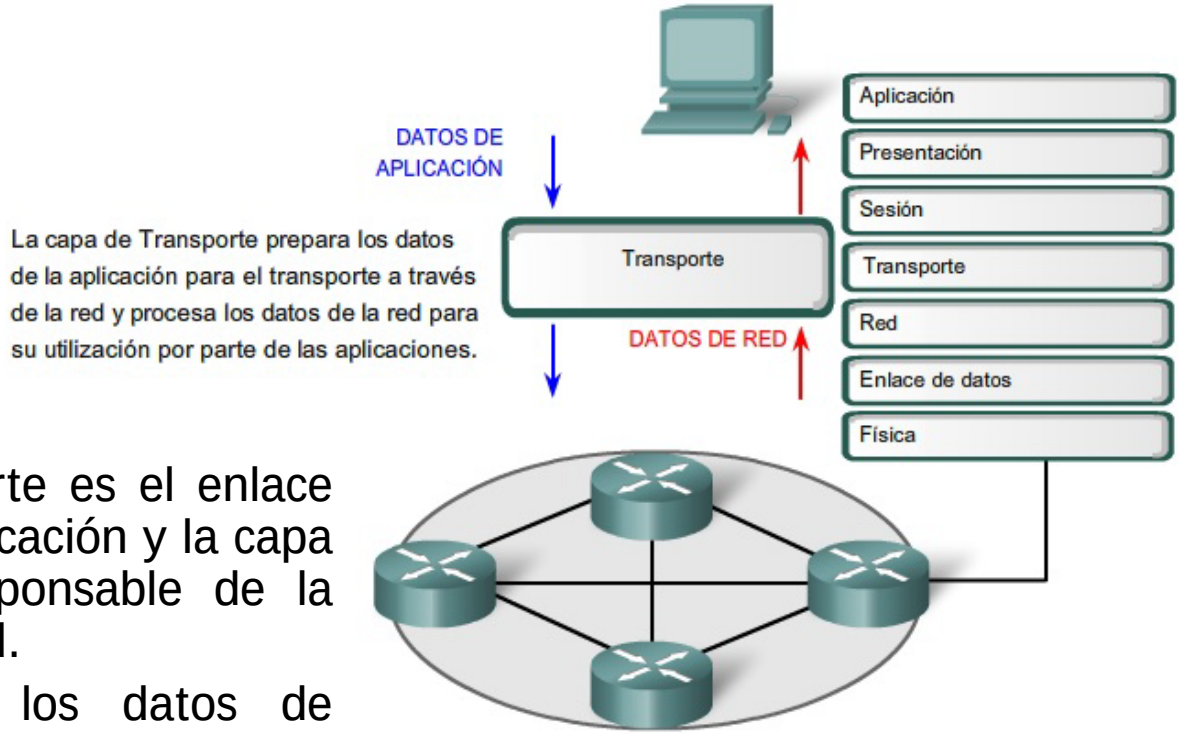
Capa de transporte

- La función principal del nivel de transporte es aceptar los datos de las capas superiores, dividirlos si es necesario, en unidades más pequeñas y pasarlos al nivel de red garantizando que lleguen a su destino de una forma económica y segura, independientemente la red o redes físicas que se encuentren en uso
- **Entidades de transporte**: hardware y/o software que se encargan de realizar este trabajo
- El dialogo entre entidades de transporte es extremo a extremo.
- También se encarga de enriquecer la calidad de servicio suministrada por el nivel de red.
- El nivel de transporte mejora la calidad ofrecida por el nivel de red mediante:
    - La multiplexación/demultiplexación.
    - La introducción de redundancia en la información

Capa de transporte



Capa de transporte

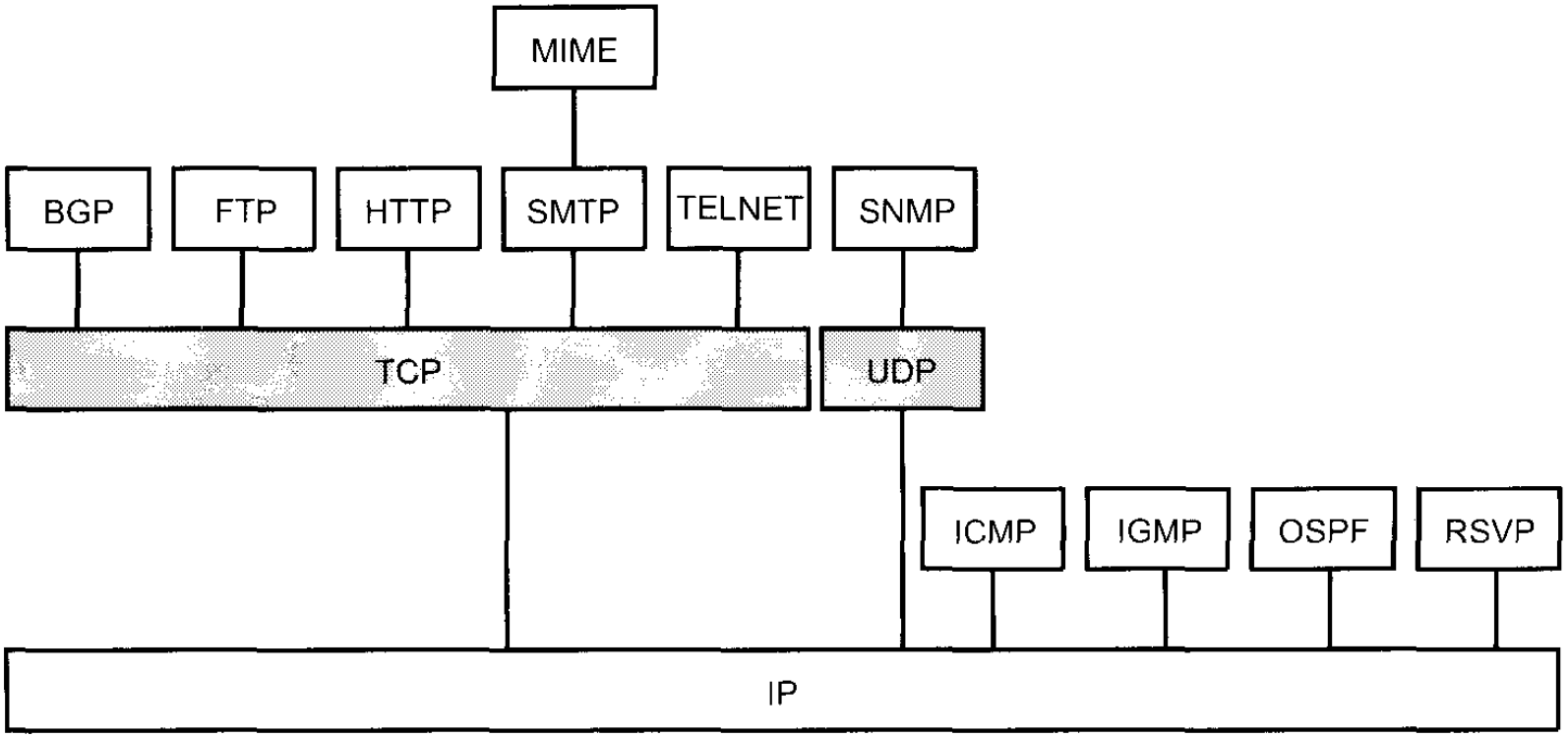


- La capa de transporte es el enlace entre la capa de aplicación y la capa inferior que es responsable de la transmisión de la red.
- Esta capa acepta los datos de diferentes conversaciones y las pasa a las capas inferiores como partes manejables que se pueden multiplexar de forma eventual en la red.

Capa de transporte

- El protocolo de transporte proporciona un servicio de transferencia de datos extremo-a-extremo.
- Si el servicio de interconexión es no seguro, como el caso de IP, el protocolo de nivel de transporte orientado a conexión resulta complejo.
  - Necesidad de tratar retardos variables y grandes (en algunos casos) entre sistemas finales.
  - Estos retardos, dificultan en control de flujo y el control de errores.

Protocolos de la capa de transporte



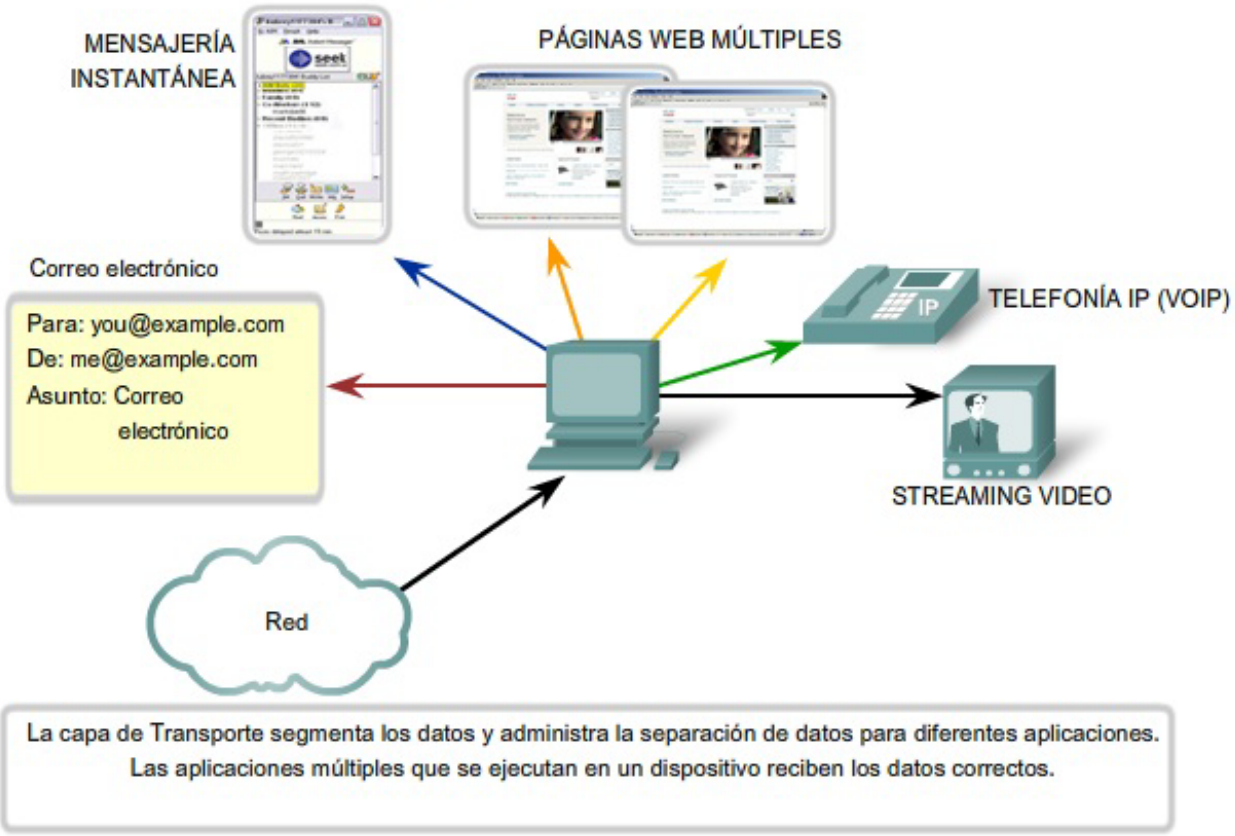
Propósito de la capa de transporte

Propósito de la capa de transporte

- La capa de transporte permite la **segmentación** de datos y brinda el **control** necesario **para reensamblar** las partes dentro de los distintos flujos de datos.
- Las **responsabilidades** principales que debe cumplir son:
  - Rastreo de comunicación individual entre origen y destino
  - Segmentación de datos y manejo de cada parte
  - Reensamble de segmentos
  - Identificación de diferentes aplicaciones

Responsabilidad de la capa de transporte

- **Rastreo de comunicación individual entre origen y destino**



Responsabilidad de la capa de transporte

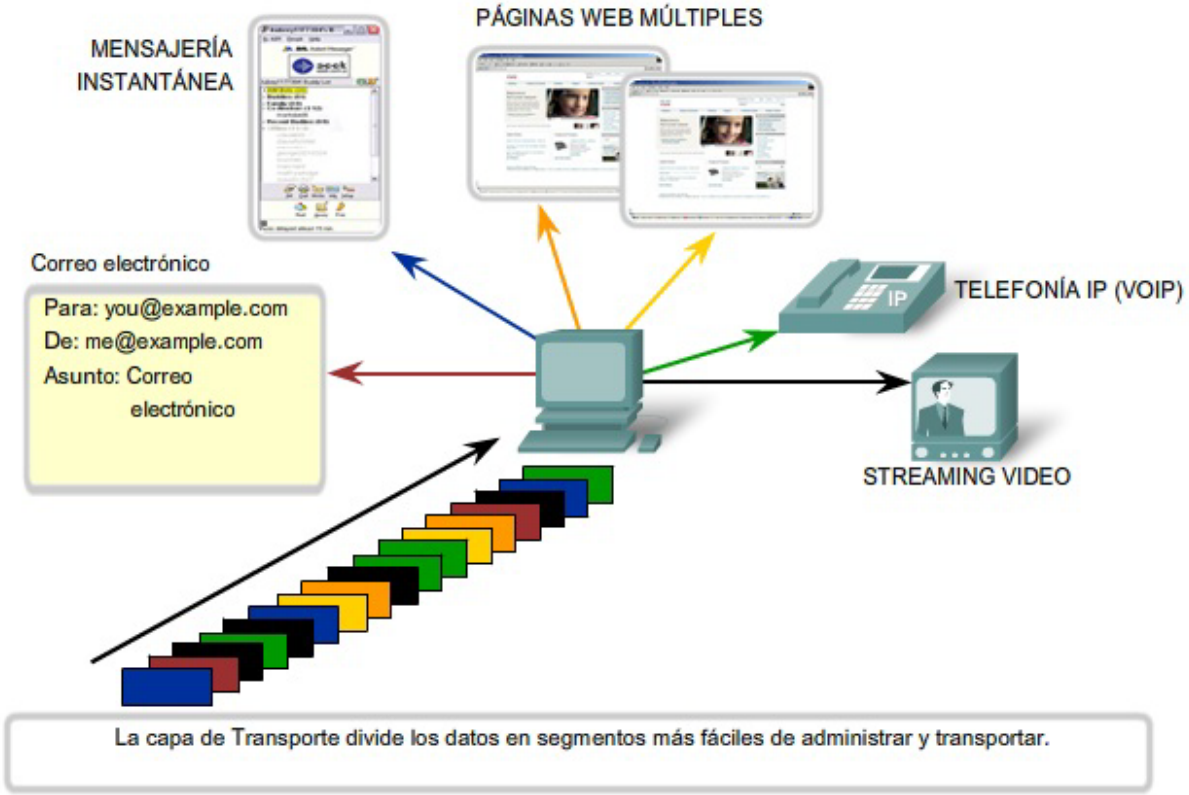
- **Rastreo de comunicación individual entre origen y destino**
  - Cada aplicaciones envía y recibe datos en la red al mismo tiempo.
  - Se precisa que el correo electrónico o página Web se reciba por completo pero retrasos ligeros se consideran aceptables para asegurar que la información se reciba y se presente por completo.
  - La pérdida ocasional de partes pequeñas de una conversación telefónica se puede considerar aceptable.

Responsabilidad de la capa de transporte

- **Segmentación de datos y manejo de cada parte**
  - Cada aplicación crea datos para enviarse a una aplicación remota
  - Estos datos se deben preparar para ser enviados a través de los medios.
  - Los protocolos de la capa de transporte describen los servicios que segmentan estos datos.
  - Se requiere que se agreguen encabezados en la capa de transporte para indicar la comunicación a la cual está asociada e identificar las partes de los segmentos.

Responsabilidad de la capa de transporte

- **Segmentación de datos y manejo de cada parte**



Responsabilidad de la capa de transporte

- **Reensamble de segmentos:**
  - En recepción, cada sección de datos se direcciona a la aplicación adecuada.
  - Estas secciones de datos individuales deben reconstruirse para generar una trama completa de datos que sea útil para la capa de aplicación.
  - Los protocolos en la capa de transporte describen cómo se utiliza la información del encabezado de la capa para reensamblar las partes de los diferentes segmentos recibidos y pasarlos a la capa de aplicación

Responsabilidad de la capa de transporte

- **Identificación de diferentes aplicaciones:**
  - Para pasar la trama de datos a las aplicaciones adecuadas, la capa de transporte debe identificar la aplicación final.
  - La capa de transporte asigna un identificador a la aplicación.
  - Los protocolos TCP/IP denominan a este identificador **número de puerto**.

Los requisitos:

- Hay múltiples protocolos de la capa de transporte debido a que las aplicaciones tienen diferentes requisitos.
- Para algunas aplicaciones, los segmentos deben llegar en una secuencia específica.
- En recepción:
  - Todos los datos deben recibirse para ser utilizados por la aplicación.
  - La aplicación puede tolerar cierta pérdida de datos durante la transmisión.

Los requisitos:

- Tipos de protocolos:
  - Algunos protocolos proporcionan sólo las funciones básicas para enviar de forma eficiente partes de datos entre las aplicaciones adecuadas (útiles para aplicaciones con datos son sensibles a retrasos).
  - Otros, describen los procesos que proporcionan características adicionales, como asegurar un envío confiable entre las aplicaciones (requieren recursos adicionales y generan mayor trafico en la red)
- 2 tipos de protocolos
  - Orientado a conexión (TCP)
  - No orientado a conexión (UDP)

Calidad de servicio y coste

Parámetros de calidad

- **Retardo de establecimiento de conexión:** tiempo que transcurre entre la solicitud de conexión de transporte y la confirmación que recibe el usuario del servicio. Incluye el retardo de procesamiento en la entidad de transporte remota.
- **Probabilidad de fallo de establecimiento de la conexión:** Probabilidad de que la conexión no se pueda establecer en el tiempo máximo permitido por la aplicación.
- **Caudal:** Volumen de información transmitido en la unidad de tiempo.
- **Retardo de tránsito:** Tiempo que transcurre entre el envío de un mensaje por el usuario del transporte en la máquina fuente hasta su recepción por el usuario de transporte en la máquina destino.
- **Tasa residual de errores:** Probabilidad de que un mensaje enviado por la entidad de transporte no llegue a su destino o llegue defectuoso

Parámetros de calidad

- **Probabilidad de fallo de transferencia:** Probabilidad de respetar los parámetros de calidad de servicio acordados al solicitar la conexión.
- **Retardo de liberación de conexión:** Tiempo que transcurre desde que un usuario comunica su deseo de liberar la conexión hasta la liberación real en el otro extremo.
- **Prioridad:** La posibilidad de especificar la importancia de unas conexiones frente a otras permite, al sistema en situaciones de congestión, garantizar que las conexiones más prioritarias se establecerán en primer lugar.
- **Fiabilidad:** Probabilidad de que el sistema no libere conexiones de modo inesperado o indeseado por problemas internos de congestión.

Negociación de la calidad de servicio

- Previo al establecimiento de una conexión las dos entidades de transporte establecen una negociación sobre los parámetros de calidad de servicio.
  - Normalmente se aportan dos valores el **deseado y el mínimo aceptable**.
  - Si los valores mínimos no son alcanzables se aborta el intento de conexión.
- Si no se puede suministrar el valor deseado se suministra otro por encima del mínimo.
- El nivel de transporte del destinatario estudia la viabilidad de conexión con los valores recibidos:
  - No puede aceptarlos-- hace una contraoferta con otros valores
  - Acepta-- se establece la conexión
- El usuario es informado del resultado del proceso y de los valores acordados

Coste y eficiencia

- El nivel de transporte es el encargado de administrar los recursos de red de forma que satisfagan las necesidades de sus usuarios.
- Si el coste de establecimiento de conexión es elevado se multiplexarán varias conexiones de nivel de transporte sobre una conexión del nivel de red.
- Se pueden utilizar varias conexiones de nivel de red para manejar el caudal de transferencia que ofrece el usuario del servicio.

Primitivas del servicio

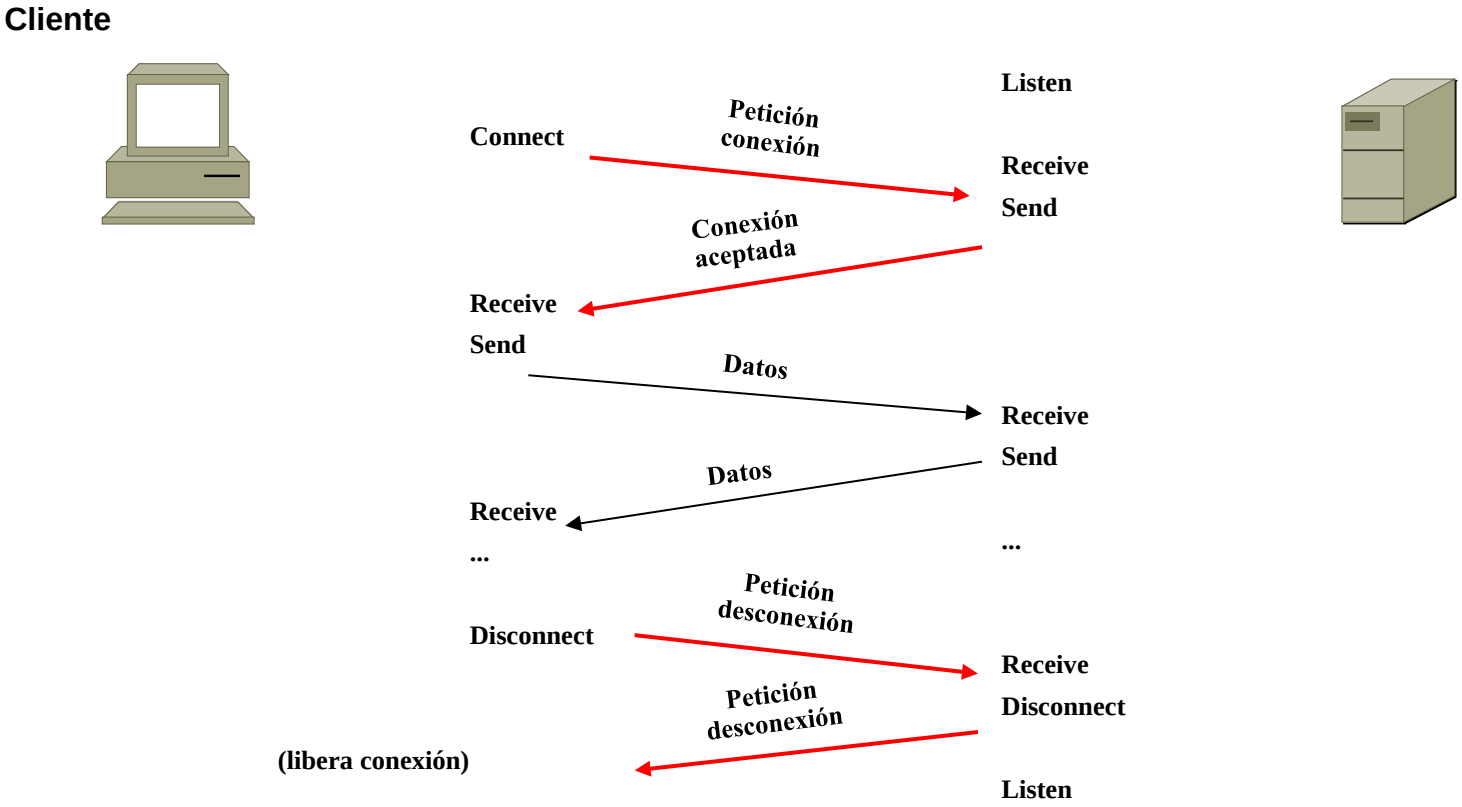
INTRODUCCIÓN

- Para permitir que los usuarios accedan la servicio de transporte, la capa de transporte debe proporcionar algunas operaciones a los programas de aplicación, es decir una interfaz con el servicio de transporte.
- Las primitivas de servicio son procedimientos, funciones o métodos del lenguaje de alto nivel con el que programan los usuarios del servicio de transporte.
- Se entiende por entidad de transporte local a aquella que atiende las peticiones de un usuario, dialoga directamente con éste intercambiando primitivas y que se ejecuta en la misma máquina.

Primitivas de un servicio de transporte sencillo

Primitiva	Paquete enviado	Significado
LISTEN	(ninguno)	Se bloquea hasta que un proceso intenta la conexión
CONNECT	CONNECTION REQ.	Intenta activamente establecer una conexión
SEND	DATA	Envía información
RECEIVE	(ninguno)	Se bloquea hasta que llega un paquete DATA
DISCONNECT	DISCONNECTION REQ.	Este lado quiere liberar la conexión

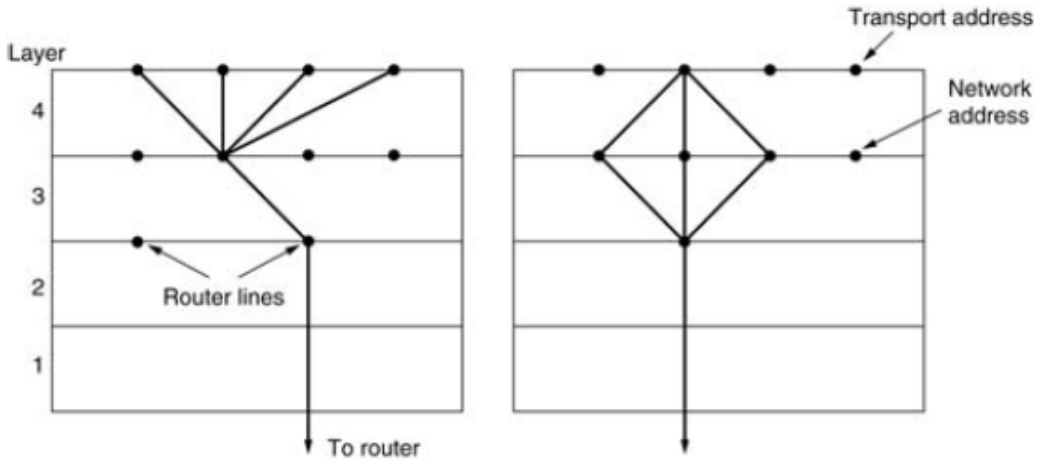
Ejemplo: Servicio orientado a la conexión



Proceso de Multiplexación y Demultiplexacion

Multiplexación/Demultiplexación:

- Varias conexiones de transporte en una conexión de red. Se utiliza cuando el coste por conexión del servicio de red es elevado
- Una conexión de transporte en varias conexiones de red. Se utiliza cuando quiere aumentar el caudal o reducirse el retardo en una conexión de trasporte, con el fin de ofrecer la calidad de servicio demandada por un usuario del nivel de transporte.



Resumen:

- Funciones y servicios de la capa de transporte:

  Comunicación **extremo a extremo** (*end-to-end)*.

  Multiplexación/demultiplexación de aplicaciones → *puerto.*

- Protocolo UDP:

  Multiplexación/demultiplexación de aplicaciones**.**

  Servicio **no orientado a conexión, no fiable.**

- Protocolo TCP:

  Multiplexación/demultiplexación de aplicaciones**.**

  Servicio **orientado a conexión, fiable:** Control **de errores y de flujo**. Control de **la conexión**. Control de **congestión.**

Tema 3. CAPA DE TRANSPORTE EN INTERNET

1. Introducción a los protocolos de Capa de Transporte
2. **Protocolo de datagrama de usuario (UDP).**
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
4. Control de congestión.
4. Extensiones TCP.
5. Ejercicios.

USER DATAGRAM PROTOCOL (UDP)

- El protocolos **UDP (User Datagram Protocol)** se define en la RFC 768.
- Protocolo UDP, protocolo de datagrama de usuario, proporciona un servicio de entrega de datagramas sin conexión y no confiable.
- Utiliza IP, pero agrega la capacidad para distinguir entre varios destinos dentro de un mismo sistema terminal.
- Un programa de aplicación que utiliza UDP, asume la responsabilidad por los problemas de confiabilidad, incluyendo la pérdida , duplicación y retraso de los mensajes así como la entrega desordenada de los mismos o las posibles pérdidas de conectividad.
- UDP proporciona puertos de protocolo para distinguir entre muchos programas que se ejecutan dentro de una misma máquina

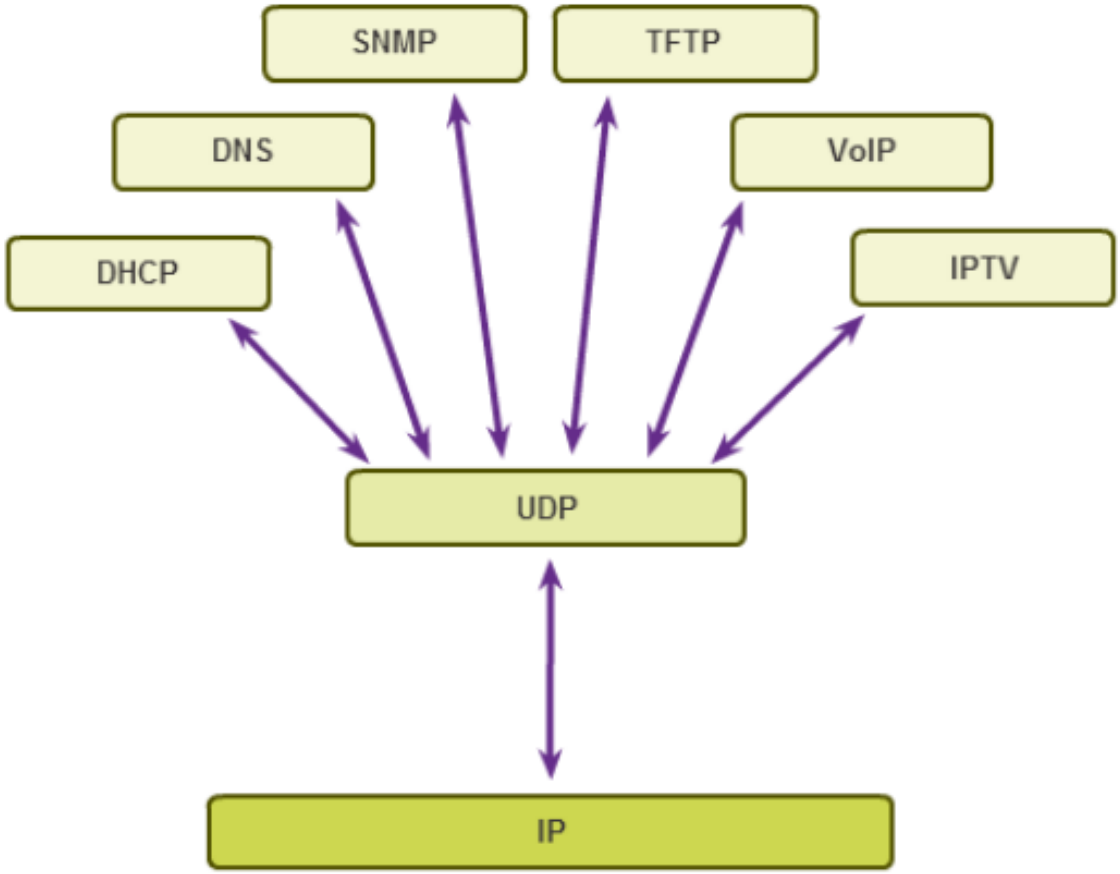
USER DATAGRAM PROTOCOL (UDP)

- Basa su funcionamiento en **"*best-effort*". ** Por tanto los segmentos UDP podrían:
  - Perderse
  - Entregarse fuera de orden a la aplicación
- Por tanto sus características principales son:
  - Servicio **no orientado a conexión**
  - Servicio **no fiable**
  - **No** hay garantías de **entrega ordenada**.
  - **No** hay **control de congestión**: entrega tan rápida como se pueda.
  - **Multiplexación/demultiplexación**: transportar las TPDU al proceso correcto.

USER DATAGRAM PROTOCOL (UDP)

- Suele usarse para:
  - Frecuentemente utilizado para aplicaciones de **streaming multimedia**
    - Tolerante a las pérdidas y sensibles a retardos
  - Otros usos de UDP:
    - El intercambio de mensajes es muy escaso, ej.:consultas al DNS (servidor de nombres)
    - La aplicación es en tiempo real y no puede esperar los ACKs. Ej.: videoconferencia, voz sobre IP.
    - Los mensajes se producen regularmente y no importa si se pierde alguno. Ej: SNMP
    - Se envía tráfico broadcast/multicast

➢ Resumen Usos



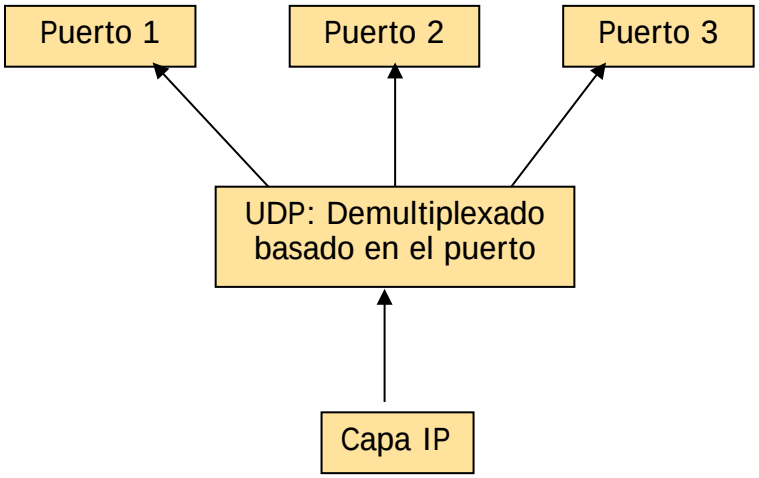
Formato Datagrama UDP

Formato Datagrama UDP

- Los números de puerto identifican los procesos emisor y receptor.
  - Los números de puerto UDP son independientes de los de TCP
- Longitud UDP = longitud de la cabecera UDP + longitud de datos.
  - El valor mínimo es de 8 bytes.
  - Es redundante con la información de la cabecera IP.
- Checksum: se calcula sobre la cabecera UDP y los datos UDP.
  - Se calcula de manera similar al checksum en IP (complemento a 1 de la suma de las palabras de 16 bits).
  - Diferencias:
    - El datagrama UDP puede contener un número impar de bytes → Se añade un byte de relleno (todo ceros).
    - Se considera una pseudo-cabecera de 12 bytes para el cálculo del checksum, que contiene algunos campos de la cabecera IP → Doble comprobación de estos campos (igual en TCP).

Puertos UDP

- UDP acepta datagramas de muchos programas de aplicación y los pasa al nivel de red IP para su transmisión y a la inversa.
- El multiplexado y demultiplexado entre el software UDP y los programas de aplicación ocurre a través del mecanismo de puerto.



Asignación de Puertos UDP

- Los números de puerto se pueden asignar de dos formas:
  - **Asignación estática**: existe una autoridad central que asigna los números de puerto conforme se necesitan y publica la lista de todas las asignaciones. Este enfoque se conoce como enfoque universal y las asignaciones de puerto especificada se conocen como asignaciones bien conocidas.
  - **Asignación dinámica:** siempre que un proceso necesita un puerto el software de red le asignará uno.
- Los diseñadores de TCP/IP adoptaron una solución híbrida, que preasigna muchos números de puerto pero que también deja muchos de ellos disponibles.

Asignación de Puertos UDP (Rangos)

- El campo de puerto tiene una longitud de 16 bits, lo que permite un rango que va desde 0 a 65535, pero no todos estos puertos son de libre uso:
  - El puerto **0 es un puerto reservado**, pero es un puerto permitido si el emisor no permite respuestas del receptor.
  - Los puertos **1 a 1023** reciben el nombre de **Puertos bien conocidos**, y en sistemas Unix, para enlazar con ellos, es necesario tener acceso como superusuario.
  - Los puertos **1024 a 49151** son los llamados **Puertos registrados**, y son los de libre utilización.
  - Los puertos del **491552 al 65535 son puertos efímeros**, de tipo temporal, y se utilizan sobre todo por los clientes al conectar con el servidor.

Algunos de los Puertos Bien conocidos (I)

- 20 (TCP), utilizado por FTP (File Transfer Protocol) para datos
- 21 (TCP), utilizado por FTP (File Transfer Protocol) para control
- 25 (TCP), utilizado por SMTP (Simple Mail Transfer Protocol)
- 53 (TCP), utilizado por DNS (Domain Name System)
- 53 (UDP), utilizado por DNS (Domain Name System)
- 67 (UDP), utilizado por BOOTP BootStrap Protocol (Server) y por DHCP
- 68 (UDP). utilizado por BOOTP BootStrap Protocol (Client) y por DHCP
- 69 (UDP), utilizado por TFTP (Trivial File Transfer Protocol)
- 80 (TCP), utilizado por HTTP (HyperText Transfer Protocol)
- 88 (TCP), utilizado por Kerberos (agente de autenticación)
- 110 (TCP), utilizado por POP3 (Post Office Protocol)
- 137 (TCP), utilizado por NetBIOS (servicio de nombres)
- 137 (UDP), utilizado por NetBIOS (servicio de nombres)
- 138 (TCP), utilizado por NetBIOS (servicio de envío de datagramas)
- 138 (UDP), utilizado por NetBIOS (servicio de envío de datagramas)

Algunos de los Puertos Bien conocidos (II)

- 139 (TCP), utilizado por NetBIOS (servicio de sesiones)
- 139 (UDP), utilizado por NetBIOS (servicio de sesiones)
- 143 (TCP), utilizado por IMAP4 (Internet Message Access Protocol)
- 443 (TCP), utilizado por HTTPS/SSL (transferencia segura de páginas web)
- 631 (TCP), utilizado por CUPS (sistema de impresión de Unix)
- 993 (TCP), utilizado por IMAP4 sobre SSL
- 995 (TCP), utilizado por POP3 sobre SSL
- 1080 (TCP), utilizado por SOCKS Proxy
- 1433 (TCP), utilizado por Microsoft-SQL-Server
- 1434 (TCP), utilizado por Microsoft-SQL-Monitor
- 1434 (UDP), utilizado por Microsoft-SQL-Monitor
- 1701 (UDP), utilizado para Enrutamiento y Acceso Remoto para VPN con L2TP.
- 1723 (TCP). utilizado para Enrutamiento y Acceso Remoto para VPN con PPTP.
- 1761 (TCP), utilizado por Novell Zenworks Remote Control utility
- 1863 (TCP), utilizado por MSN Messenger

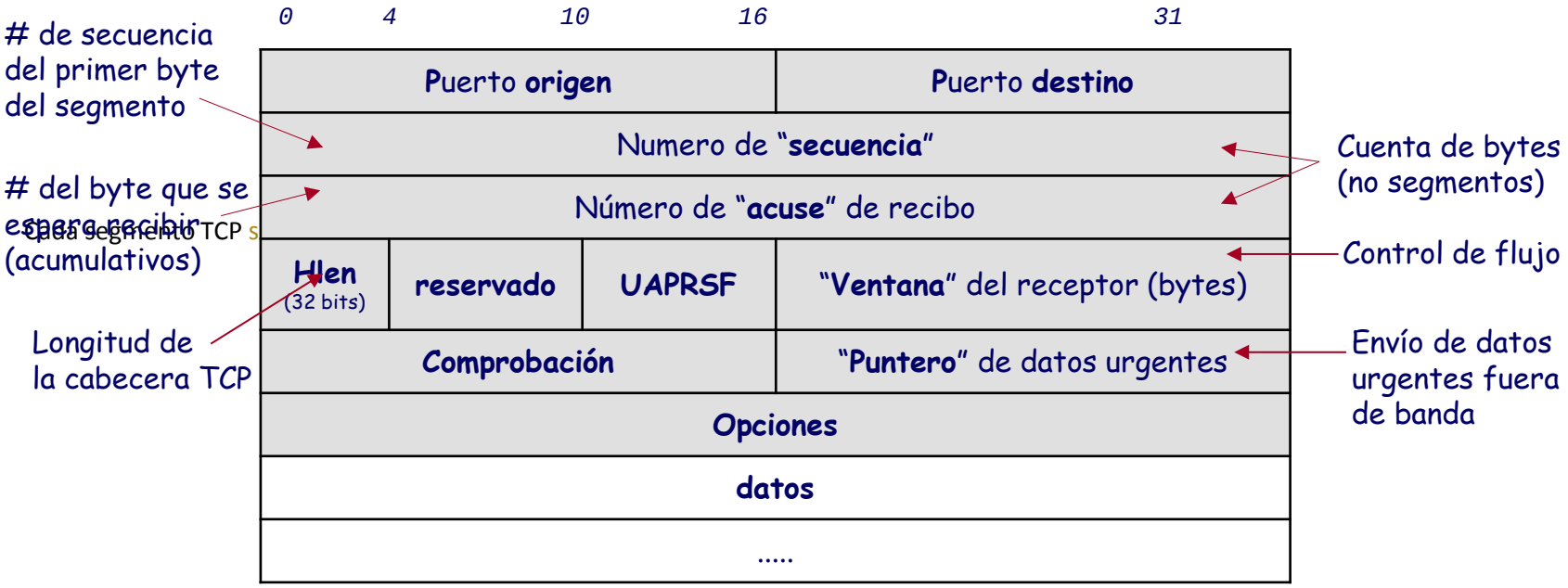
1. Introducción.
2. Protocolo de datagrama de usuario (UDP).
3. **Protocolo de control de transmisión (TCP).**
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. Extensiones TCP.
5. Ejercicios.

TRANSMISSION CONTROL PROTOCOL (TCP)

Características del "Transmission Control Protocol": RFC 793 (1122, 1323, 2018, 2581).

- Servicio orientado a conexión ("*hand-shaking*")
- Entrega ordenada
- *full-duplex*
- Mecanismo de control de flujo de detección y recuperación de errores (ARQ – Automatic Repeat reQuest)
  - confirmaciones positivas (ACKs) y acumulativas
  - Incorporación de confirmaciones ("piggybacking")
  - "timeouts" adaptables
  - Ventanas adaptables
- Mecanismo de control de congestión
- Servicio fiable → control de congestión y control de flujo

TPDU TCP = **Segmento TCP**:



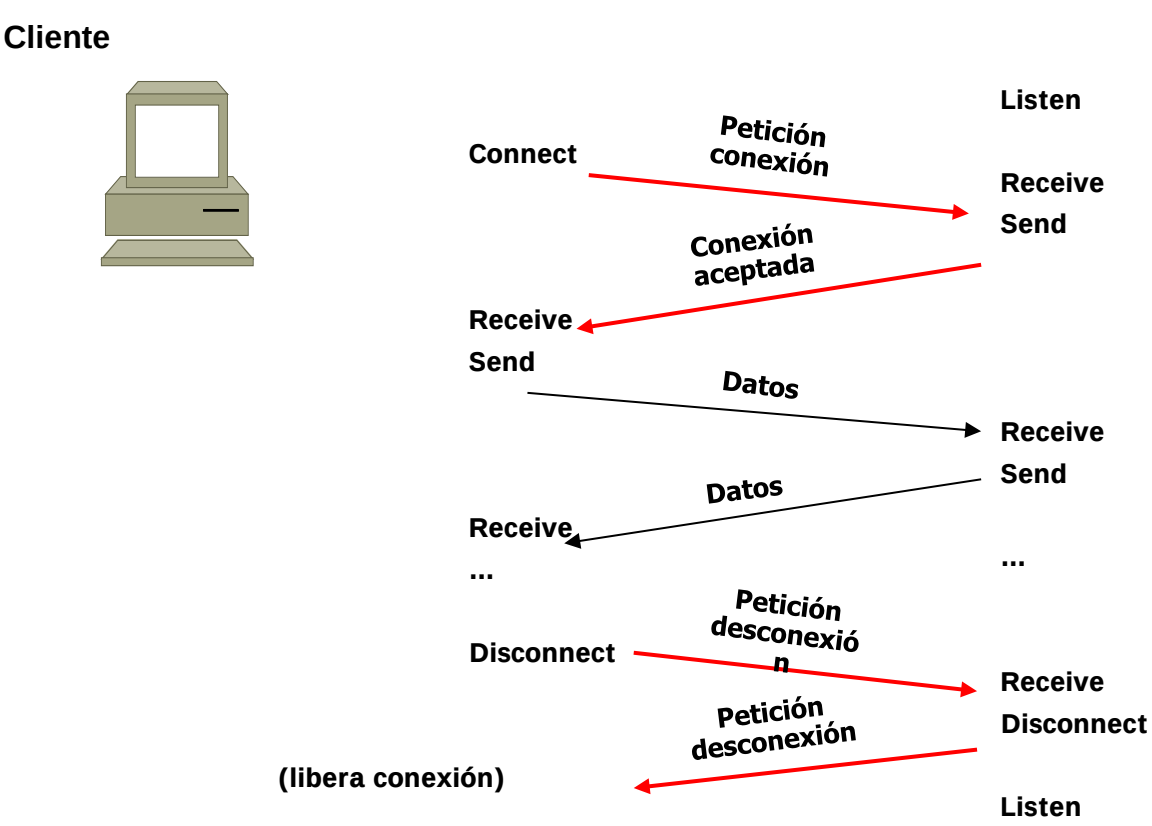
Multiplexación/demultiplexación de aplicaciones:

Puerto	Aplicación/Servicio	Descripción
20	FTP-DATA	Transferencia de ficheros: datos
21	FTP	Transferencia de ficheros: control
22	SSH	Terminal Seguro
23	TELNET	Acceso remoto
25	SMTP	Correo electrónico
53	DNS	Servicio de nombres de domino
80	HTTP	Acceso hipertexto (web)
110	POP3	Descarga de correo

La “conexión TCP” se identifica por: puerto e IP origen y puerto e IP destino (y opcionalmente protocolo)

CONEXIÓN/DESCONEXIÓN.

- Primitivas básicas de conexión/desconexión

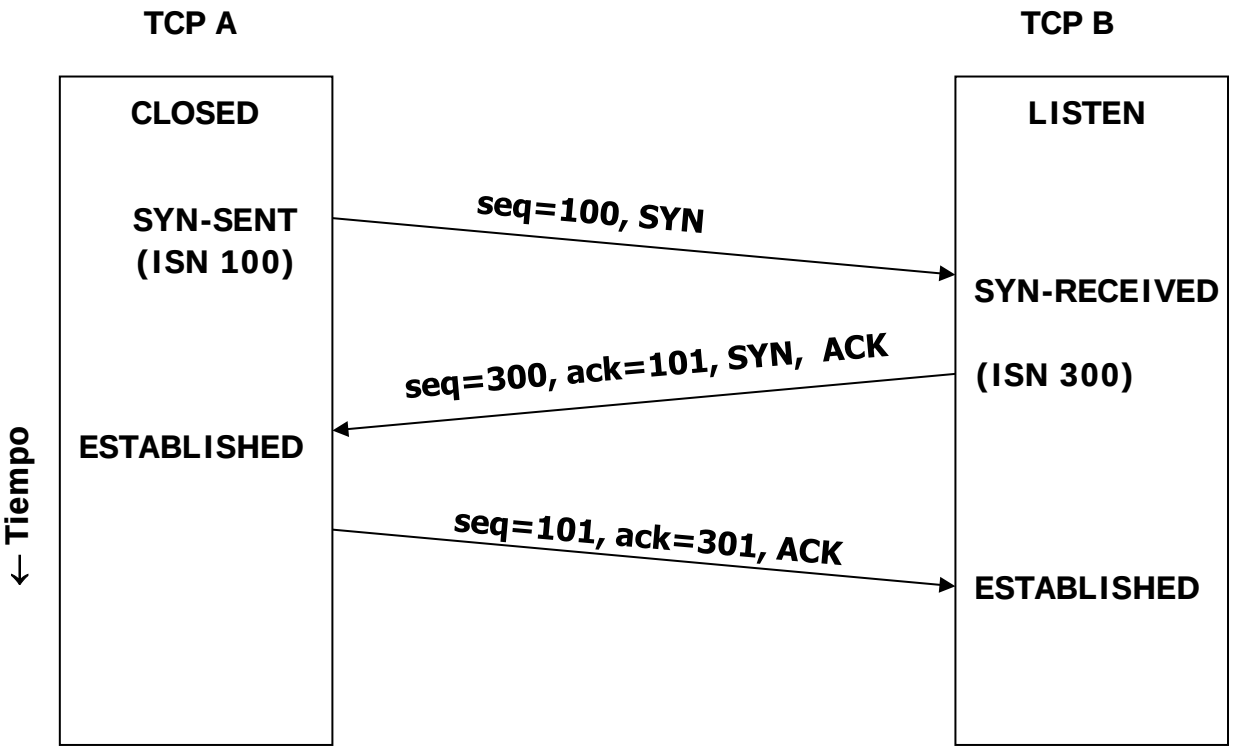


Conexión por 'Saludo a tres vías':

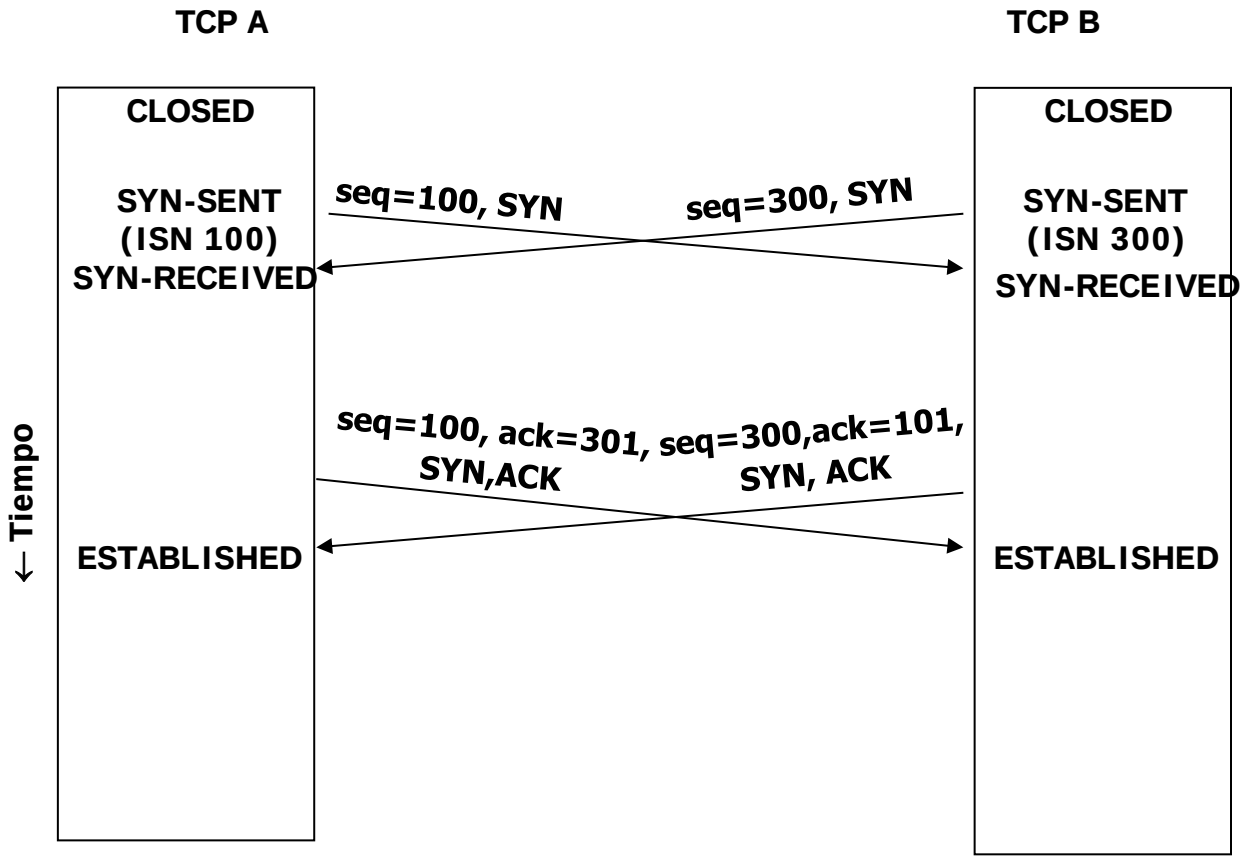
- En TCP pueden llegar segmentos duplicados (ej. Se pierde la confirmación de un segmento con lo que el emisor lo reenvía)
- Con un procedimiento de conexión simple los segmentos duplicados podrían causar problemas, como que una sesión entera se duplique.
- Para evitarlo se utiliza el saludo a tres vías, un procedimiento de conexión más elaborado que evita los problemas debidos a duplicados
- Para ello se identifica cada intento de conexión mediante un número diferente. El cliente elige un número a modo de clave para la comunicación en sentido de ida y el servidor otro para el sentido de vuelta.
- Estos dos números actúan como PINs que identifican cada intento de conexión y lo protegen de segmentos retrasados que pudieran aparecer fruto de conexiones anteriores.

- El cliente elige para cada intento de conexión un número único. El número elegido lo incluye en la petición de conexión que envía al servidor.
- El servidor, cuando recibe la petición, elige otro número único y envía una respuesta al cliente indicándoselo.
- El cliente al recibir la respuesta considera establecida la conexión. A continuación envía un tercer mensaje en el que acusa recibo del anterior. El servidor considera establecida la conexión cuando el recibe este tercer mensaje.

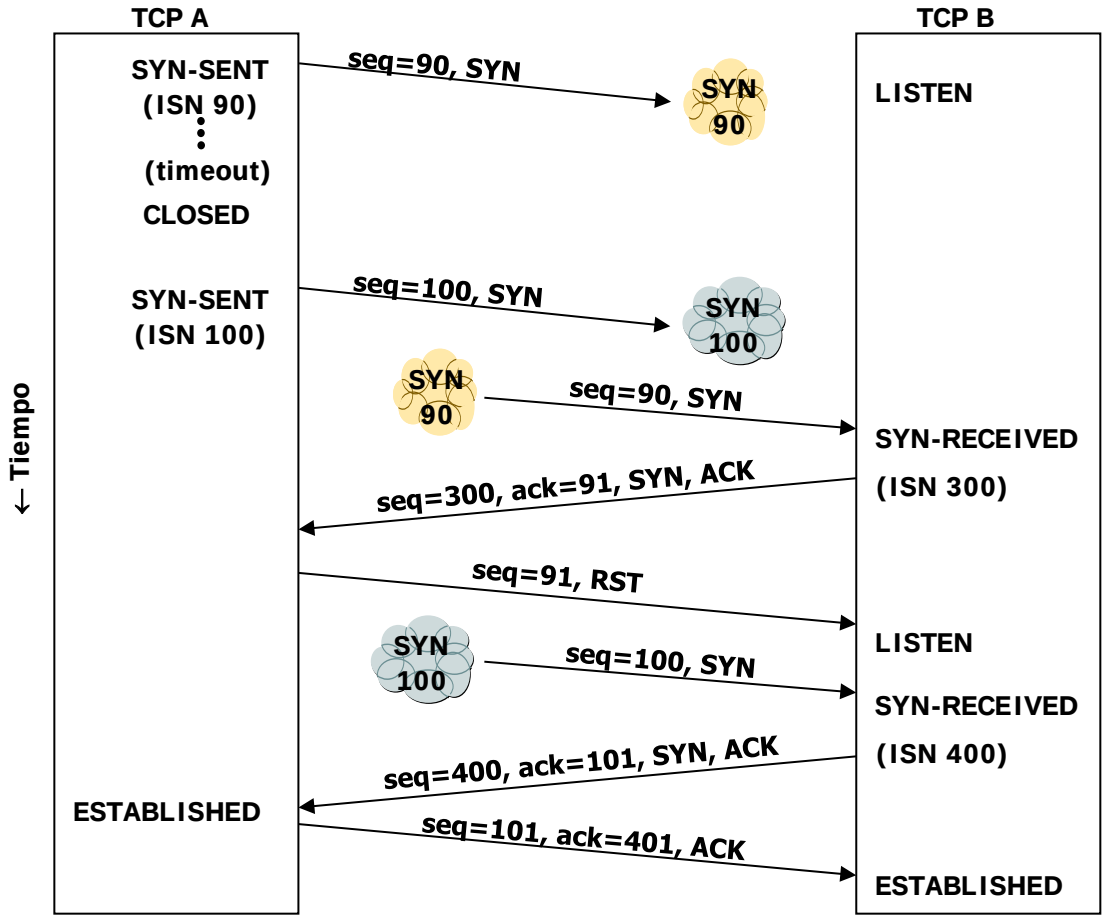
✓Establecimiento de una conexión TCP por saludo a tres vías



- Conexión saludo a tres vías, conexión simultánea



✓Conexión con SYN duplicado



Conexión en TCP

Los dos primeros segmentos de la conexión se identifican con el flag SYN.

El número de secuencia es un campo de 32 bits que cuenta bytes en módulo 2^{32} (el contador se da la vuelta cuando llega al valor máximo).

El número de secuencia no empieza normalmente en 0, sino en un valor denominado ISN (Initial Sequence Number) elegido al azar; el ISN sirve de 'PIN' en el saludo a tres vías para asegurar la autenticidad de la comunicación.

Una vez establecida la comunicación el 'seq' y el 'ack' sirven para contar los bytes transmitidos y recibidos.

- El ISN es elegido por el sistema (cliente o servidor). El estándar sugiere utilizar un contador entero incrementado en 1 cada 4 μs aproximadamente. En este caso el contador se da la vuelta (y el ISN reaparece) al cabo de 4 horas 46 min.
- El MSL (Maximum Segment Lifetime) típico es de unos 2 minutos, con lo que la probabilidad de que dos ISN coincidan es despreciable.
- El mecanismo de selección de los ISN es suficientemente fiable para proteger de coincidencias debidas al azar, pero no es un mecanismo de protección frente a sabotajes. Es muy fácil averiguar el ISN de una conexión e interceptarla suplantando a alguno de los dos participantes.

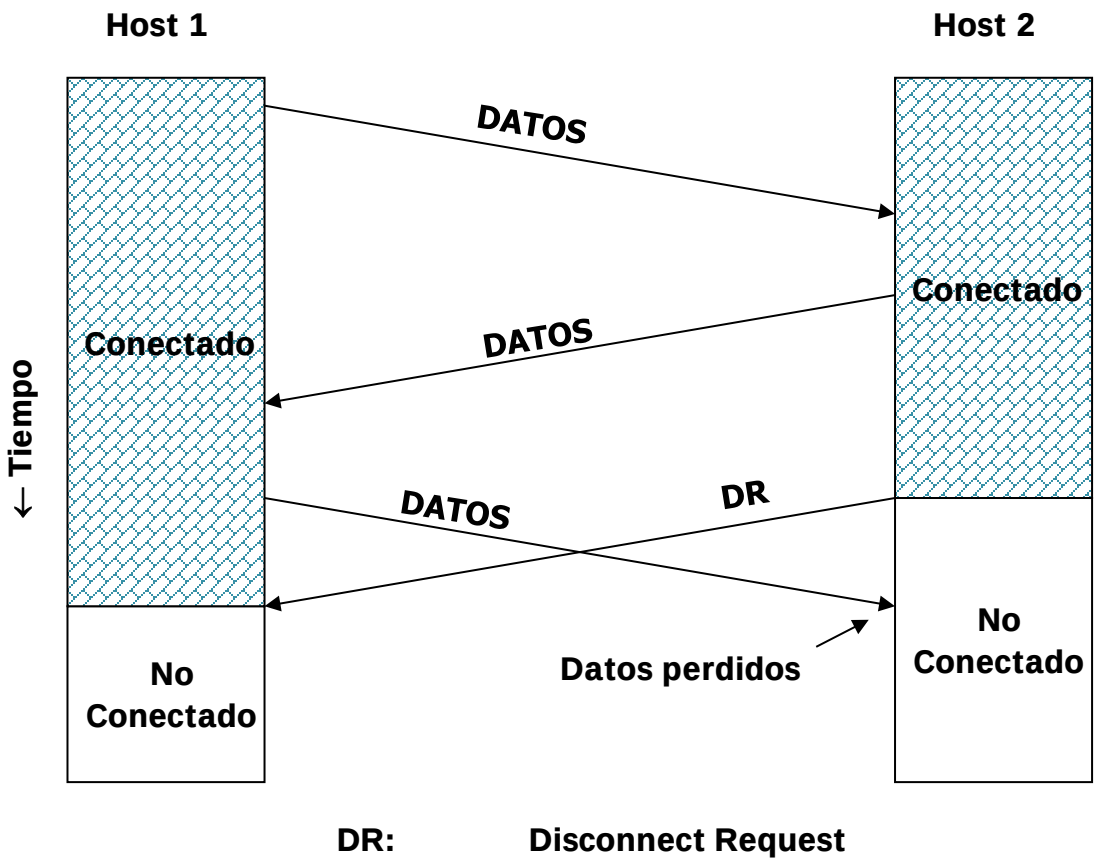
Desconexión en TCP

Puede ser de dos tipos:

Simétrica: la conexión se considera formada por dos circuitos simplex y cada host solo puede cortar uno (aquel en el que él emite datos). El cierre de un sentido se interpreta como una 'invitación' a cerrar el otro.

Asimétrica: desconexión unilateral (un host la termina en ambos sentidos sin esperar a recibir confirmación del otro). Puede provocar pérdida de información.

✓Desconexión asimétrica



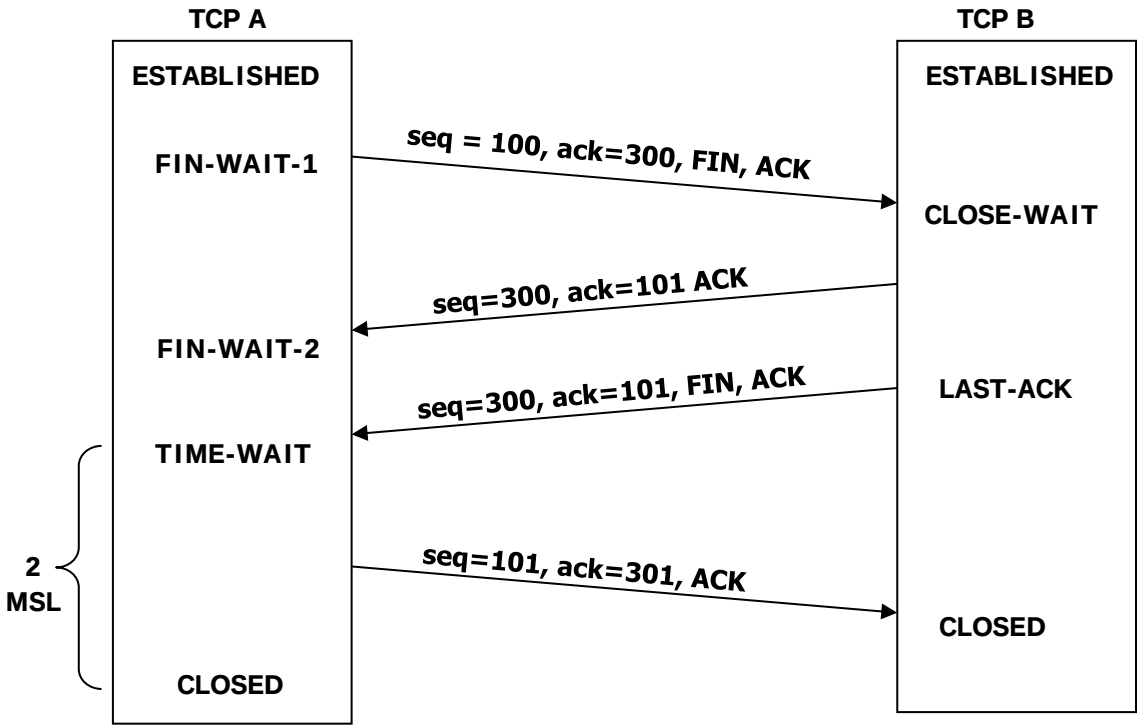
Desconexión por saludo a tres vías

- Se trata de una desconexión simétrica en la que se tiene una seguridad razonable de que no se pierden datos.
- Supone el intercambio de tres mensajes, de forma análoga a la conexión, de ahí su nombre.
- En caso de que alguno de los mensajes de desconexión se pierda una vez iniciado el proceso la conexión se termina por timeout.

Desconexión en TCP

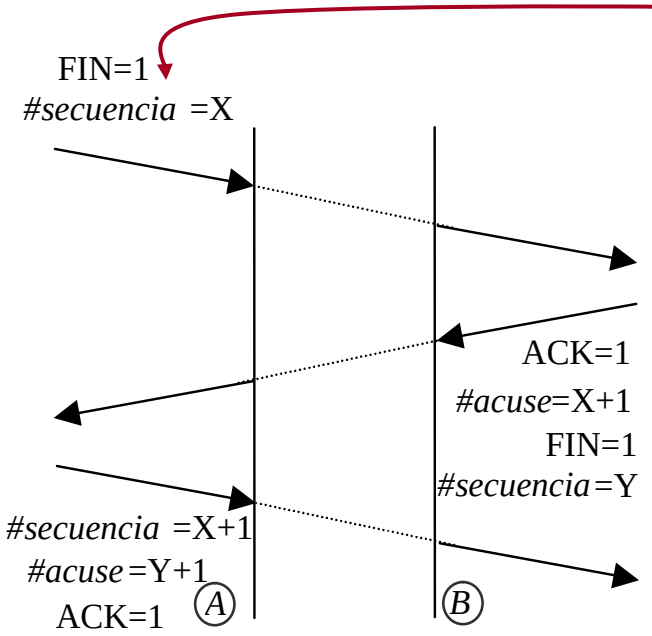
- Se utiliza el 'saludo a tres vías' invitando a la otra parte a cerrar.
- Para indicar el cierre se utiliza el flag FIN.
- La desconexión puede iniciarla el cliente o el servidor.
- Una vez efectuada la desconexión el host que inició el proceso está un cierto tiempo a la espera por si aparecen segmentos retrasados.

✓Desconexión a tres vías, caso normal

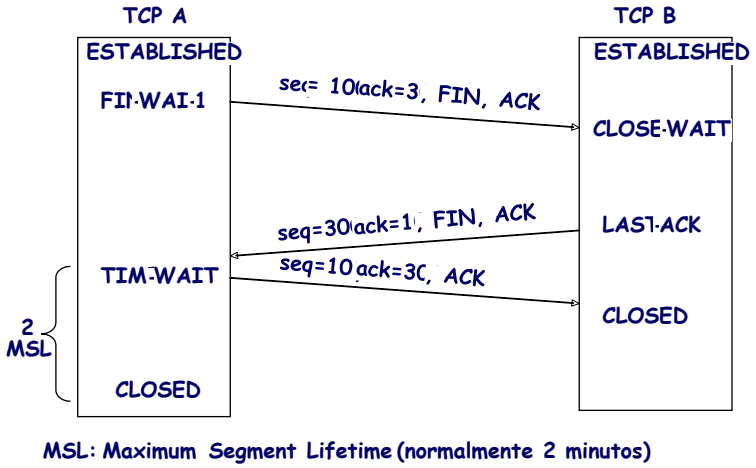


MSL: Maximum Segment Lifetime (normalmente 2 minutos)

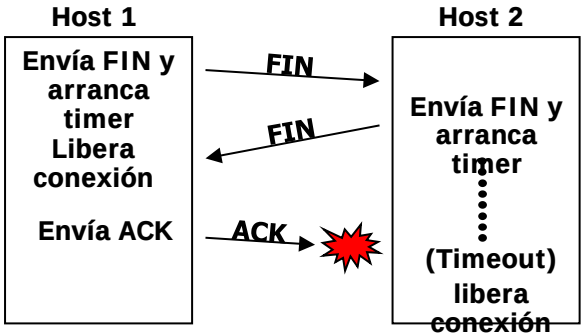
Control de la conexión: Cierre de la conexión, liberación de recursos.



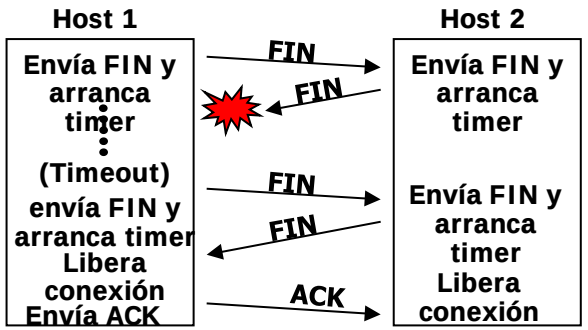
- "Cierre activo"
- "Cierre pasivo"
- Campos involucrados:
  - * Bit F (FIN) del campo control
  - * Campo secuencia
  - * Campo acuse
  - * Bit A (ACK) del campo control



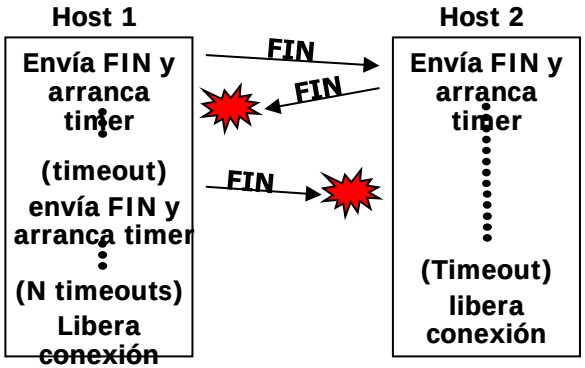
✓Otras formas de terminar una conexión TCP



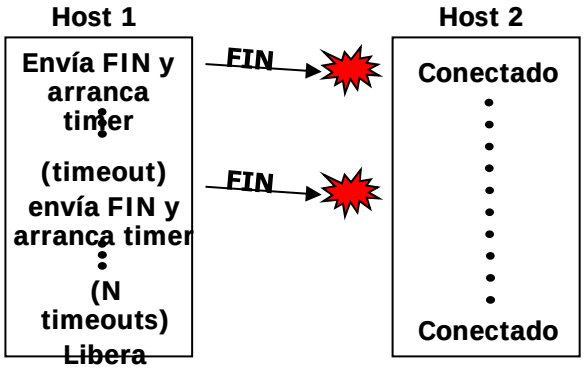
Pérdida de ACK final



Pérdida de respuesta FIN



Pérdida de todo menos primer FIN



Pérdida de todos los FIN de host 1

Intercambio de Información en TCP

Intercambio de datos TCP ↔ aplicación

- ✓ **Aplicación → TCP**: la aplicación envía los datos a TCP cuando quiere (siempre y cuando TCP tenga espacio libre en el buffer de emisión).
- ✓ **TCP → Aplicación**: la aplicación lee del buffer de recepción de TCP cuando quiere y cuanto quiere. Excepción: datos urgentes.
- ✓ Para TCP los datos de la aplicación son un flujo continuo de bytes, independientemente de la separación que pueda tener la aplicación (registros, etc.). Es responsabilidad de la aplicación asegurarse que esa separación (si existe) se mantendrá después de transmitir los datos.

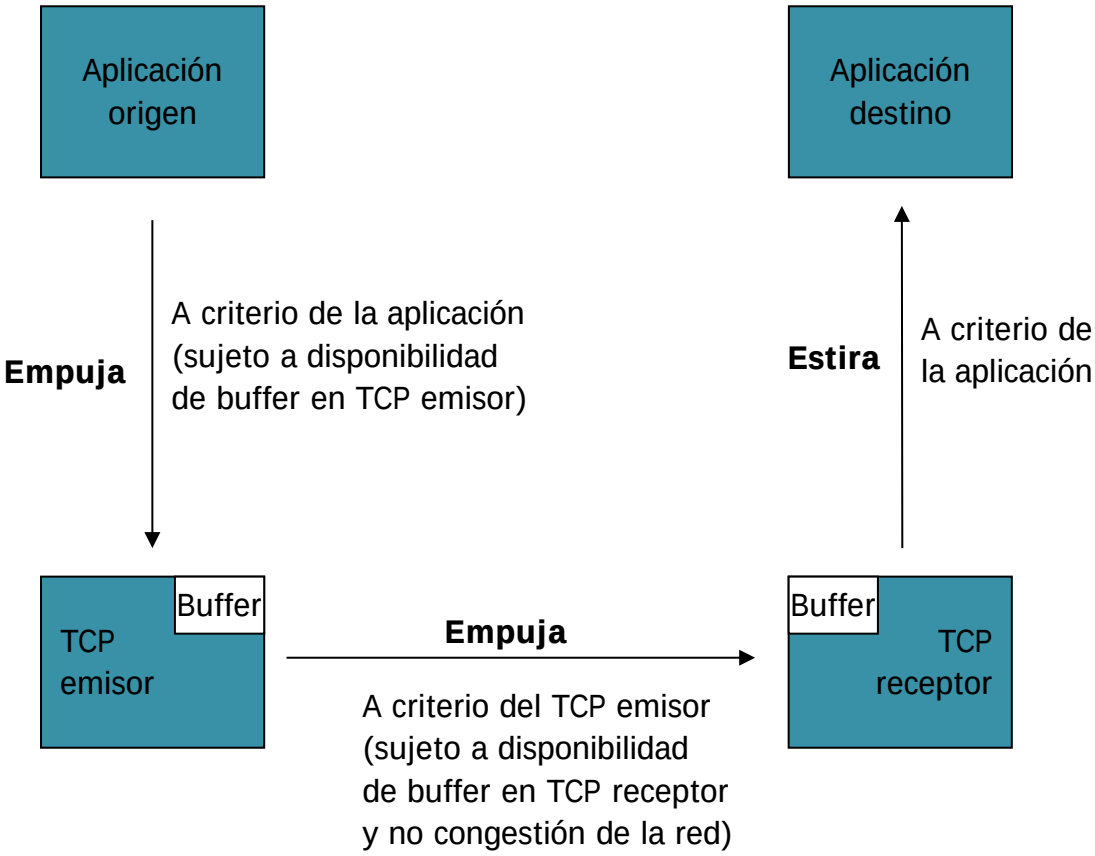
Intercambio de datos TCP ↔ TCP

El TCP emisor manda los datos cuando quiere. Excepción: datos 'Pushed'.

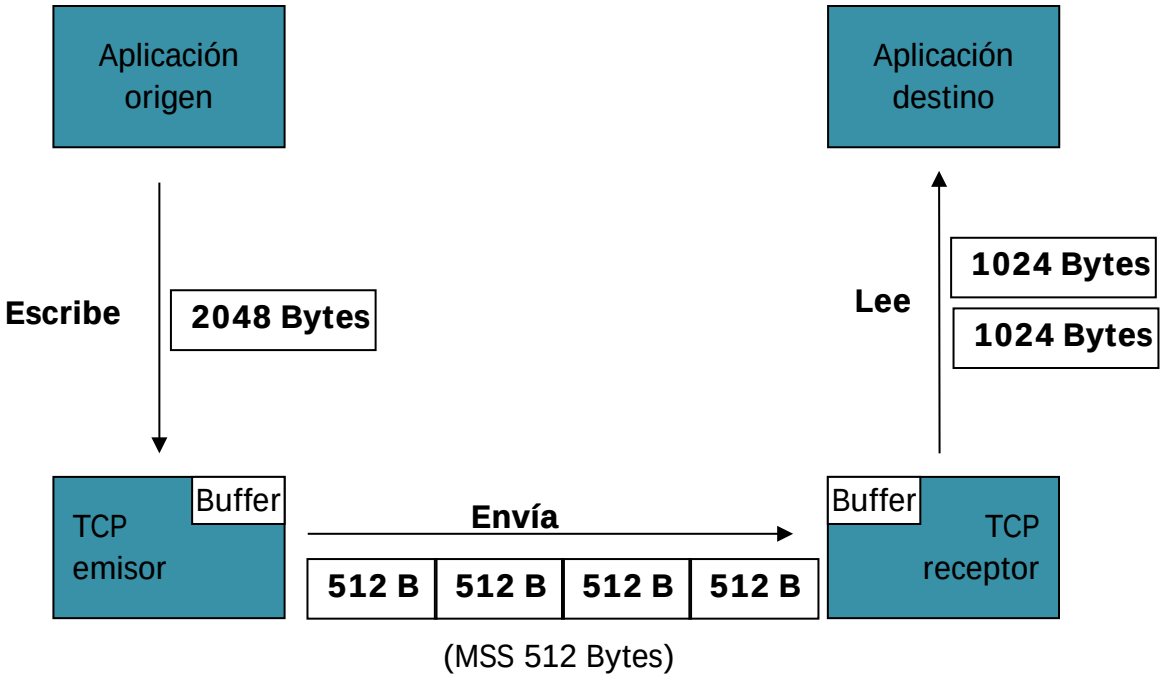
El TCP emisor decide el tamaño de segmento según sus preferencias. Al inicio de la conexión se negocia el MSS (Maximum Segment Size).

Normalmente TCP intenta agrupar los datos para que los segmentos tengan la longitud máxima, reduciendo así el overhead debido a cabeceras y proceso de segmentos..

✓Intercambio de datos. TCP ↔ Aplicación y TCP ↔ TCP



✓ Intercambio de datos. TCP ↔ Aplicación y TCP ↔ TCP



Intercambio de datos: casos excepcionales

Datos 'Pushed' (bit PSH)

La aplicación pide al TCP emisor que envíe esos datos lo antes posible. El TCP receptor los pondrá a disposición de la aplicación de inmediato, para cuando ésta le pida datos. Ejemplo: telnet.

Datos Urgentes (bit URG y Urgent Offset)

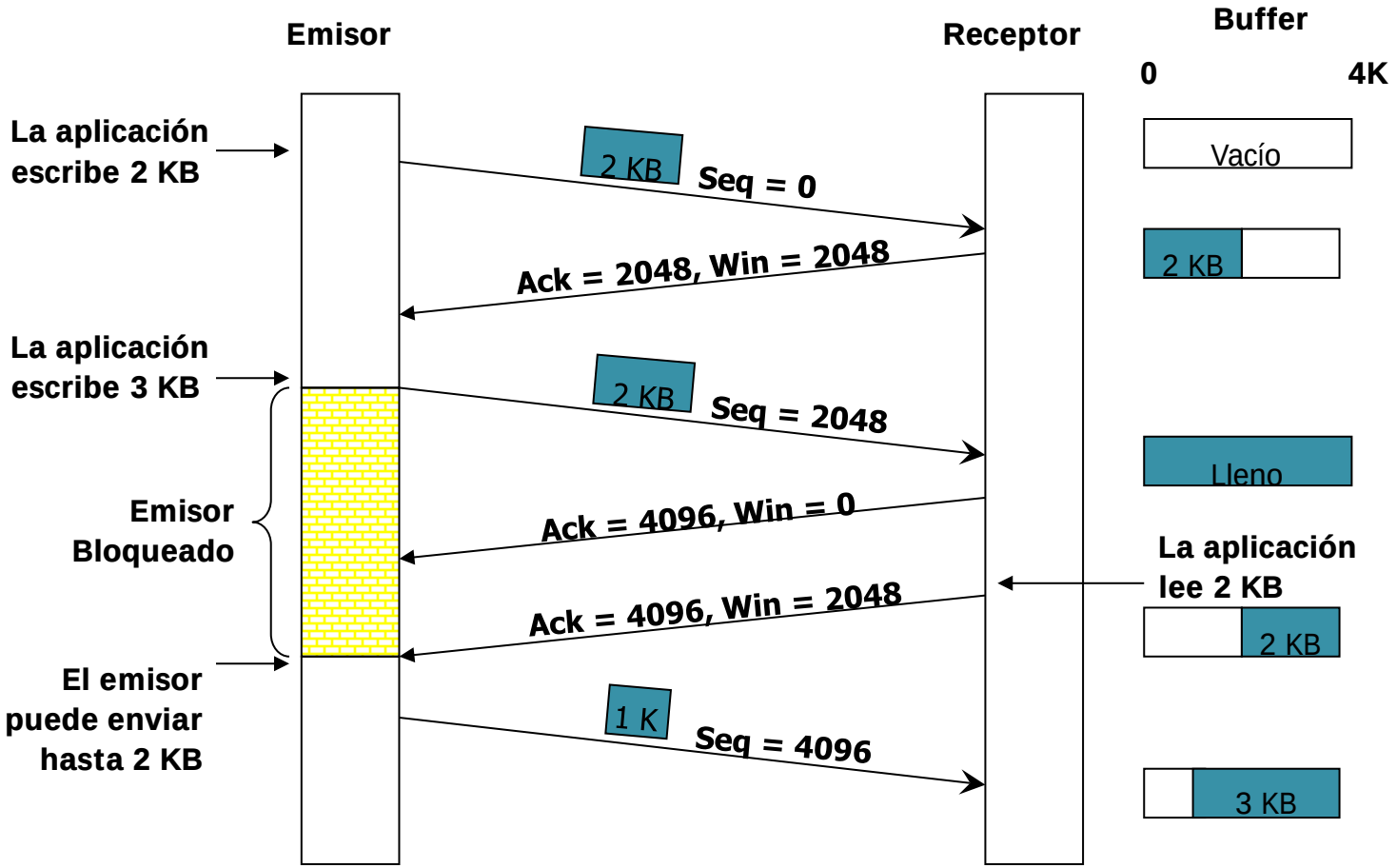
Los datos se quieren entregar a la aplicación remota sin esperar a que esta los pida. Ejemplo: abortar un programa con CTRL-C en una sesión telnet

http://www.eoi.es/blogs/ciberseguridad/2017/01/12/tcp-flags-psh-and-urg/

Gestión de Buffers y control de flujo

- El TCP receptor informa en cada segmento al emisor del espacio que le queda libre en el buffer para esa comunicación. Para ello usa el campo tamaño de ventana.
- Anunciando una ventana cero el receptor puede bloquear al emisor, y ejercer así control de flujo.
- La ventana anunciada es un espacio que el TCP receptor reserva para esa comunicación en su buffer.
- Tanto los números de secuencia como los tamaños de ventana cuentan bytes.

✓Gestión de buffers y Control de flujo



Control de errores y de flujo:

Control de errores: generación de ACKs (RFC 1122, 2581).

Evento	Acción del TCP receptor
Llegada ordenada de segmento, sin discontinuidad, todo lo anterior ya confirmado.	Retrasar ACK. Esperar recibir al siguiente segmento hasta 500 mseg. Si no llega, enviar ACK.
Llegada ordenada de segmento, sin discontinuidad, hay pendiente un ACK retrasado.	Inmediatamente enviar un único ACK acumulativo.
Llegada desordenada de segmento con # de sec. mayor que el esperado, discontinuidad detectada.	Enviar un ACK duplicado, indicando el # de sec. del siguiente byte esperado.
Llegada de un segmento que completa una discontinuidad parcial o totalmente.	Confirmar ACK inmediatamente si el segmento comienza en el extremo inferior de la discontinuidad.

Control de errores y de flujo:

Control de errores: ¿cómo estimar los "**timeouts**"?

Mayor que el tiempo de ida y vuelta (RTT).

Si es demasiado **pequeño**: **timeouts prematuros**.

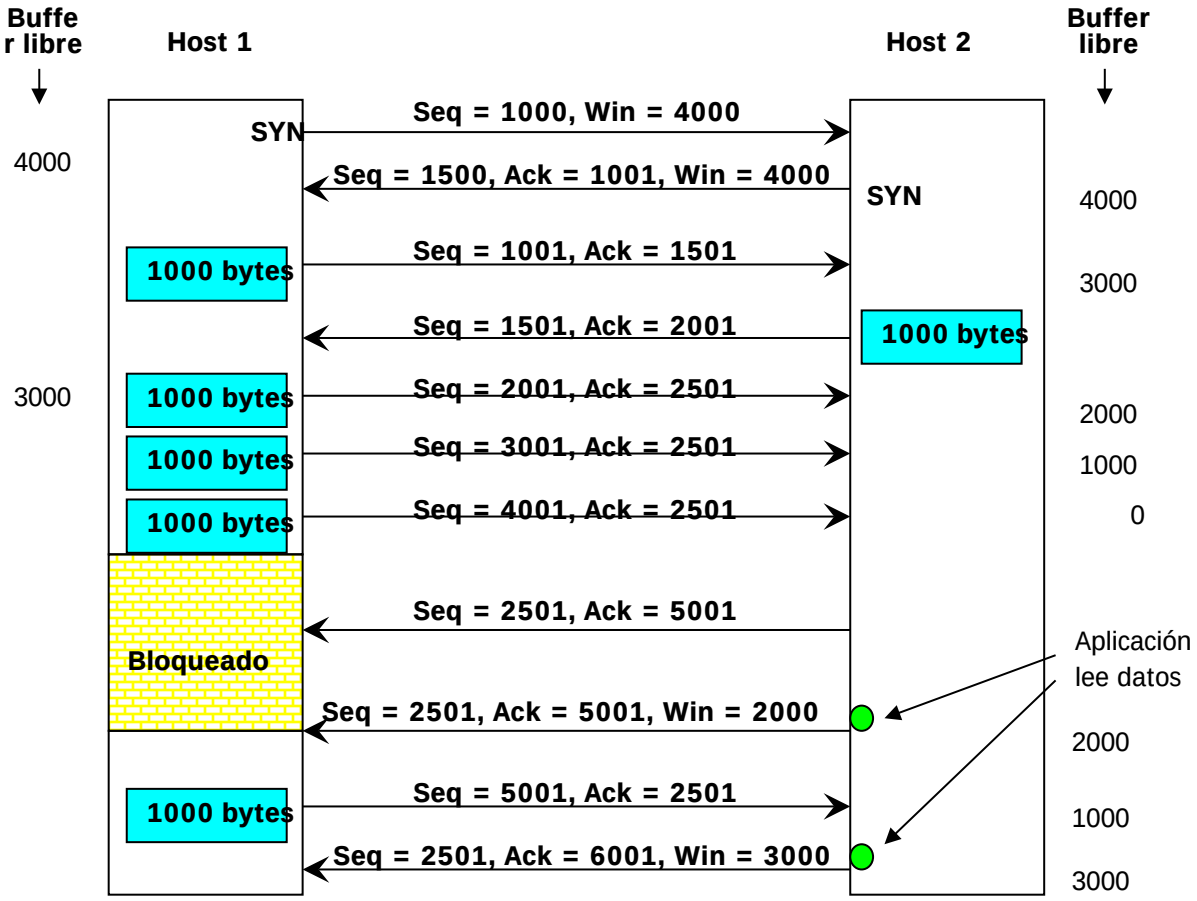
Si es demasiado **grande**: **reacción lenta** a pérdida de segmentos.

Para situaciones cambiantes la mejor solución es la adaptable.

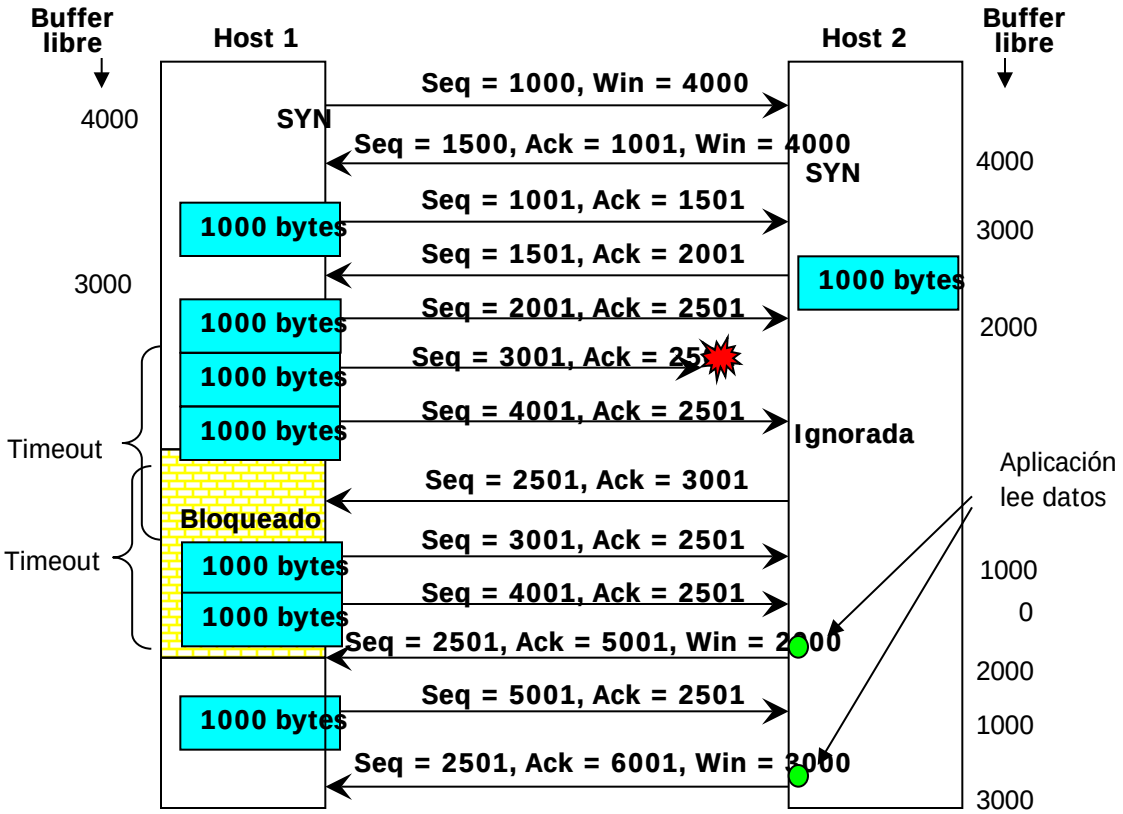
Reenvio de segmentos

- En caso de pérdida de un paquete en la red el segmento TCP no llegará a su destino
- Cada TCP cuando envía un segmento espera recibir el ACK; si este no llega dentro de un tiempo razonable reenvía el segmento.
- Si se enviaron varios segmentos y se pierde uno se puede hacer dos cosas:
  - Enviar solo ese segmento (repetición selectiva)
  - Enviar todos los segmentos a partir de ese (retroceso n)
- Lo normal es utilizar retroceso n

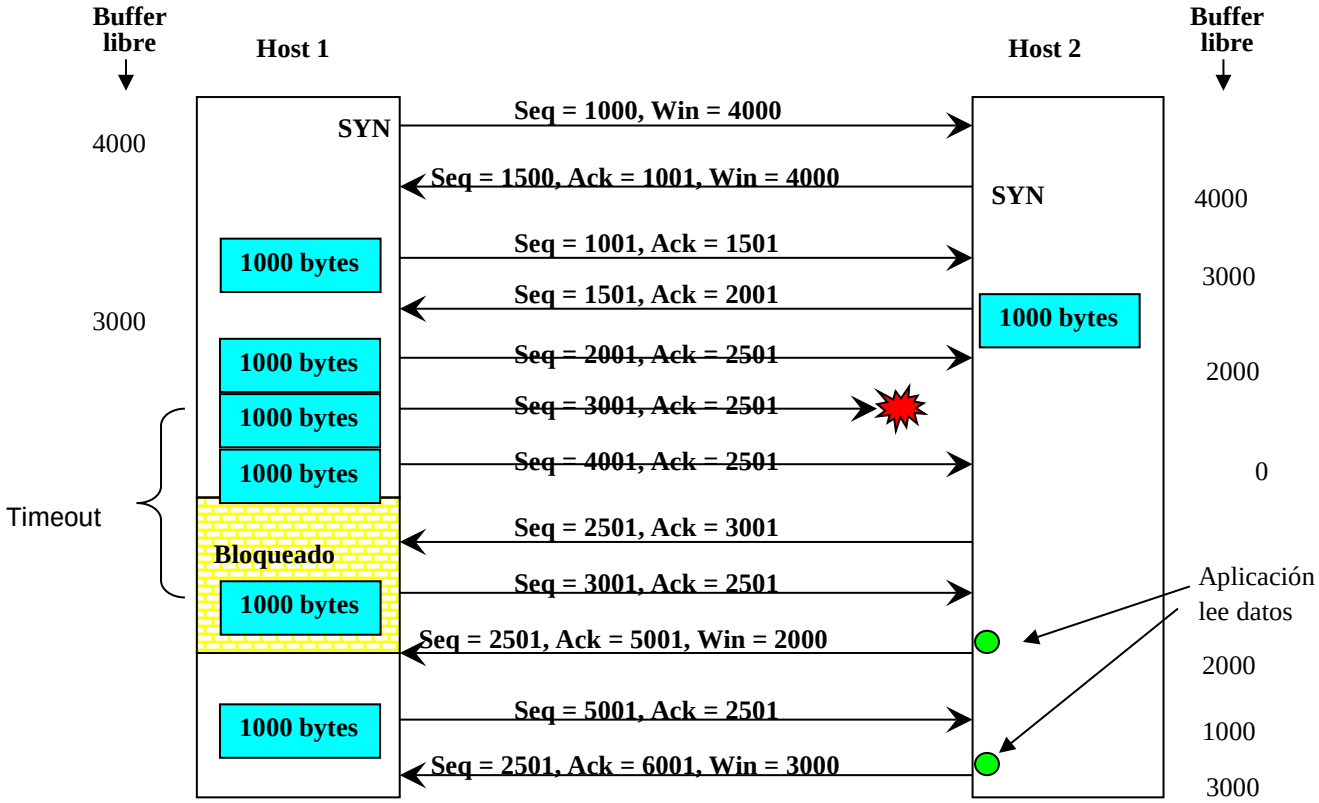
✓Control de flujo y números de secuencia



✓Retransmisión con retroceso n



✓Retransmisión con repetición selectiva

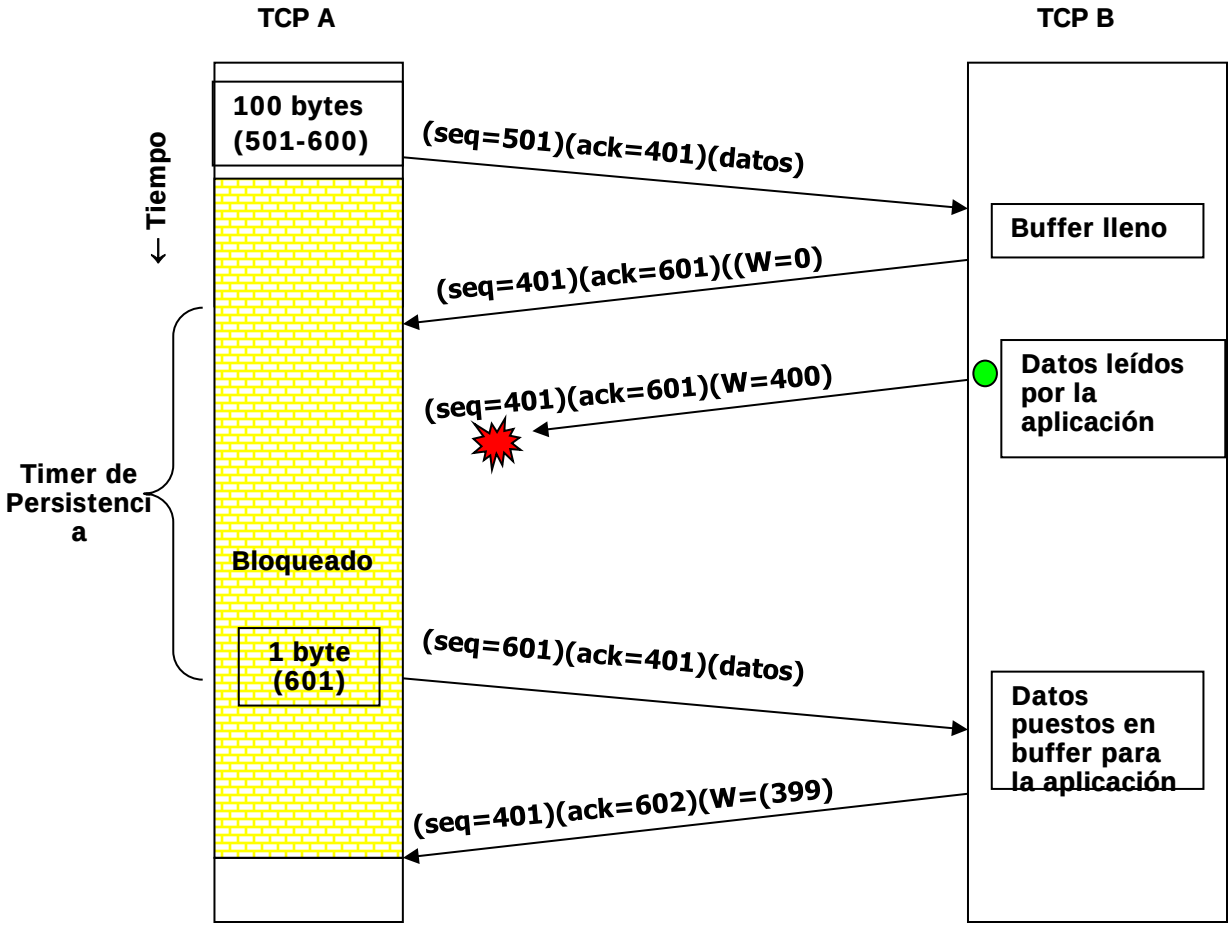


Timer de Persistencia

Mientras la ventana está cerrada el TCP emisor puede enviar de vez en cuando un segmento con un byte de datos; esto provoca el envío de un ACK por parte del receptor y evita el bloqueo que se podría producir en caso de pérdida de un segmento anunciando una ventana mayor que cero

La frecuencia con que el TCP emisor envía los reintentos se fija en el 'Timer de Persistencia'.

✓Timer de persistencia



Mensaje y timer de keepalive

Evita que se queden conexiones 'medio abiertas'

Se implementa reenviando el último byte transmitido en un segmento; el receptor descarta el dato pero devuelve un ACK

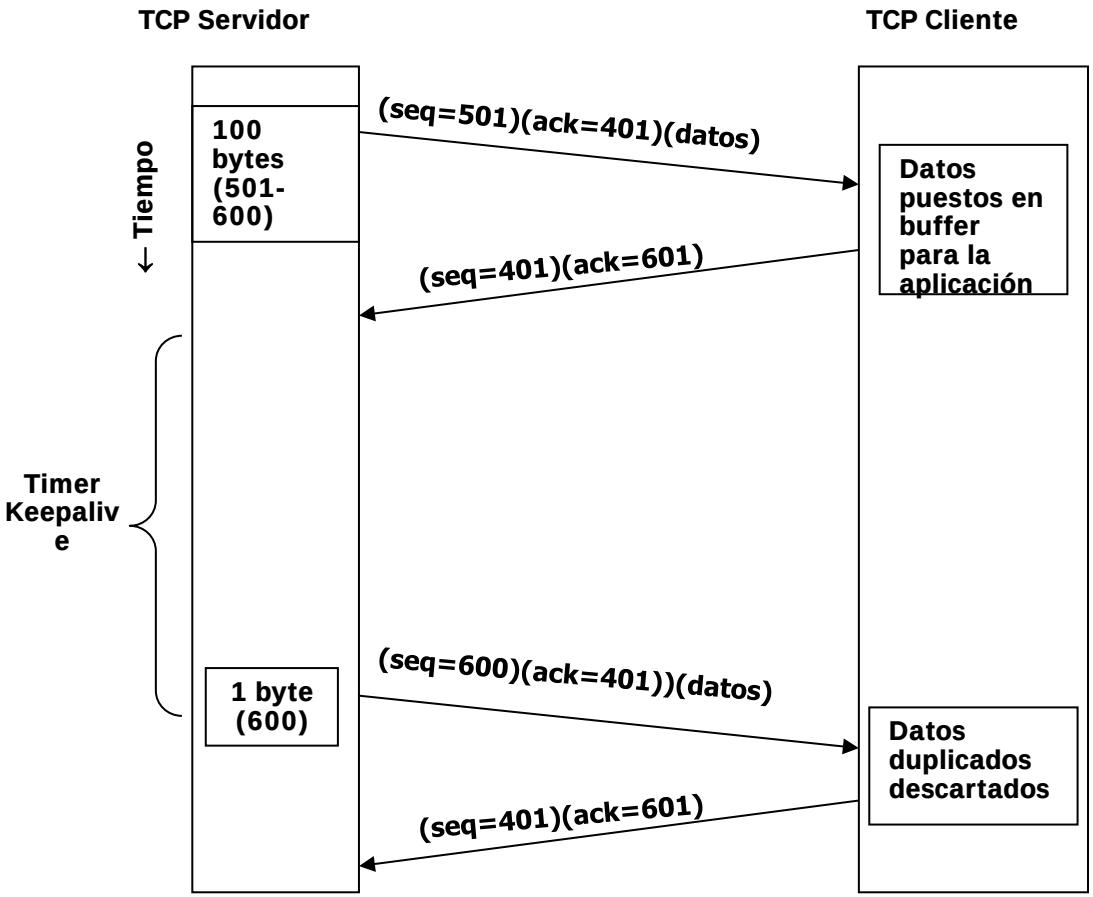
Si se envían varios mensajes de keepalive sin respuesta se considera que se trata de una conexión medio abierta y se cierra.

Para declarar una conexión medio abierta se espera a veces hasta 2 horas.

El tiempo de envío de los mensajes se regula con el timer de keepalive.

El keepalive no requiere modificaciones en el TCP receptor

✓Mensajes de keepalive



Baja eficiencia en TCP

- El funcionamiento eficiente de TCP aconseja enviar segmentos del tamaño máximo permitido
- Cuando la aplicación emisora genera los datos en pequeñas dosis (telnet con eco remoto por ejemplo) se da un problema de eficiencia que se resuelve con el **algoritmo de Nagle**.
- Si la aplicación receptora los recoge byte a byte también se puede dar una baja eficiencia; esto se conoce como síndrome de la ventana tonta y se resuelve con la **Solución de Clark**.

Algoritmo de Nagle

Cuando la aplicación envía datos en pequeños grupos TCP envía el primero y retiene los demás hasta recibir el ACK; por cada ACK recibido envía un segmento con los bytes que hubiera pendientes, y así sucesivamente.

También se envía un segmento cuando los datos acumulados igualan o superan el MSS (tamaño máximo de un segmento), o la mitad de la ventana.

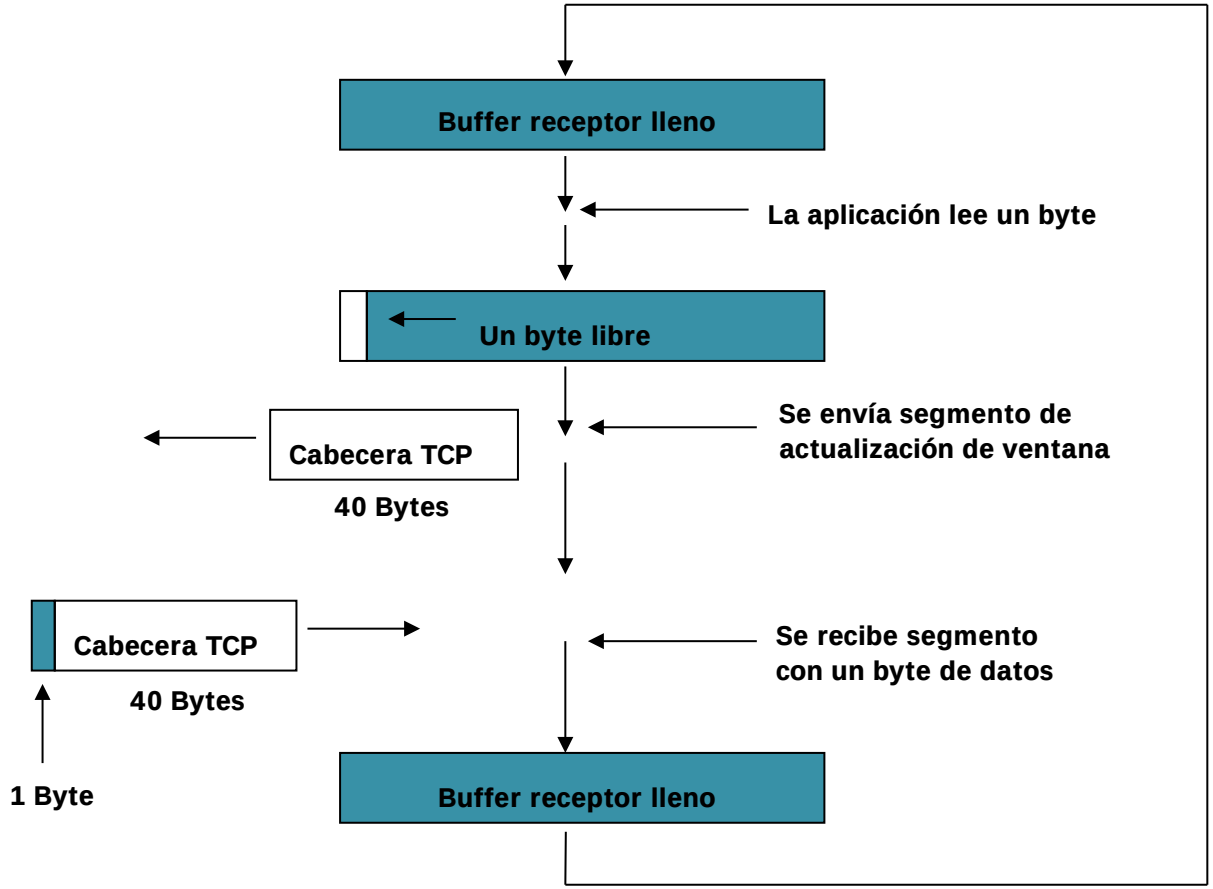
El mecanismo es autoadaptativo, pues cuanto más cargada esté la red mas tardarán los ACK y mas agrupados irán los datos

Se puede aplicar a datos pushed en caso necesario

Síndrome de la ventana tonta

1. La aplicación que envía datos los genera rápidamente
2. La aplicación receptora los recupera lentamente, un byte cada vez
3. El buffer del TCP receptor se llena
4. El TCP receptor notifica al emisor que su ventana está cerrada
5. La aplicación receptora lee un byte
6. El TCP receptor envía un ACK al emisor para anunciarle que dispone de un byte libre
7. El TCP emisor envía un segmento con un byte de datos
8. Volvemos al punto 3

✓Síndrome de la ventana tonta

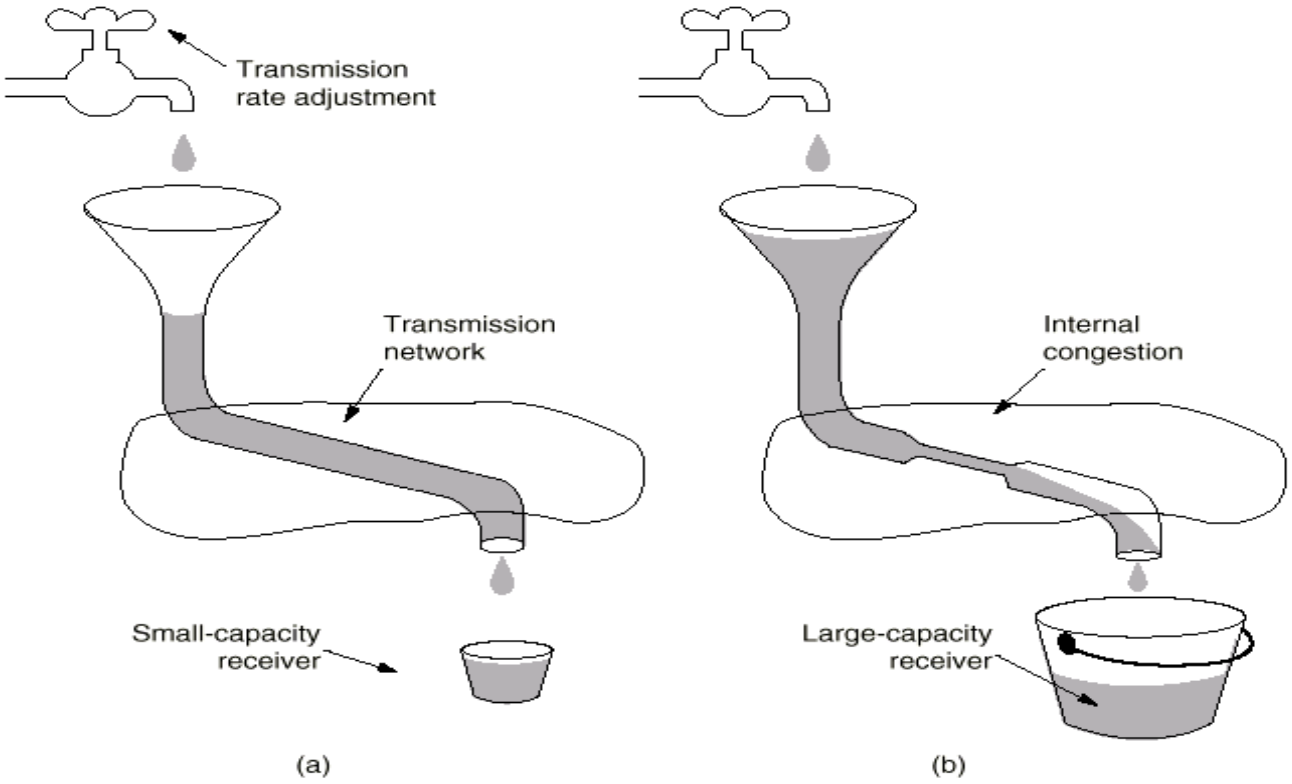


Solución de Clark (RFC 813)

- El TCP receptor solo debe notificar una nueva ventana cuando tenga una cantidad razonable de espacio libre. Razonable significa:
  - Un MSS (segmento del tamaño máximo), o
  - La mitad del espacio disponible en el buffer .Solución de Clark (RFC 813)

Control de la congestion

- Por medio del tamaño de ventana el receptor puede dosificar al emisor en función del buffer disponible. Esto es control de flujo.
- Pero puede que el receptor tenga espacio de sobra pero la red esté congestionada. En este caso el TCP debe regularse para no inyectar demasiado tráfico, a pesar de que la ventana disponible sea muy grande. Esto es control de congestión.
- Normalmente en TCP se utiliza control de congestión implícito. Ahora se está empezando a experimentar en Internet con el control de congestión explícito.



Control de flujo

Control de congestión

Fig. 6-31. (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver.

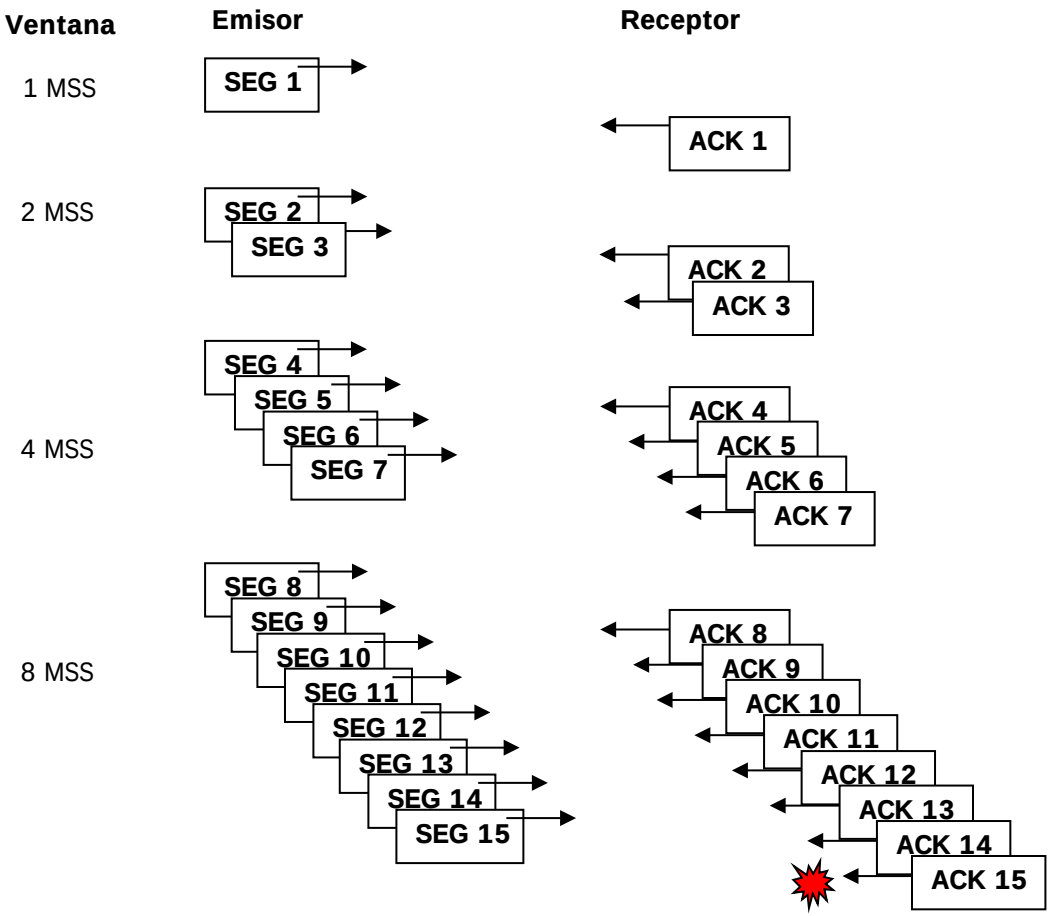
Control de congestión en TCP

- Cuando hay congestión TCP ha de reducir el flujo
- El mecanismo para detectarla es implícito, por la pérdida de segmentos. Cuando ocurre TCP baja el ritmo.
- Además de la ventana de control de flujo (dictada por el receptor y transmitida en la cabecera TCP) el emisor tiene una ventana de control de congestión, que ajusta a partir de los segmentos perdidos. En cada momento se usa la más pequeña de ambas.
- El mecanismo de control de congestión de TCP se denomina *slow-start* (arranque lento) y fue diseñado por Van Jacobson en los años 80.

Slow Start (primera fase)

- Inicialmente la ventana de congestión tiene el tamaño de un MSS (Maximum Segment Size)
- Por cada segmento enviado con éxito la ventana se amplía en un MSS
- En la práctica esto supone un crecimiento exponencial (en potencias de dos)
- Si la ventana de congestión supera a la de control de flujo se aplica ésta con lo cual aquella deja de crecer

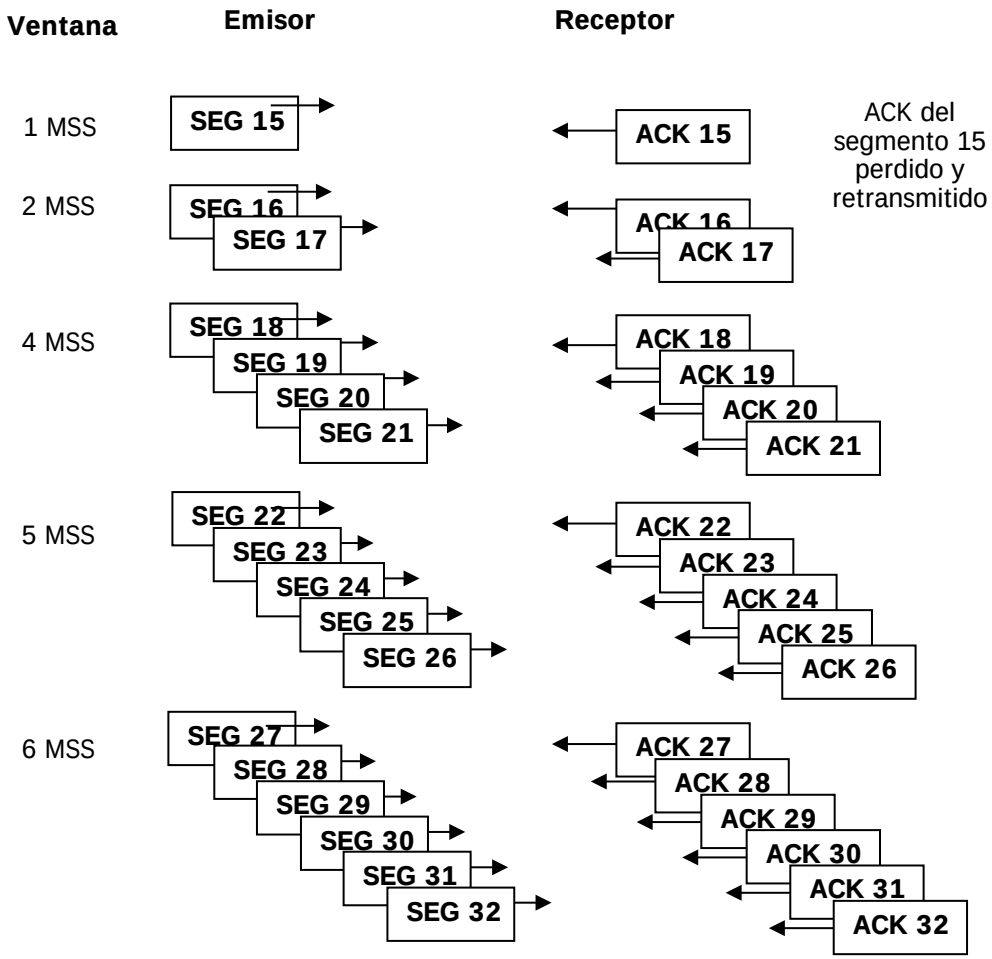
✓Funcionamiento de slow start, fase 1



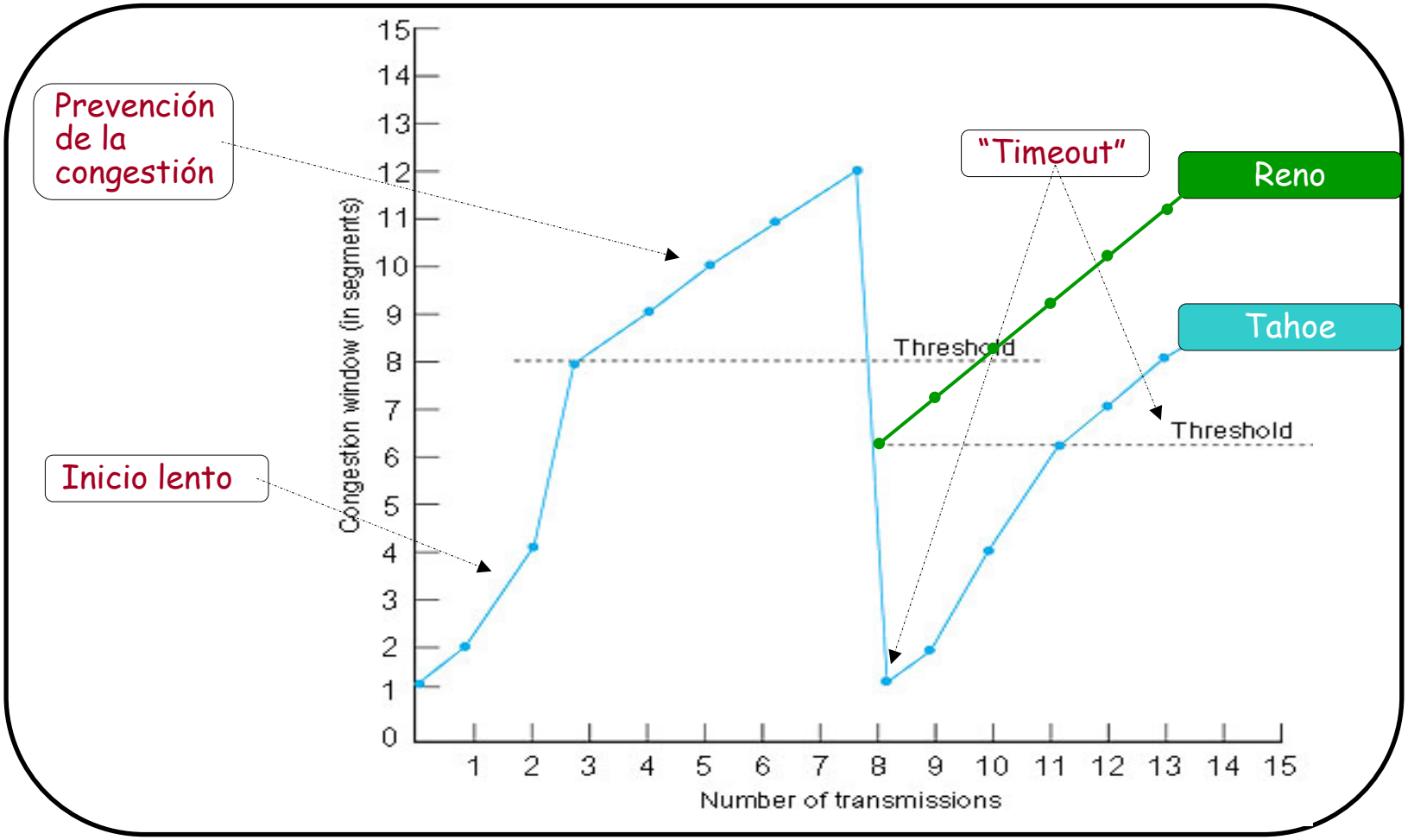
Slow start (segunda fase)

- Cuando se pierde un segmento:
  - La ventana de congestión vuelve a su valor inicial
  - Se fija un 'umbral de peligro' en un valor igual a la mitad de la ventana que había cuando se produjo la pérdida.
  - La ventana de congestión crece como antes hasta el umbral de peligro; a partir de ahí crece en sólo un segmento cada vez

✓ Funcionamiento de slow start, fase 2



Control de congestión (gráfica © James F. Kurose):



Control de congestión:

En el emisor se utilizan una **ventana** y un **umbral**.

TCP Tahoe

Inicialmente

VCongestion = MSS (maximum segment size)

Umbral a un cierto valor

Si *VCongestion* < *Umbral*, por cada ACK recibido

VCongestion += MSS **(crecimiento exponencial)**

Inicio lento

Si *VCongestion* > *Umbral*, por cada ventana completada (todos ACKs recibidos)

VCongestion += MSS (**crecimiento lineal**)

Prevención de la congestión

Si hay timeout entonces

Umbral = *VCongestion*/2

VCongestion = MSS

Timer de retransmisión

- Debe ser adecuado para la comunicación:
  - Si es excesivo se esperará innecesariamente
  - Si es muy corto se harán reenvíos innecesarios
- Como la fluctuación es muy grande se utilizan mecanismos autoadaptativos. A partir de los ACK se mide el tiempo de ida y vuelta o Round Trip Time (RTT)
- La estimación de este timer es crucial para el correcto funcionamiento del 'slow-start'.

Timer de retransmisión

Control de errores y de flujo:

Control de errores: ¿cómo estimar los "**timeouts**"?

***RTTmedido**: tiempo desde la emisión de un segmento hasta la recepción del ACK.*

$$RTTnuevo = (1-\alpha) \times RTTviejo + \alpha \times RTTmedido \;,\; \alpha \;\&\; \beta \in [0,1]$$

$$Desviacion_{nueva} = (1-\beta) \times Desviacion_{vieja} + \beta \times |\, RTTmedido - RTTnuevo \,|$$

$$Timeout = RTTnuevo + 4 * Desviacion$$

Problema con ACKs repetidos: ambigüedad en la interpretación.

Solución: Algoritmo de Karn:

- *actualizar el RTT sólo para los no repetidos*
- si hay que repetir un segmento incrementar el timeout: $tout_{nuevo} = \gamma \,.\, tout_{viejo}$, $\gamma = 2$.

1. Introducción.
2. Protocolo de datagrama de usuario (UDP).
3. Protocolo de control de transmisión (TCP).
    1. Multiplexación/demultiplexación.
    2. Control de conexión.
    3. Control de errores y de flujo.
    4. Control de congestión.
4. **Extensiones TCP.**
5. Ejercicios.

EXTENSIONES TCP

- TCP se define con mútiples "Sabores"
- Los diferentes sabores no afectan a la interoperabilidad entre los extremos
- Desde cualquier versión de Linux con kernel mayor que la 2.6.19 se usa por defecto TCP CuBIC

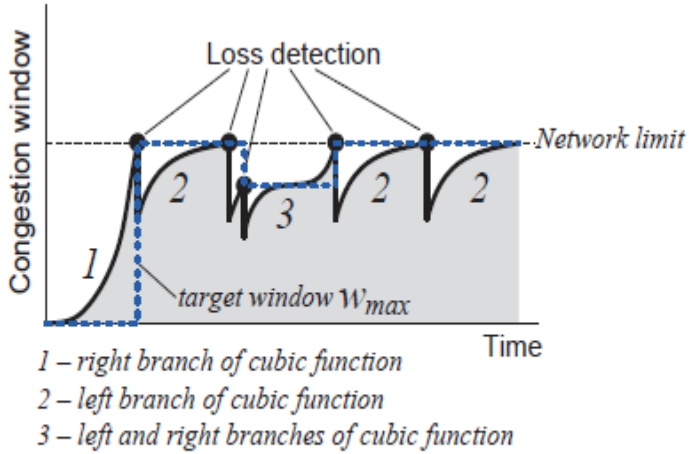
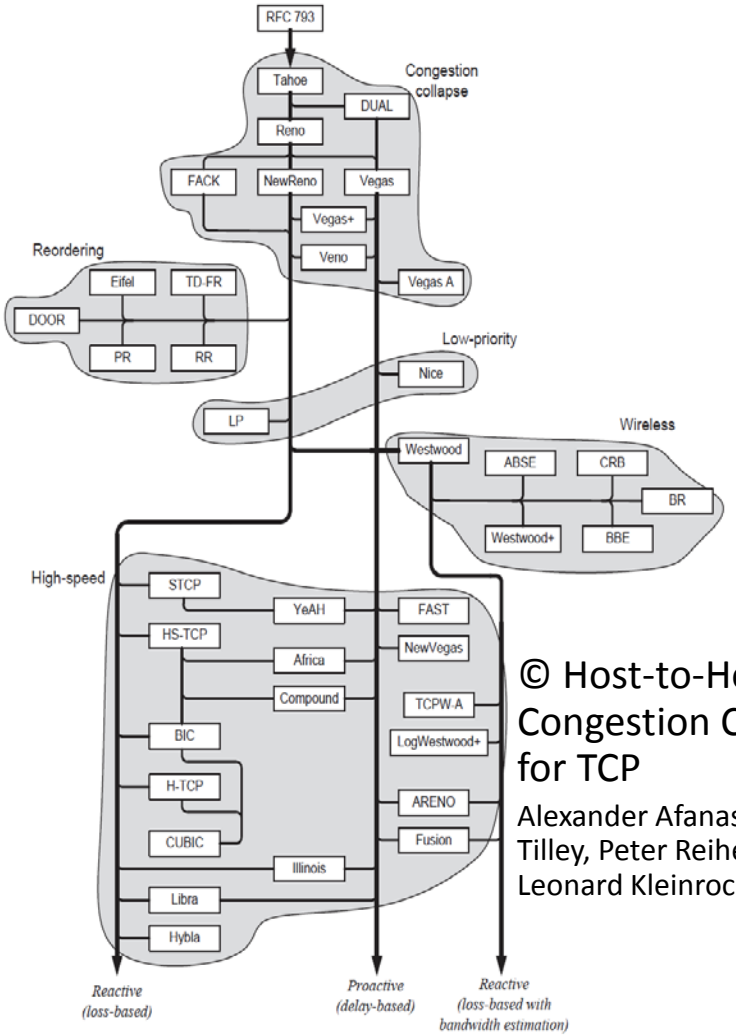


Fig. 50. Congestion window dynamics in CUBIC



57. Evolutionary graph of variants of TCP congestion control.

© Host-to-Host Congestion Control for TCP

Alexander Afanasyev, Neil Tilley, Peter Reiher, and Leonard Kleinrock

EXTENSIONES TCP

Adaptación de TCP a redes actuales (RFC 1323, 2018).

Ventana escalada:

Opción TCP en segmentos SYN:

Hasta 2^{14}x2^{16} bytes (=2^{30} bytes=1GB) autorizados.

Estimación RTT:

Opción TCP de *sello de tiempo*, en todos los segmentos.

PAWS ("Protect Against Wrapped Sequence numbers"):

Sello de tiempo y rechazo de segmentos duplicados.

SACK:

Confirmaciones selectivas.

TEMA 3
CAPA DE TRANSPORTE EN INTERNET

Fundamentos de Redes

2016/2017

TSTC CITIC-UGR ugr Universidad de Granada