Fundamentos de Redes

Tema 3

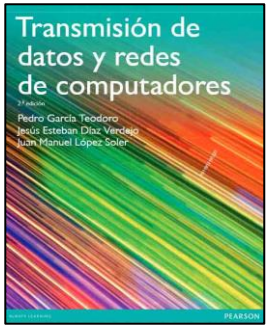
Capa de Transporte en Internet

Antonio M. Mora García

Bibliografía

Básica

- P. García-Teodoro, J.E. Díaz-Verdejo, J.M. López-Soler. Transmisión de datos y redes de computadores, 2ª Edición. Editorial Pearson, 2014. ***CAPÍTULO 10***



Complementaria

- James F. Kurose, Keith W. Ross. Redes de computadoras. Un enfoque descendente. 7º Edición. Editorial Pearson S.A., 2017.

  CAPÍTULO 3

Índice

TEMA 3. Capa de transporte en Internet

- **3.1. Introducción a los protocolos de Capa de Transporte**
- 3.2. Protocolo de datagrama de usuario (UDP)
- 3.3. Protocolo de control de transmisión (TCP)
  - Multiplexación/demultiplexación
  - Control de conexión
  - Control de errores y de flujo
  - Control de congestión
- 3.4. Extensiones TCP
- 3.5. Cuestiones y 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, se pueden utilizar **varios servicios** (correo electrónico, acceso Web, mensajería instantánea, etc).
- Los **datos de cada** una de estas **aplicaciones** se empaquetan, se transportan y **se entregan al servidor adecuado** o aplicación en el **dispositivo de destino**.
- La función principal de la **capa de transporte** es aceptar los **datos de las capas superiores**, dividirlos en unidades más pequeñas si es necesario, y **pasarlos a la capa de red** garantizando que lleguen a su destino independientemente la red o redes físicas que utilicen.

Capa de transporte

- **Entidades de transporte**: hardware y/o software que se encargan de realizar este trabajo.

- La **comunicación** entre entidades de **transporte** es **extremo a extremo** (*end-to-end*). Es decir, se produce **entre el emisor/receptor finales**, no teniendo en cuenta a ningún otro dispositivo intermedio de las subredes.

- El nivel de **transporte mejora** la calidad del **servicio** ofrecida por el **nivel de red** mediante:

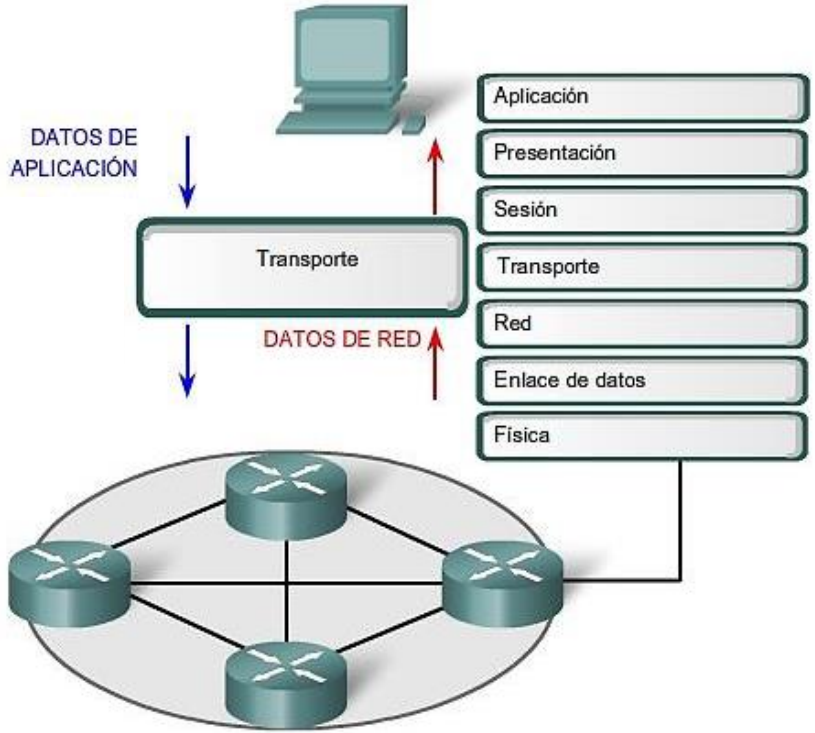
  - La multiplexación/demultiplexación
  - La introducción de redundancia en la información

- Dos **protocolos**: **TCP** y **UDP**.

Capa de transporte

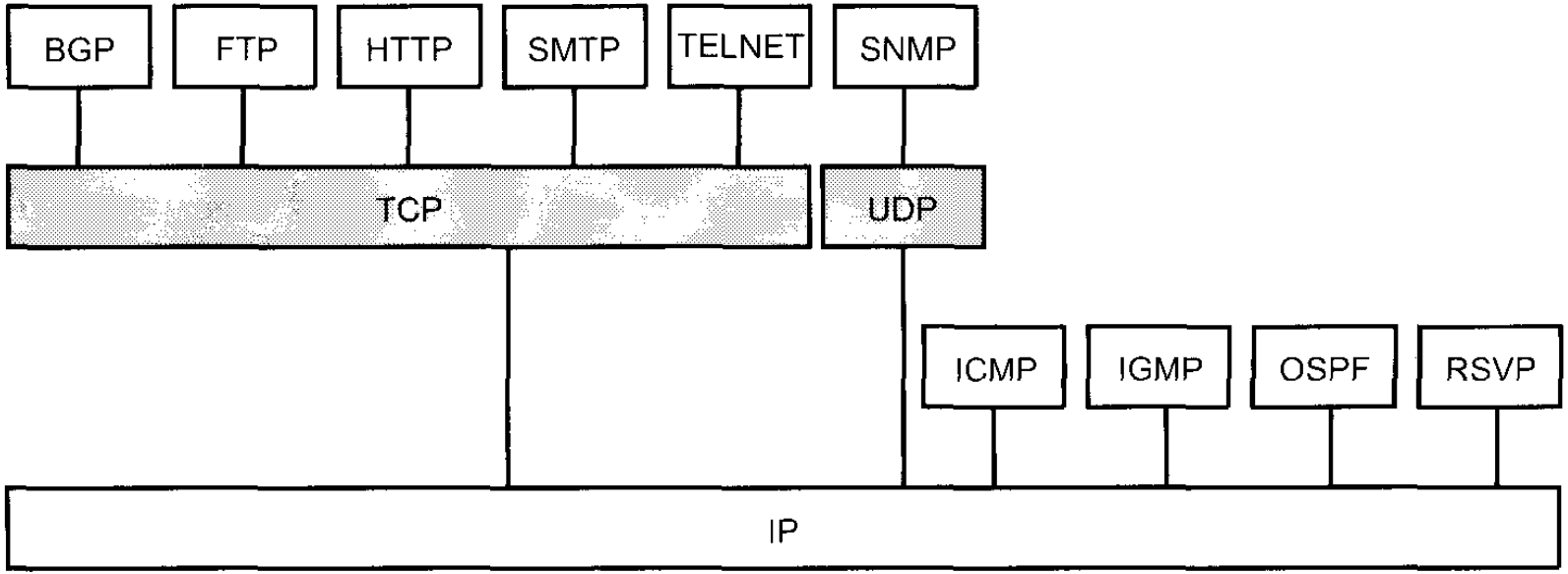
- La capa de transporte es el **enlace** entre la **capa de aplicación** y la **capa** responsable de la **transmisión en la red**:
  - **Red** en modelo OSI
  - **Internet** en modelo TCP/IP

- Prepara los datos de la aplicación para su transporte en la red y procesa los datos recibidos por la red para su uso en las aplicaciones.



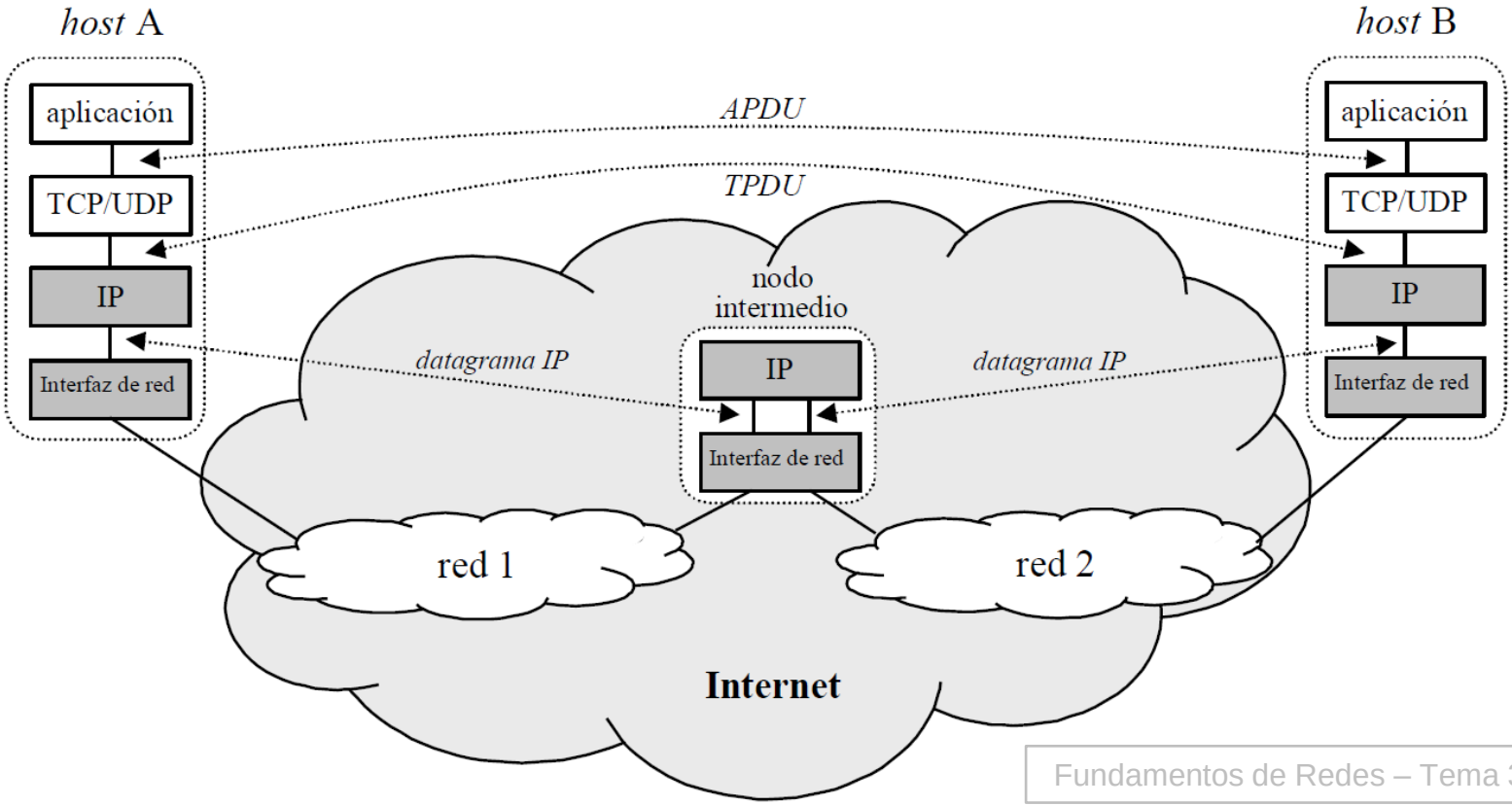
Capa de transporte

- **Protocolos** de las capas de **aplicación** y **transporte**.



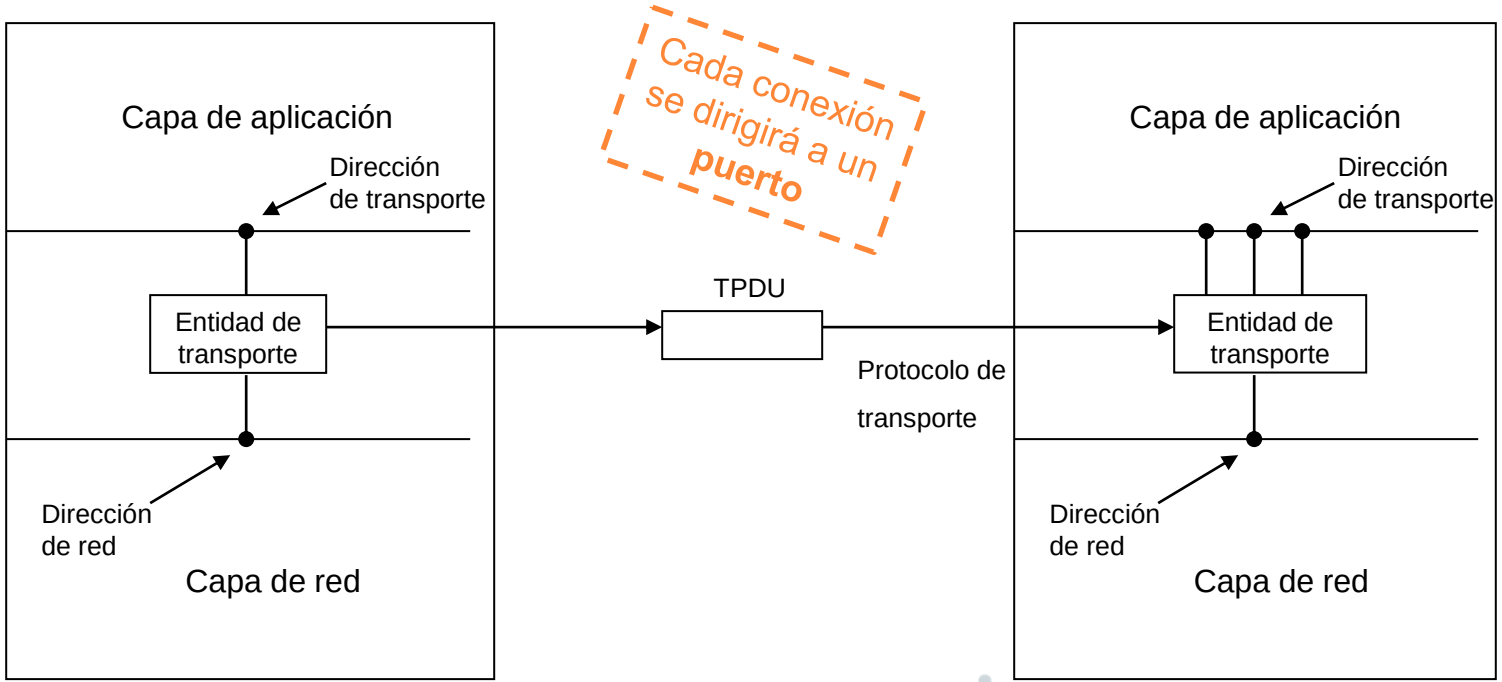
Capa de transporte

- La **comunicación** entre entidades de **transporte** es **extremo a extremo** (*end-to-end*).



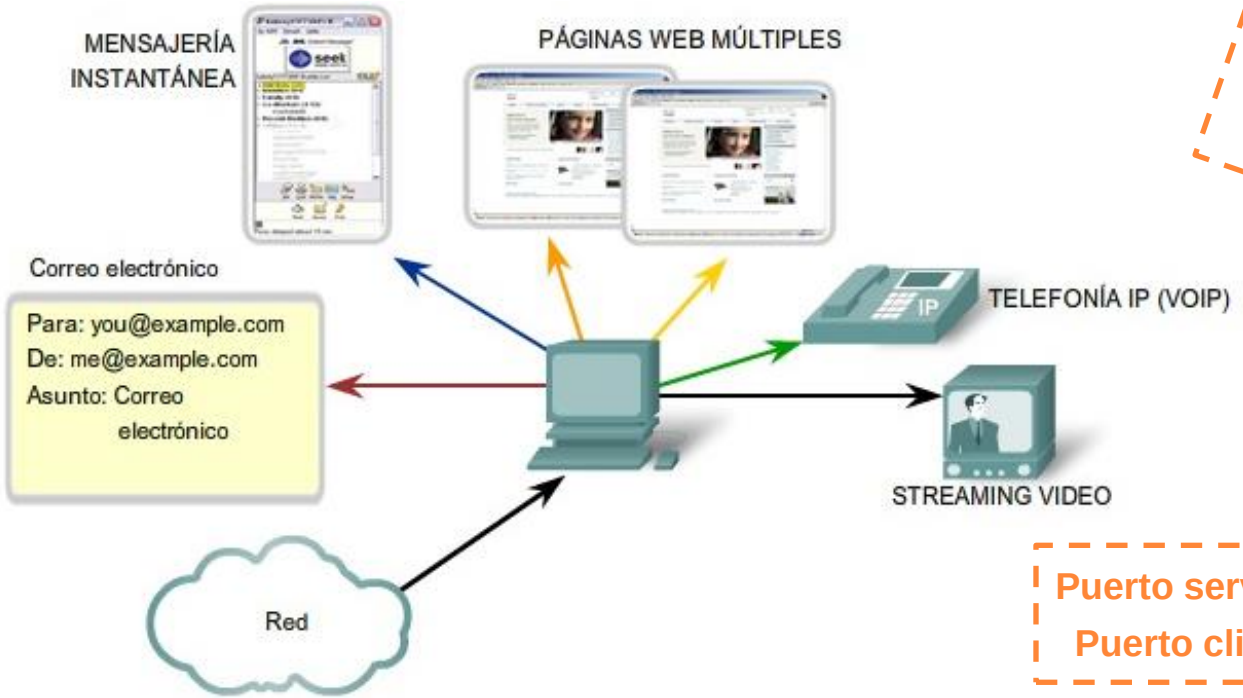
Capa de transporte

- Permite realizar **multiplexación de comunicaciones** (de aplicaciones).



Capa de transporte

- Permite realizar **multiplexación de comunicaciones** (de aplicaciones).



Cada aplicación se dirigirá a un **puerto**

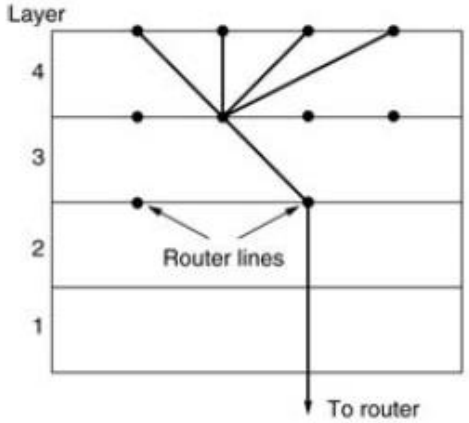
Puerto servidor → conocido
Puerto cliente → aleatorio

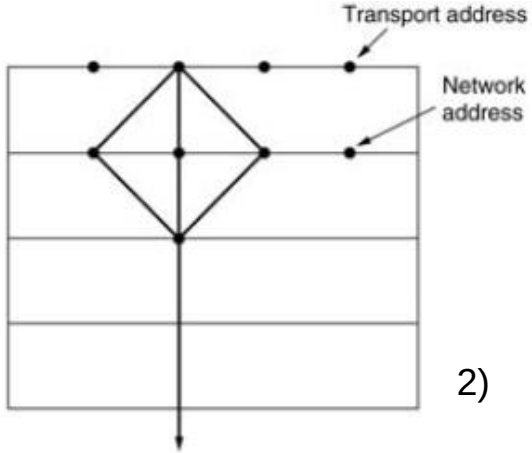
Capa de transporte

- **Multiplexación/Demultiplexación**:

1) Varias conexiones de transporte en una conexión de red. Se utiliza cuando el coste por conexión del servicio de red es elevado.
2) 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.

1)





2)

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:
  - Seguimiento de la comunicación individual entre origen y destino.
  - Segmentación de datos (de aplicación) y manejo de cada parte.
  - Reensamblaje de segmentos.
  - Identificación de diferentes aplicaciones en origen y destino.
  - Multiplexación y Demultiplexación del tráfico de las aplicaciones.

Propósito de la capa de transporte

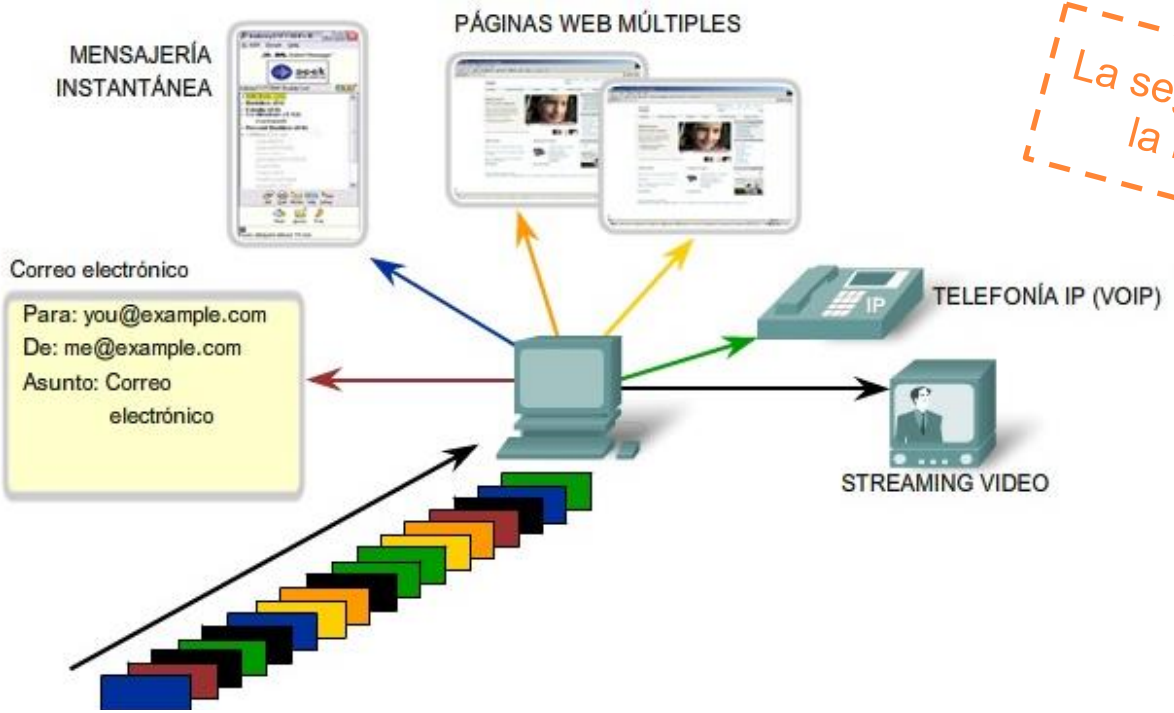
- **Segmentación de datos y manejo de cada parte**.
  - Cada **aplicación** crea **datos para enviarse** a una aplicación remota.
  - Estos **datos** se deben **preparar para ser enviados** a través de los medios.
  - Los protocolos de la **capa de transporte** describen **los servicios que segmentan** 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**.

La segmentación facilita la multiplexación

Los segmentos son más fáciles de administrar y controlar (errores, flujo, etc)

Propósito de la capa de transporte

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



La segmentación facilita la multiplexación

Propósito de la capa de transporte

- **Reensamblaje de segmentos**:
  - Al recibirlo, cada **segmento** de datos **se traslada** a la **aplicación adecuada**.
  - Los **segmentos** de datos individuales **deben unirse para reconstruir 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**.

- **Restricciones**:
  - Los segmentos/datos deberán llegar en una secuencia específica para ser usados por la aplicación.
  - Se espera recibir todos los datos, aunque algunas aplicaciones toleran pérdidas.

Propósito 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 cada aplicación final.
  - La capa de transporte **asigna** un **identificador a la aplicación**.
  - Los protocolos TCP/IP denominan a este identificador **número de puerto**.

Puertos

ASIGNACIÓN DE PUERTOS

- **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 especificadas se conocen como **asignaciones bien conocidas**.

- **Dinámica**: siempre que un **proceso** necesita un puerto el **software de red le asignará uno**. Se asigna de forma aleatoria dentro de un rango y evitando los puertos bien conocidos.

- Los diseñadores de TCP/IP adoptaron una solución híbrida, que preasigna muchos números de puerto pero que también deja muchos de ellos disponibles.

Puertos

ASIGNACIÓN DE PUERTOS - 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 utilizado si el emisor no permite respuestas del receptor.
- Los **puertos 1 a 1023** reciben el nombre de **Puertos bien conocidos**.

  (En sistemas Unix, para enlazar con ellos (‘abrirlos’), 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 49152 al 65535** son **Puertos efímeros**, de tipo temporal, y se **utilizan** sobre todo por los **clientes** al conectar con el servidor.

Puertos

PUERTOS BIEN CONOCIDOS (I) (RFC 6335)

20 (TCP), utilizado por FTP (File Transfer Protocol) para datos
21 (TCP), utilizado por FTP (File Transfer Protocol) para control
22 (TCP), utilizado por SSH (Secure Shell)
23 (TCP), utilizado por TELNET (Teletype Network)
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)

Puertos

PUERTOS BIEN CONOCIDOS (II) (RFC 6335)

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

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 (puertos).
  - Servicio **no orientado a conexión**, **no fiable**.

- Protocolo **TCP**:
  - **Multiplexación/demultiplexación** de aplicaciones (puertos).
  - Servicio **orientado a conexión**, **fiable**:
    . Control de **errores**
    . Control de **flujo**
    . Control de **congestión**

TEMA 3. Capa de transporte en Internet

- 3.1. Introducción a los protocolos de Capa de Transporte
- **3.2. Protocolo de datagrama de usuario (UDP)**
- 3.3. Protocolo de control de transmisión (TCP)
  - Multiplexación/demultiplexación
  - Control de conexión
  - Control de errores y de flujo
  - Control de congestión
- 3.4. Extensiones TCP
- 3.5. Cuestiones y ejercicios

Introducción

- El **protocolo UDP** (***User Datagram Protocol***) se define en la RFC 768.
- Proporciona un servicio de **entrega de datagramas**:

- no orientado a conexión:

. sin conexión previa (*no hand-shaking*).

. no hay retardo de establecimiento de la conexión.

. cada TPDU (datagrama) es independiente.

Funcionalidad **"Best Effort"**

- no confiable:

. no se comprueban errores.

. puede haber pérdidas de paquetes.

- no hay garantía de **entrega ordenada**.

- no hay **control de congestión** (se entrega tan rápido como se pueda).

- multiplexación/demultiplexación (transportar el TPDU al proceso/aplicación correcto).

Introducción

- **UDP utiliza IP** (capa Red), pero agrega la capacidad para distinguir entre varias aplicaciones de destino dentro de un mismo sistema destino.

- Una **aplicación** que utiliza UDP, **asume la responsabilidad** por los **problemas** de **confiabilidad**, incluyendo la pérdida, duplicación y retraso de los paquetes, 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.

Utilización

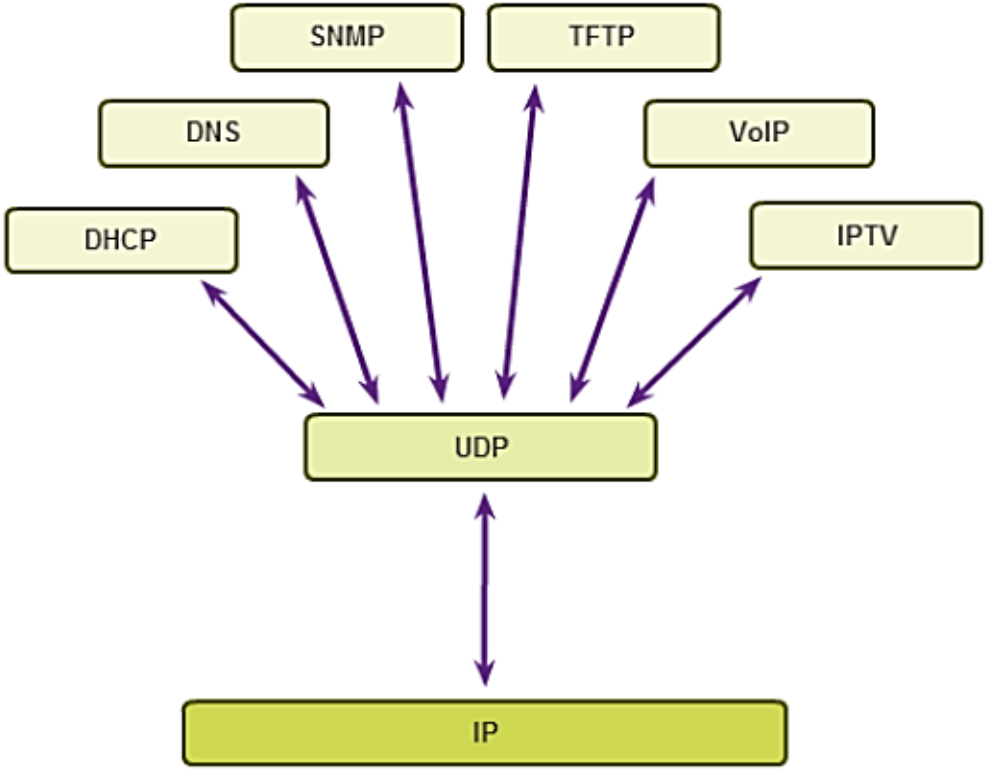
- **UDP suele utilizarse para:**
  - Aplicaciones de streaming multimedia (tolerantes a pérdidas, pero sensibles a retardos).
  - Intercambio de mensajes (escaso). Ej: Consultas DNS (< 512 bytes).
  - Aplicaciones en tiempo real (no pueden esperar confirmaciones).

    Ej: videoconferencia, voz sobre IP.
  - Mensajes producidos periódicamente, ya que no importa si se pierde alguno.

    Ej: SNMP (*Simple Network Management Protocol*)
  - Para el envío de tráfico broadcast/multicast.

Utilización

- **UDP suele utilizarse para:**



Formato Datagrama UDP

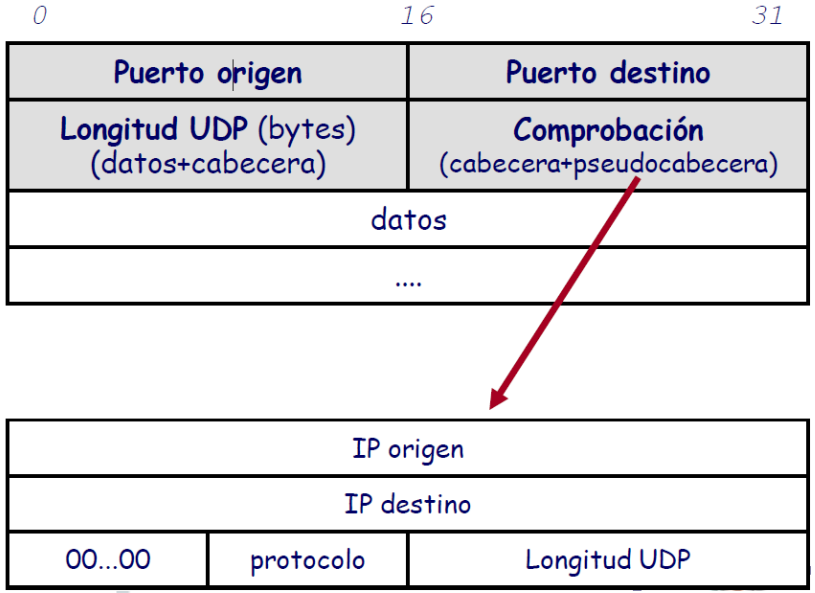
- **Cabecera:**

  - Los números de puerto identifican los procesos emisor y receptor

  - Longitud UDP → longitud de la cabecera UDP + longitud de datos (valor mínimo 8 bytes)

  - Datos → PDU de la capa superior

Puertos UDP diferentes de Puertos TCP



Formato Datagrama UDP

- **Cabecera:**

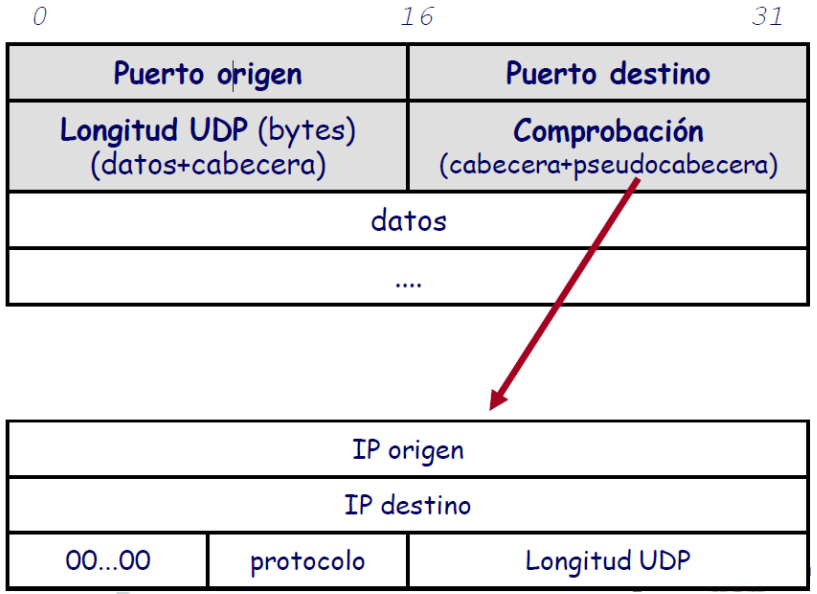
- Comprobación (*Checksum*):

  . se calcula sobre la cabecera UDP y los datos UDP.

  . complemento a 1 de la suma de todo el datagrama.

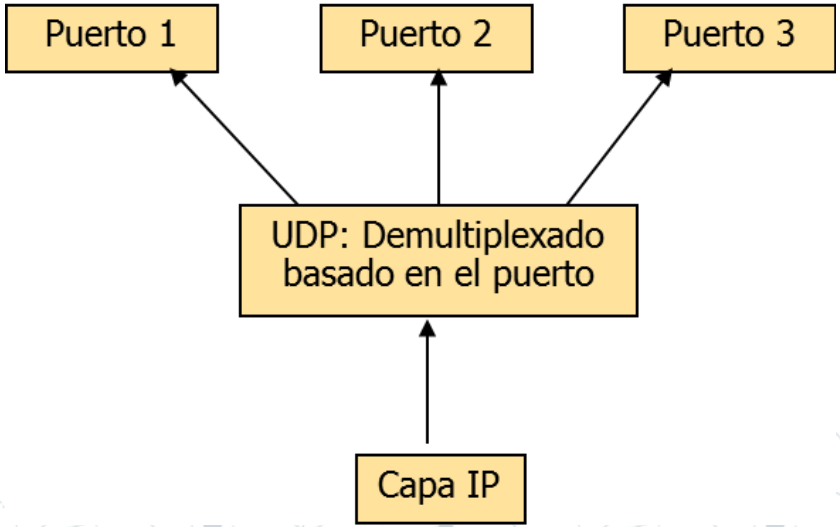
  . El datagrama UDP puede contener un número impar de bytes → Se añade un byte de relleno (todo ceros).

  . 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)



Puertos UDP

- **UDP** acepta **datagramas** de **muchos programas** de aplicación.
- **Pasa los datagramas** al nivel de **red IP para** su **transmisión** y los recibe de ese nivel en el otro extremo.
- El **multiplexado y demultiplexado** entre el software UDP y los **programas de aplicación** ocurre **a través** del mecanismo **de puerto** (identificación).



Puertos UDP diferentes de Puertos TCP

TEMA 3. Capa de transporte en Internet

- 3.1. Introducción a los protocolos de Capa de Transporte
- 3.2. Protocolo de datagrama de usuario (UDP)
- **3.3. Protocolo de control de transmisión (TCP)**
  - Multiplexación/demultiplexación
  - Control de conexión
  - Control de errores y de flujo
  - Control de congestión
- 3.4. Extensiones TCP
- 3.5. Cuestiones y ejercicios

Características del *Transmission Control Protocol*

- RFC 793 (1122, 1323, 2018, 2581).
- Servicio **orientado a conexión**: exige un acuerdo entre emisor y receptor (*hand shaking*).
- **Entrega ordenada**: de las secuencias de bytes generadas por las aplicaciones (*stream oriented*).
- Transmisión ***full duplex***: se pueden enviar datos en ambos sentidos al mismo tiempo.
- Mecanismo de **detección y recuperación de errores** (***ARQ*** – *Automatic Repeat reQuest*):
  - con confirmaciones positivas (***ACKs***) acumulativas.
  - ***timeouts*** adaptables.
  - incorporación de confirmaciones con los datos (***piggybacking***)
- Servicio **fiable**: **control de congestión** y **control de flujo** con **ventanas deslizantes** con tamaño máximo adaptable.
- Servicio **punto a punto**: no puede usarse para multicast.

Características del *Transmission Control Protocol*

SERVICIOS QUE OFRECE TCP

- **Establecimiento y cierre de la conexión**:

  - Al ser un protocolo orientado a conexión, dispone de mecanismos para establecer la conexión (antes de la transmisión de datos) y para cerrarla (al final de la transmisión).

- **Control de errores y de flujo**:

  - Se garantiza la recepción correcta y ordenada de los datos en la aplicación destino tal y como los generó la aplicación origen.

  - Es capaz de ajustar las diferencias que haya entre la tasa de generación de datos (en el origen) y la de consumo de los mismos (destino).

- **Control de congestión**:

  - Gestiona los recursos de la red (ancho de banda, almacenamiento temporal en los routers) para evitar su agotamiento, adaptando el tráfico a generar.

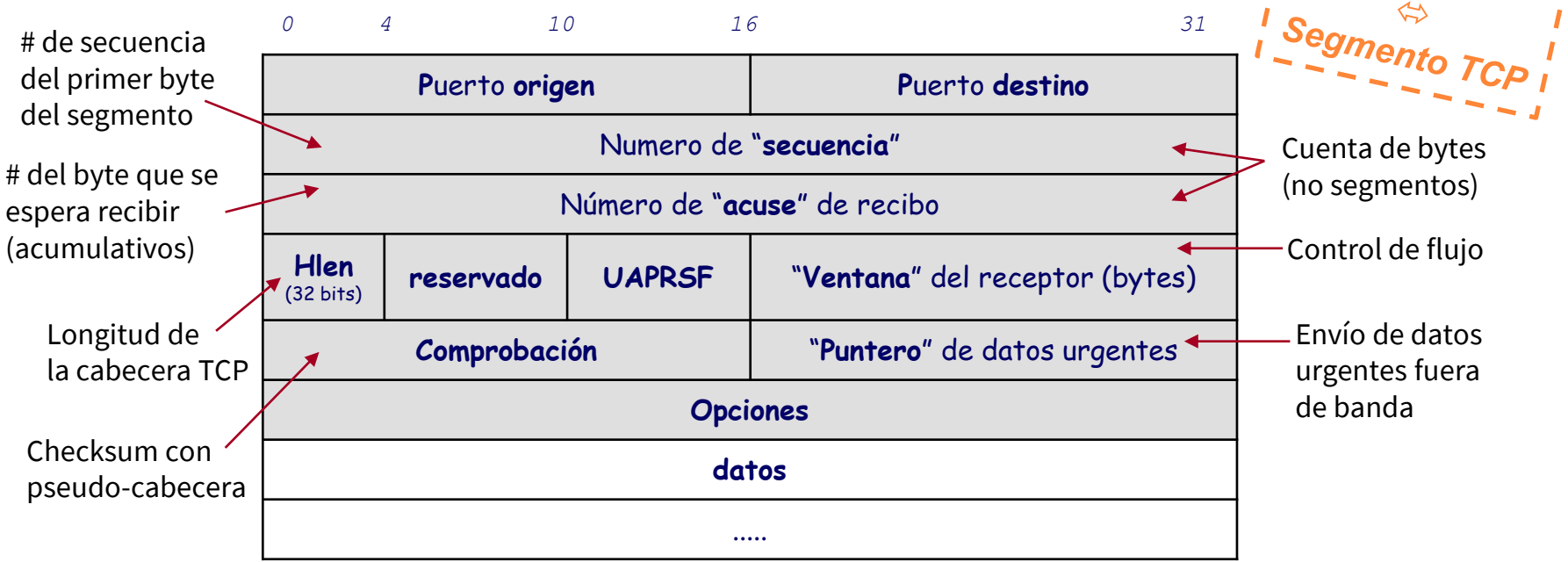
- **Multiplexación de aplicaciones**:

  - Al igual que UDP, utiliza puertos para dirigir los datos a las aplicaciones pertinentes en el destino.

Características del *Transmission Control Protocol*

CABECERA TCP

TPDU TCP ⇔ Segmento TCP



- Cada **segmento TCP** se **encapsula** en un **datagrama IP**.

TEMA 3. Capa de transporte en Internet

- 3.1. Introducción a los protocolos de Capa de Transporte
- 3.2. Protocolo de datagrama de usuario (UDP)
- **3.3. Protocolo de control de transmisión (TCP)**
  - **Multiplexación/demultiplexación**
  - Control de conexión
  - Control de errores y de flujo
  - Control de congestión
- 3.4. Extensiones TCP
- 3.5. Cuestiones y ejercicios

Multiplexación/Demultiplexación

- Consiste en **transportar** los **segmentos** a la **aplicación correcta**.
- Se realiza (al igual que en UDP) utilizando **puertos asociados a cada aplicación**.
- Existen **puertos preasignados**:

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

Puertos TCP diferentes de Puertos UDP

Una conexión TCP se identifica por:
IP y puerto origen
IP y puerto destino

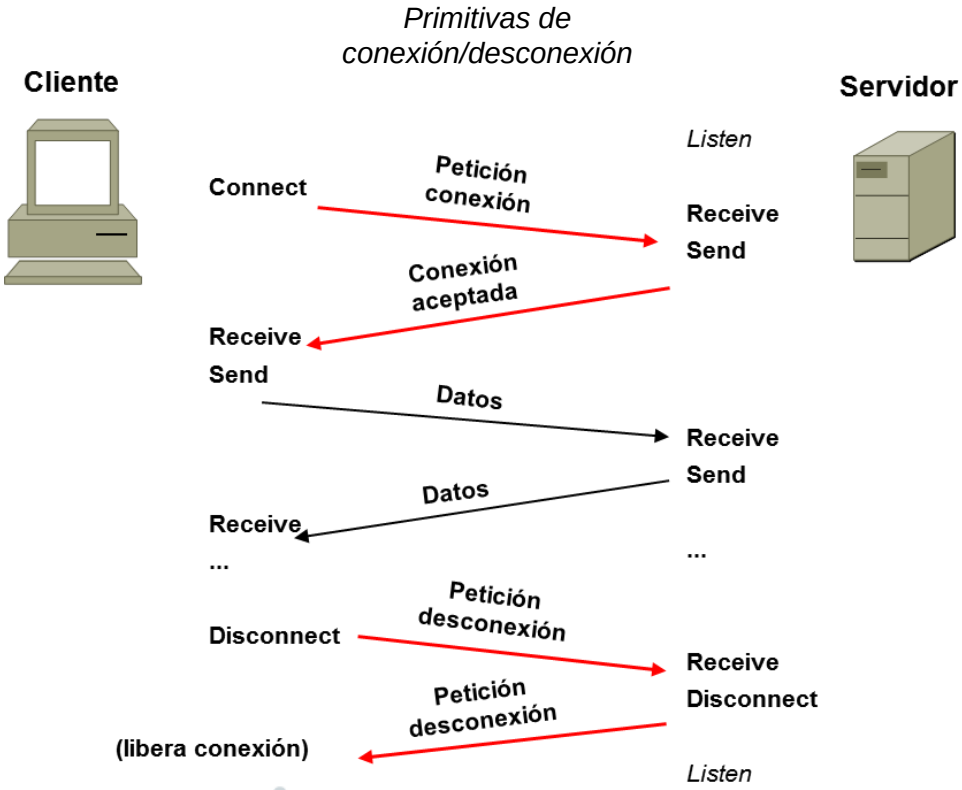
TEMA 3. Capa de transporte en Internet

- 3.1. Introducción a los protocolos de Capa de Transporte
- 3.2. Protocolo de datagrama de usuario (UDP)
- **3.3. Protocolo de control de transmisión (TCP)**
  - Multiplexación/demultiplexación
  - **Control de conexión**
  - Control de errores y de flujo
  - Control de congestión
- 3.4. Extensiones TCP
- 3.5. Cuestiones y ejercicios

Control de la conexión

- TCP es **orientado a conexión**.
- El intercambio de información tiene **tres fases**:
  - Establecimiento de la conexión (sincronizar # de secuencia y reservar recursos).
  - Intercambio de datos (full-duplex).
  - Cierre de la conexión (liberar recursos).

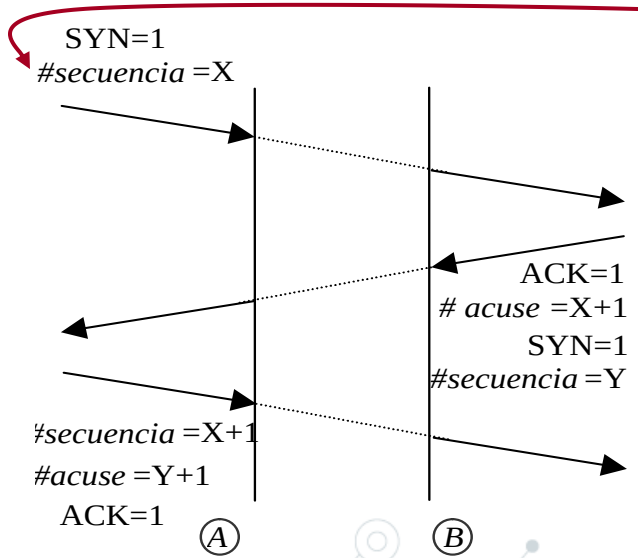
Es un mecanismo de sincronización entre emisor y receptor. ***Para garantizar comunicación ordenada y sin errores***



Control de la conexión

ESTABLECIMIENTO

- ¿Se podría **garantizar** un establecimiento/cierre **fiable** de la conexión sobre un servicio no fiable (IP)? **NO**
- Por ello se establece la conexión con ***three-way handshaking***.



"Apertura activa": cliente
"Apertura pasiva": servidor
Campos involucrados:
* *Bit S (SYN) del campo control*
* *Campo secuencia (valor aleatorio)*
* *Campo acuse (secuencia recibida + 1)*
* *Bit A (ACK) del campo control*

Control de la conexión

NÚMEROS DE SECUENCIA

- 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 "teóricamente" al azar; para evitar confusiones con transmisiones anteriores.

- 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.

Control de la conexión

NÚMEROS DE SECUENCIA

- El **mecanismo** de **selección** de los **ISN** es suficientemente **fiable** para proteger de **coincidencias**, 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.

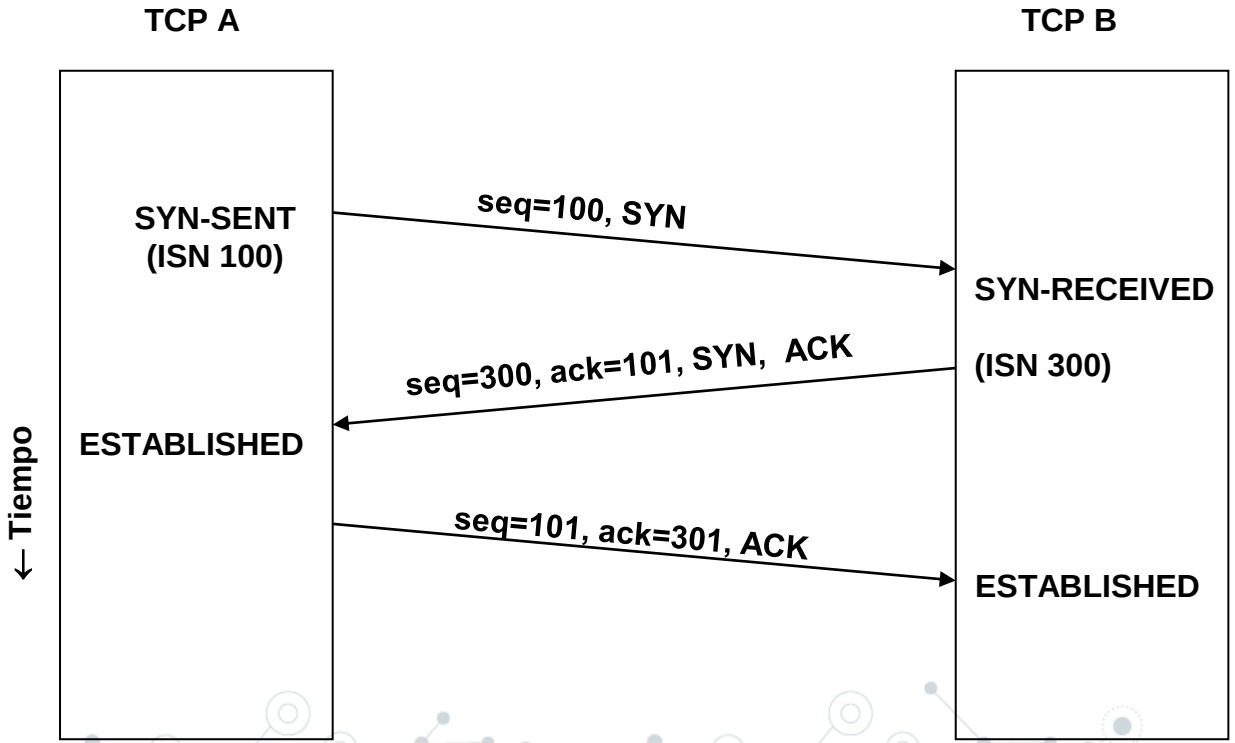
- TCP **incrementa el número de secuencia** de cada segmento **según los bytes** que tenía el **segmento anterior**, con una excepción:

  - Cuando los flags SYN y FIN están activos, se incrementa en 1 el número de secuencia.

- La presencia además del flag ACK activo implica que no se incrementa el número de secuencia (no hay datos).

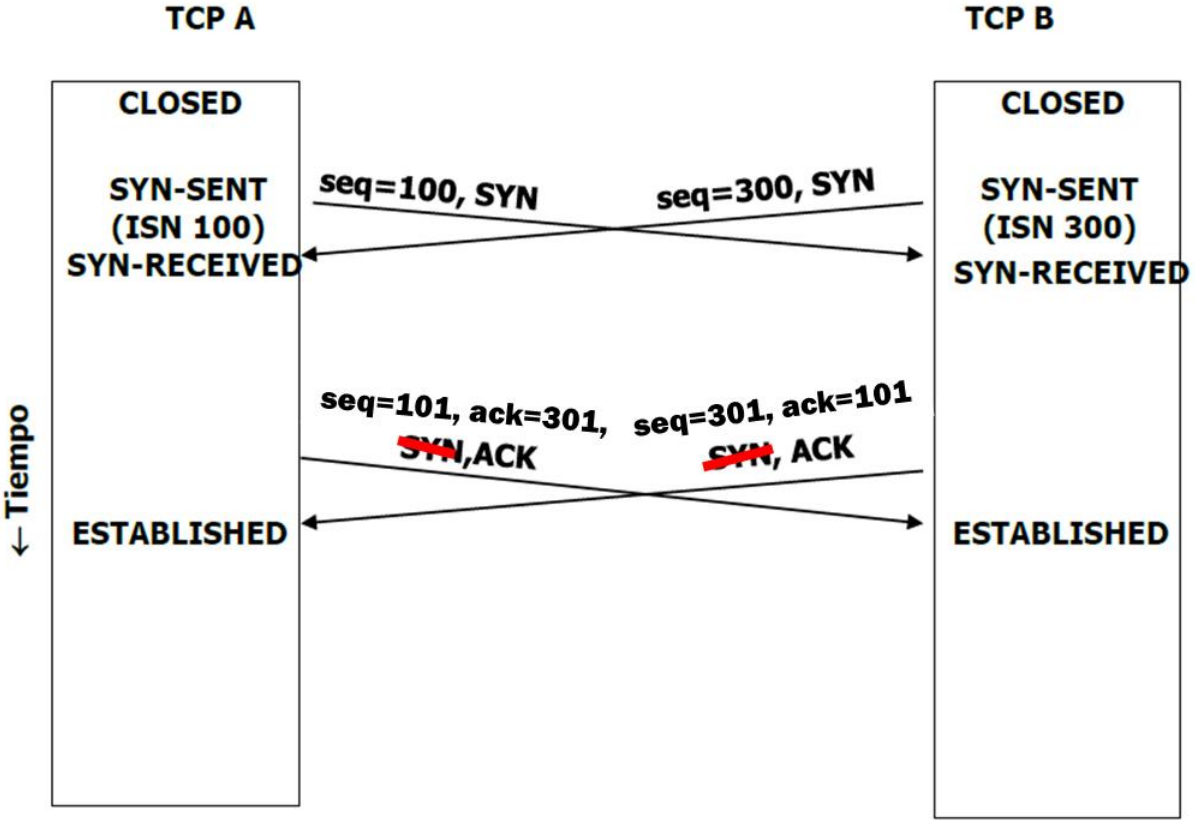
Control de la conexión

- Ejemplo: ***three-way handshaking***



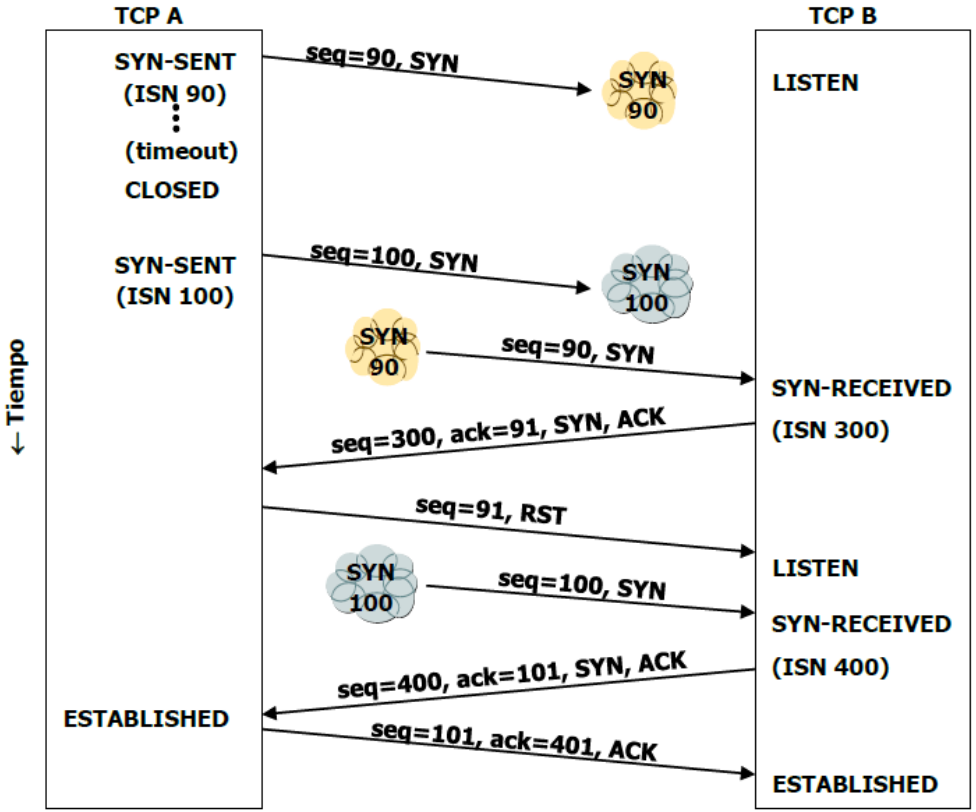
Control de la conexión

- Ejemplo: ***three-way handshaking (Conexión simultánea)***



Control de la conexión

- Ejemplo: ***three-way handshaking (SYN retrasados y duplicados)***

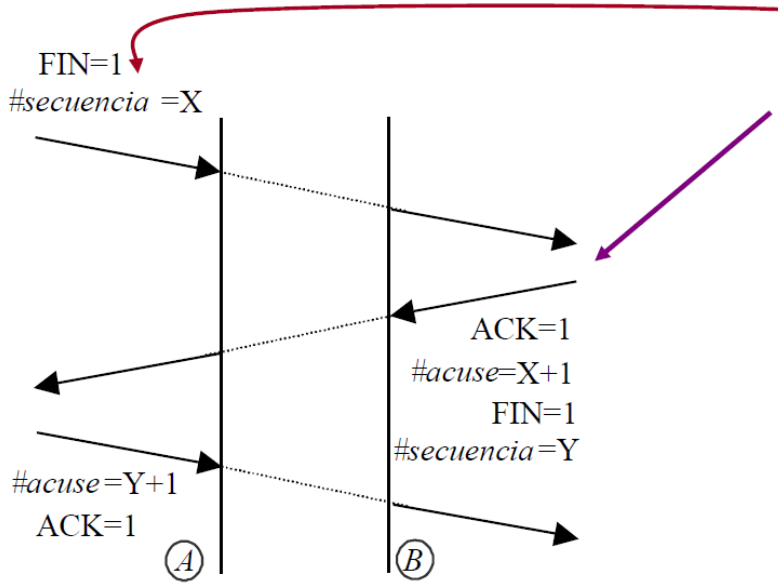


RST → señal de cancelación de la conexión

Control de la conexión

CIERRE

- Sincronización para el cierre de la conexión y liberación de recursos asociados a la misma.
- Una vez comenzado el procedimiento de cierre no se cierra inmediatamente por si hay paquetes en tránsito, sino que se usan *timeouts*.



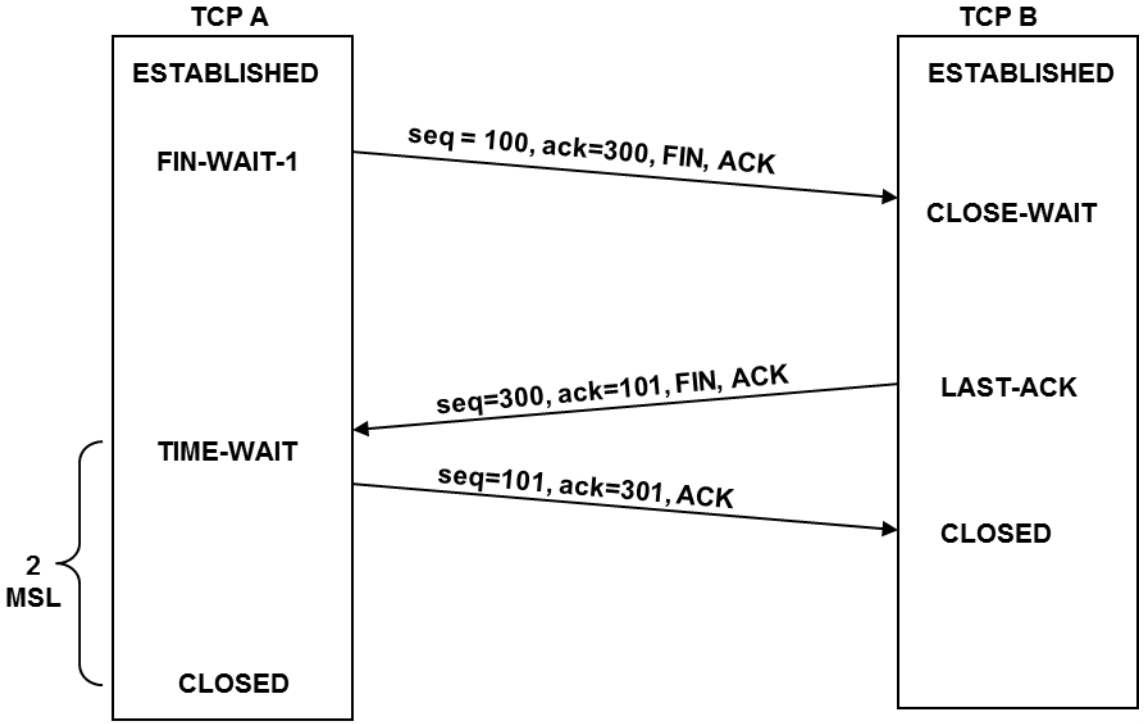
- "Cierre activo"
- "Cierre pasivo"
- Campos involucrados:
  - * Bit F (FIN) del campo control
  - * Campo secuencia
  - * Campo acuse
  - * Bit A (ACK) del campo control

Timeout ⇔
MSL (*Maximum Segment Lifetime*)

2 minutos

Control de la conexión

- Ejemplo: cierre habitual en tres pasos (con *timeout)*

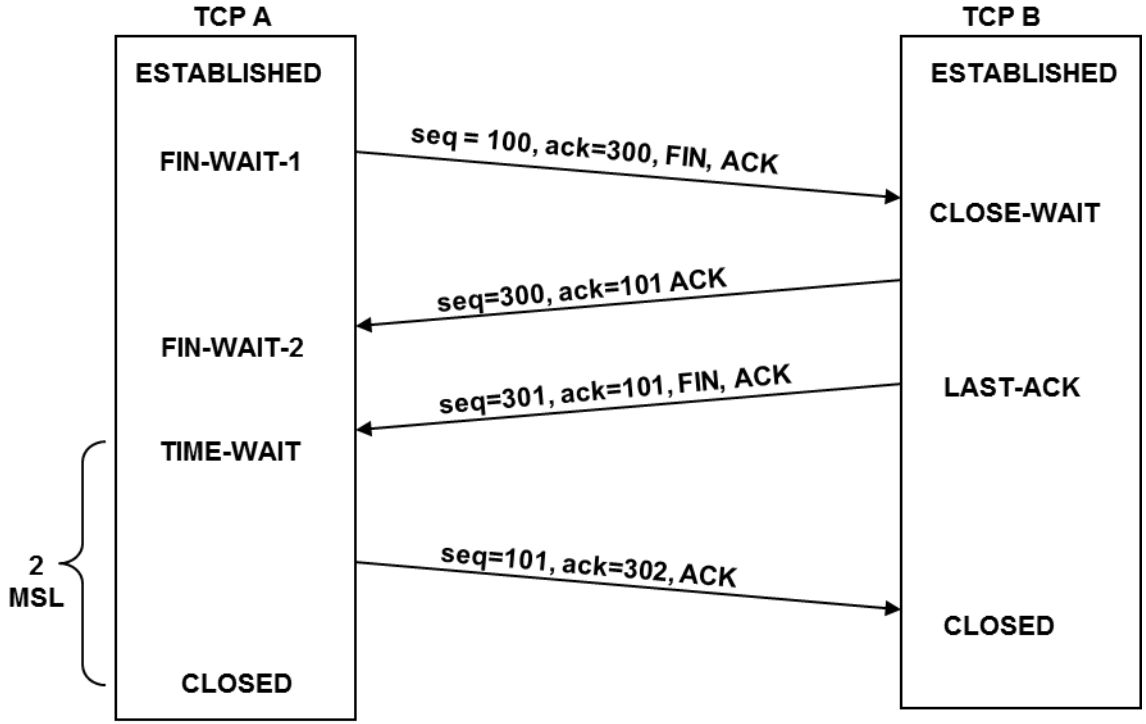


Podría hacerse un cierre unilateral

MSL: Maximum Segment Lifetime (normalmente 2 minutos)

Control de la conexión

- Ejemplo: cierre en cuatro pasos (con *timeout)*



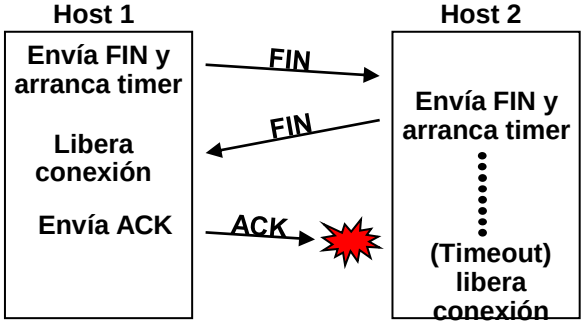
Si TCP B tiene que hacer algo más antes de cerrar

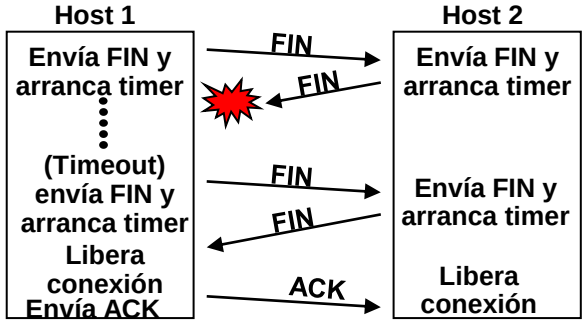
MSL: Maximum Segment Lifetime (normalmente 2 minutos)

Control de la conexión

- Ejemplos de cierres anormales:

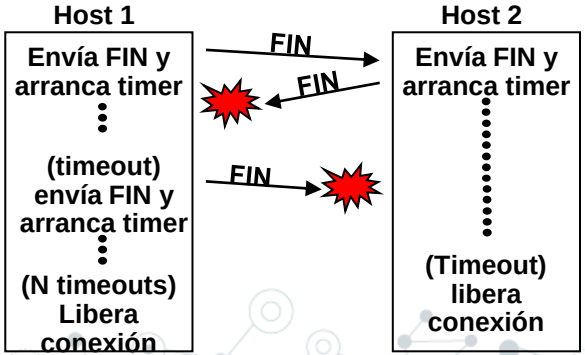
Pérdida de ACK final

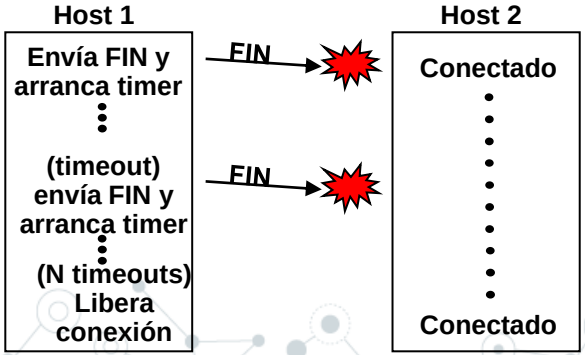




Pérdida de respuesta FIN y reenvío

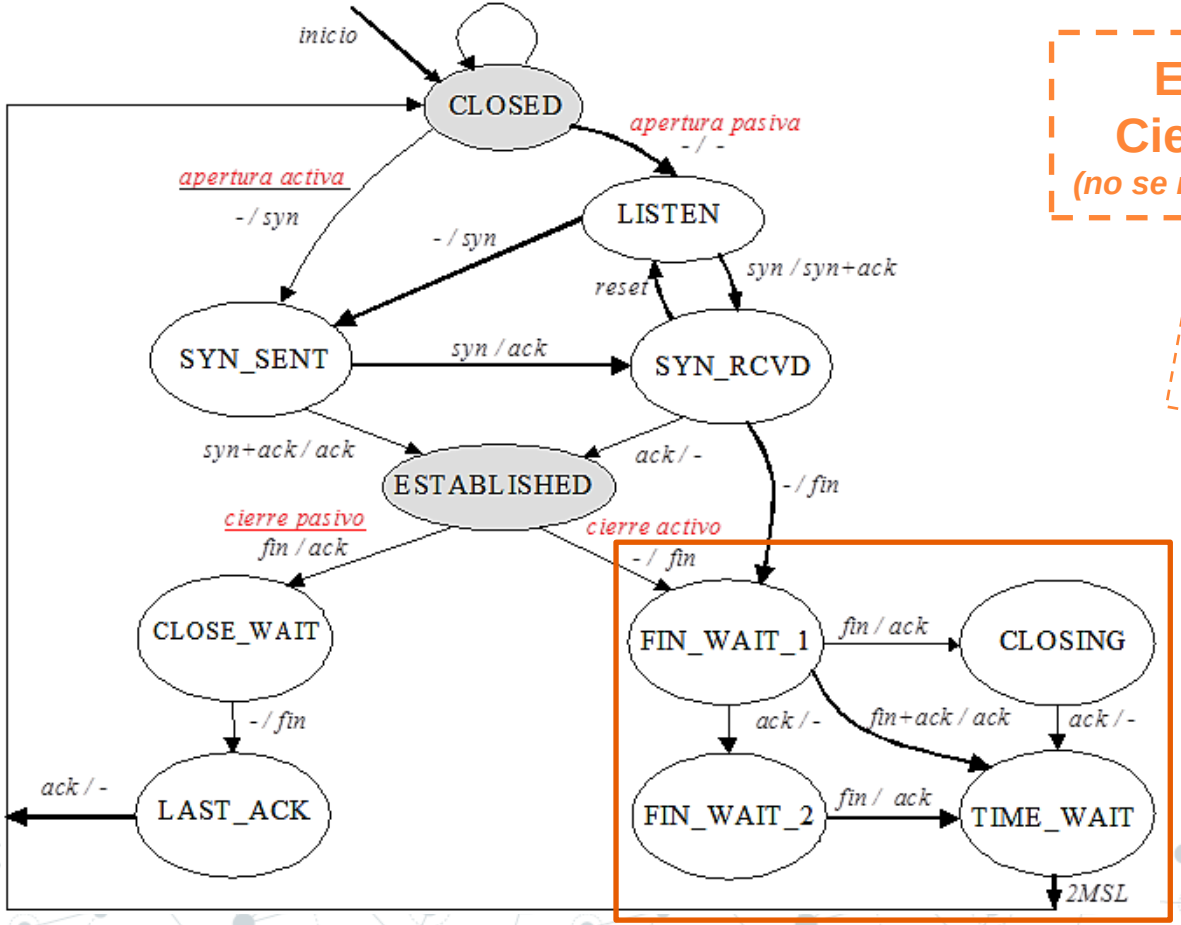
Pérdida de respuesta FIN y pérdida de reenvío





Pérdida de envío FIN

Autómata de estados finitos TCP



Establecimiento y Cierre de la conexión
(no se muestra intercambio de datos)

LEYENDA: a/b
Segmento a recibido
Segmento b transmitido

Distintas posibilidades de cierres de conexión

Intercambio de datos

TCP ⇔ APLICACIÓN

- El intercambio de datos **lo realizan** una **aplicación** en el **origen** y otra aplicación en el **destino**.

- **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 de que esa separación (si existe) se mantenga después de transmitir los datos.

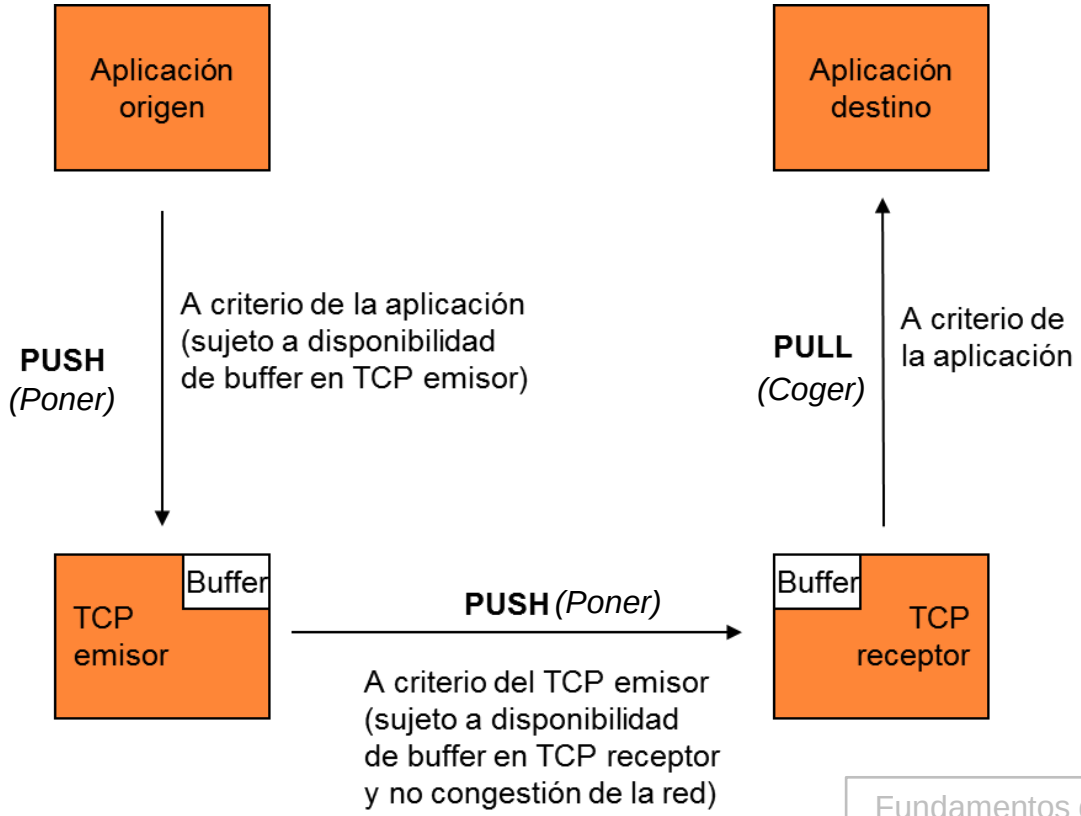
Intercambio de datos

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* (sobrecarga) debido a cabeceras y proceso de segmentos.

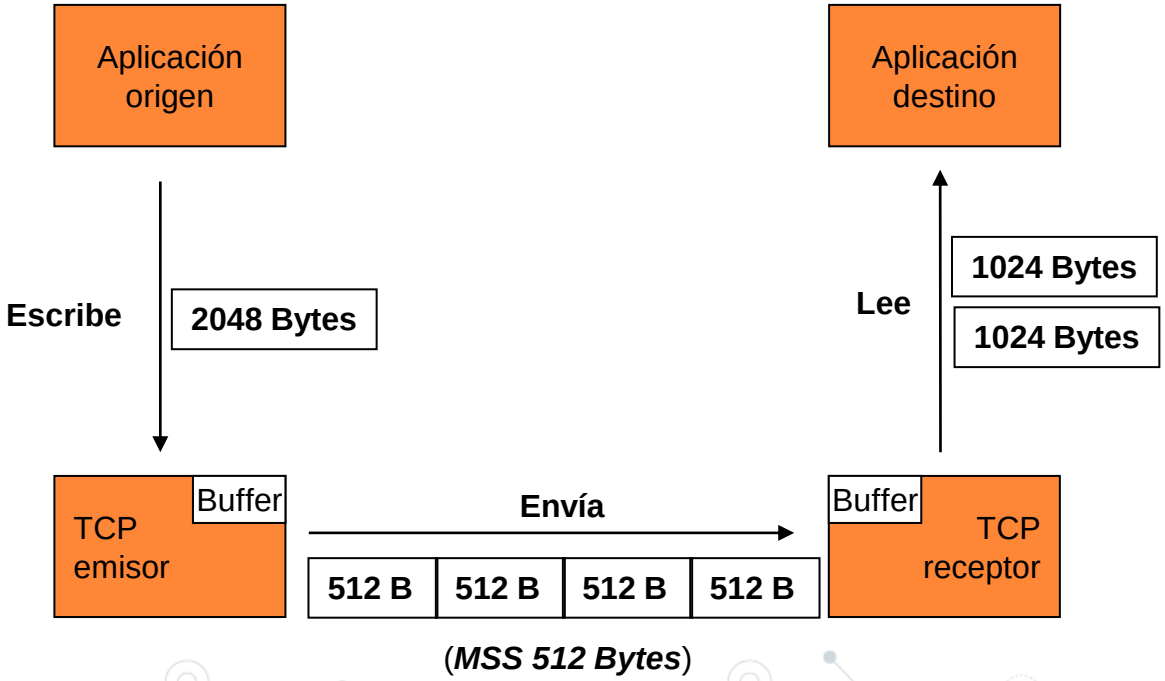
Intercambio de datos

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 (sin esperar a tener un segmento de datos completo). 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

TEMA 3. Capa de transporte en Internet

- 3.1. Introducción a los protocolos de Capa de Transporte
- 3.2. Protocolo de datagrama de usuario (UDP)
- **3.3. Protocolo de control de transmisión (TCP)**
  - Multiplexación/demultiplexación
  - Control de conexión
  - **Control de errores y de flujo**
  - Control de congestión
- 3.4. Extensiones TCP
- 3.5. Cuestiones y ejercicios

Control de errores

- Para el control de errores se sigue un **esquema ARQ** (*Automatic Repeat-reQuest*) con **confirmaciones positivas y acumulativas**.

- **Campos** involucrados:
  - Campo **secuencia**: *offset* (en bytes) dentro del mensaje.
  - Campo **acuse**: número de byte esperado en el receptor.
  - Bit **A** (ACK) del campo de **control**.
  - Campo **comprobación**: checksum de todo el segmento y uso de pseudo-cabecera.

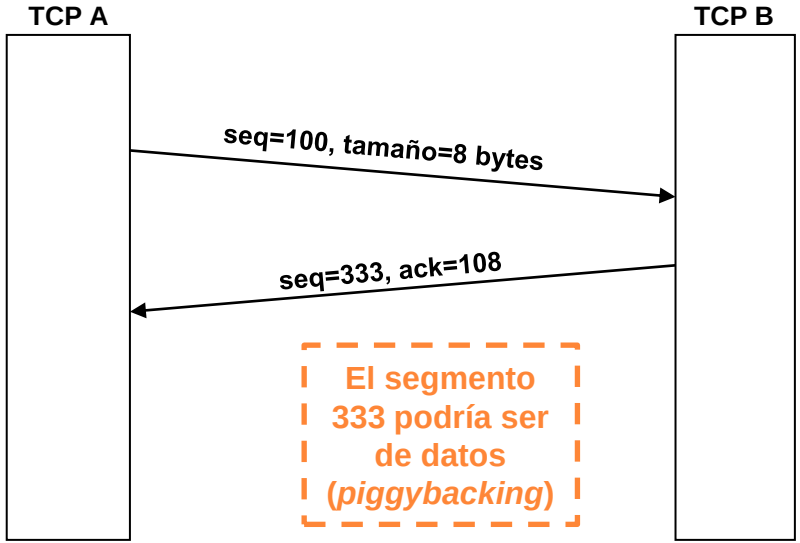
No hay confirmaciones negativas

Se pueden confirmar varios segmentos de una vez

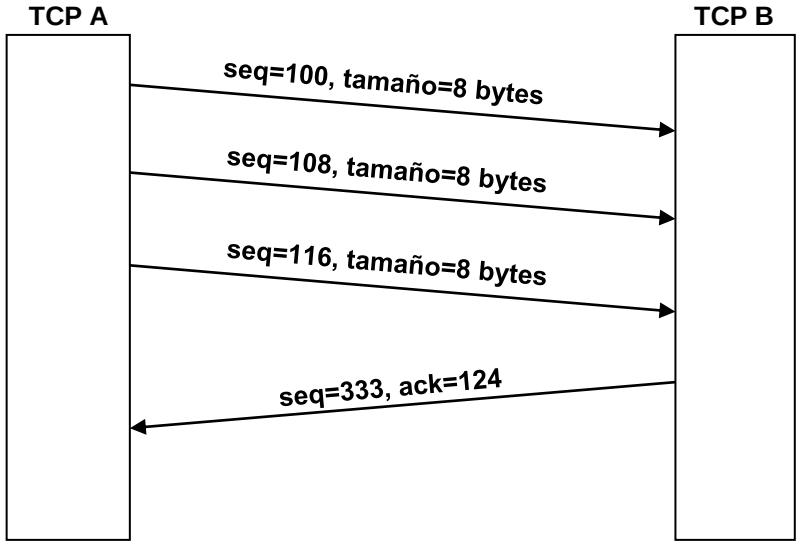
- **Confirmaciones** mediante ***piggybacking***:
  - Se envía la confirmación en un segmento con datos enviado en el otro sentido (los campos acuse y A se usan en un segmento de datos)

Control de errores

- **ARQ: Funcionamiento habitual**



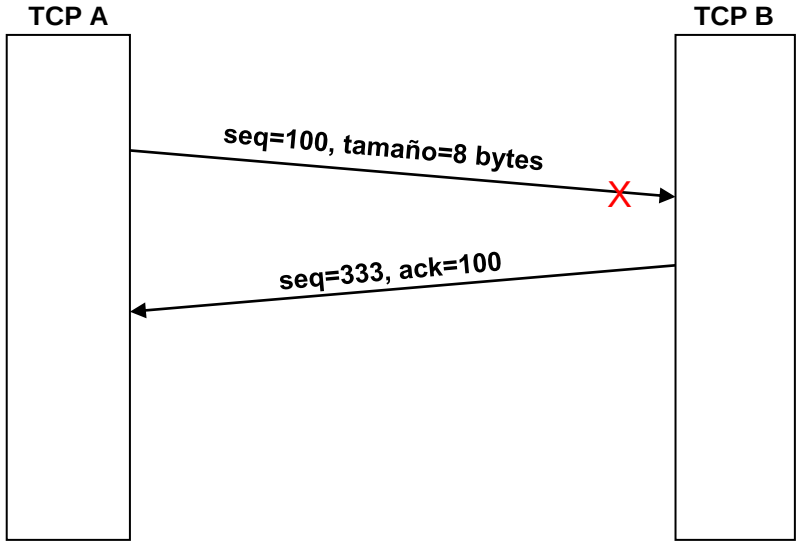
Confirmación simple



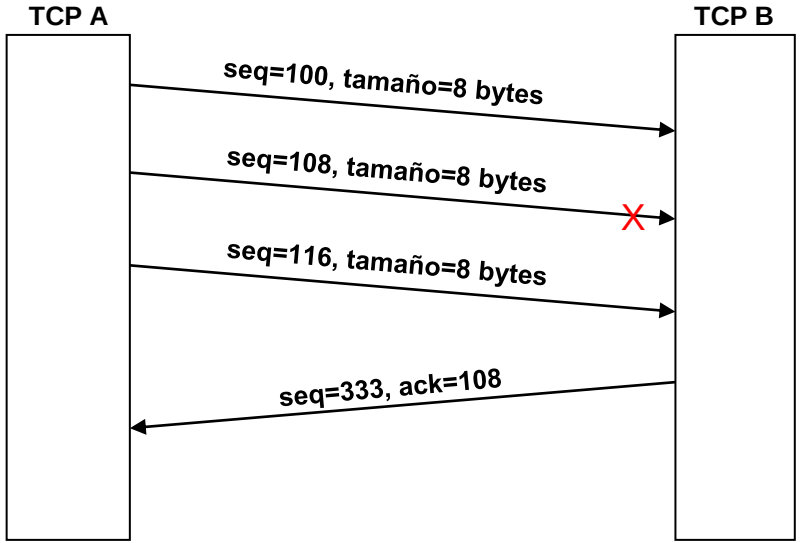
Confirmación acumulativa

Control de errores

- **ARQ: Error en segmento**



Solicitud de reenvío

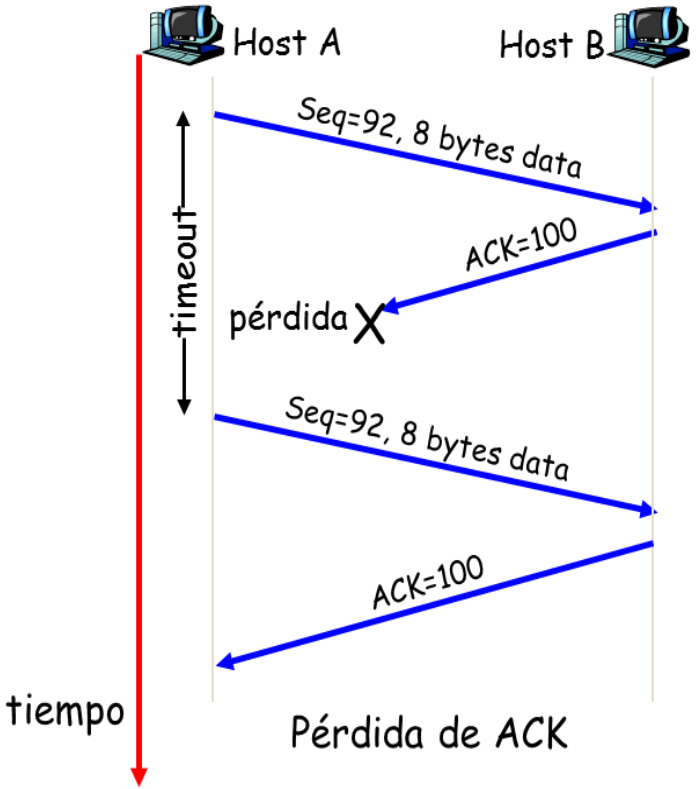


Solicitud de reenvío (desde el error)

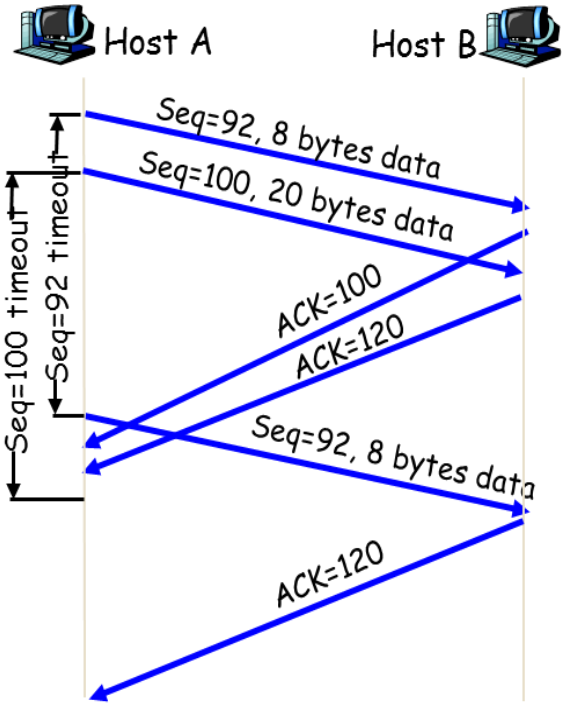
Control de errores

- **ARQ: Retransmisión**

(Gráfico J.F. Kurose)



Pérdida de ACK



timeout prematuro y
ACK acumulativo

Control de errores

- **ARQ: Protocolo de 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 núm. de secuen. mayor que el esperado, discontinuidad detectada.	Enviar un ACK duplicado, indicando el núm de secuen. 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

- **ARQ: ¿Cómo estimar los *timeouts*?**
  - Debe ser **mayor que** el **tiempo de ida y vuelta** (***RTT, Round Trip Time***), pero ¿cuánto?
  - Si es ***demasiado pequeño***: timeouts prematuros ➔ retransmisiones innecesarias
  - Si es ***demasiado grande***: reacción lenta a pérdida de segmentos ➔ baja eficacia
  - Para situaciones cambiantes … la mejor solución es **adaptarse dinámicamente**.

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

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

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

$$Timeout = RTTnuevo + 4 * Desviacion$$

Kurose & Ross

Control de errores

- **ARQ: ¿Cómo estimar los *timeouts*?**
  - Problema con ACKs repetidos: ambigüedad en la interpretación.
  - Solución: **Algoritmo de Karn**, *actualizar el RTT sólo para los no ambiguos,* pero si hay que repetir un segmento duplicar el *timeout*:

$$tout_{nuevo} = \gamma \, . \, tout_{viejo} \, , \; \gamma = 2$$

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

$$\textbf{RTTnuevo} = (1-\alpha) \cdot RTTviejo + \alpha \cdot RTTmedido \, , \quad \alpha \; \& \; \beta \in [0,1]$$

$$\textbf{Desviacion}_{\textbf{nueva}} = (1- \beta) \cdot Desviacion_{vieja} + \beta \cdot | \, RTTmedido - RTTnuevo \, |$$

$$\textbf{Timeout} = \textbf{RTTnuevo} + 4 * \textbf{Desviacion}$$

Kurose & Ross

Control de errores

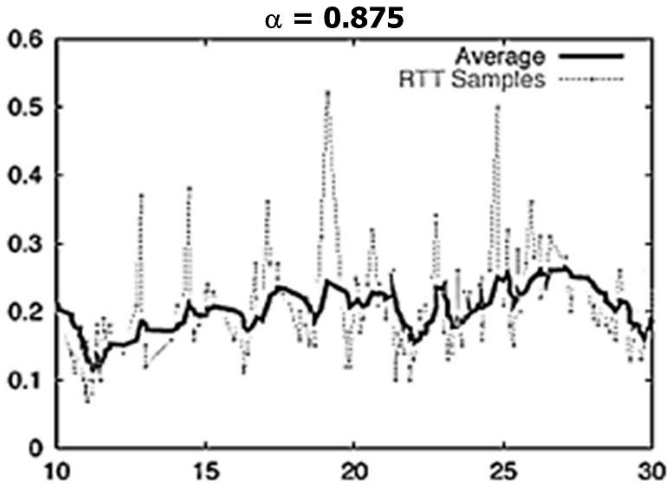
- **ARQ: ¿Cómo estimar los *timeouts*?**

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

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

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

$$Timeout = RTTnuevo + 4 * Desviacion$$

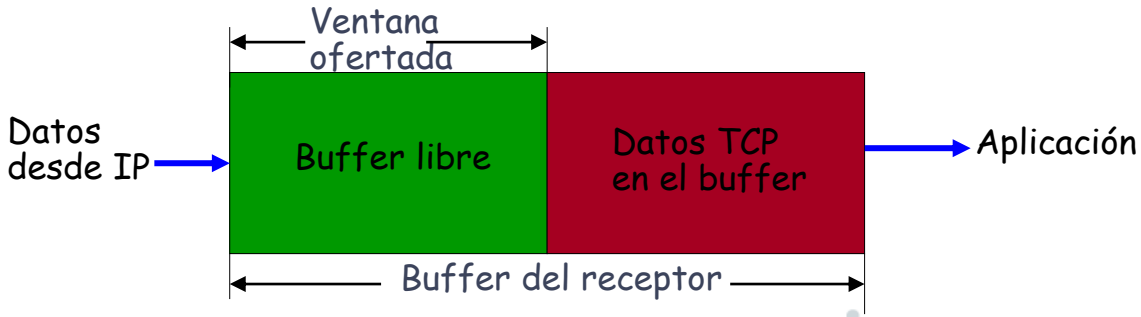


Ejemplo de RTT medidos y estimados entre Amherst, Massachusetts y St. Louis, Missouri.

Control de flujo

- Procedimiento para **evitar que el emisor sature al receptor** con el envío de demasiada información y/o demasiado rápido.
- Es un **esquema crediticio**: el receptor informa al emisor sobre los bytes autorizados a emitir sin esperar respuesta.
- Se utiliza el campo **ventana**:

ventana útil emisor = ventana ofertada receptor – bytes en tránsito

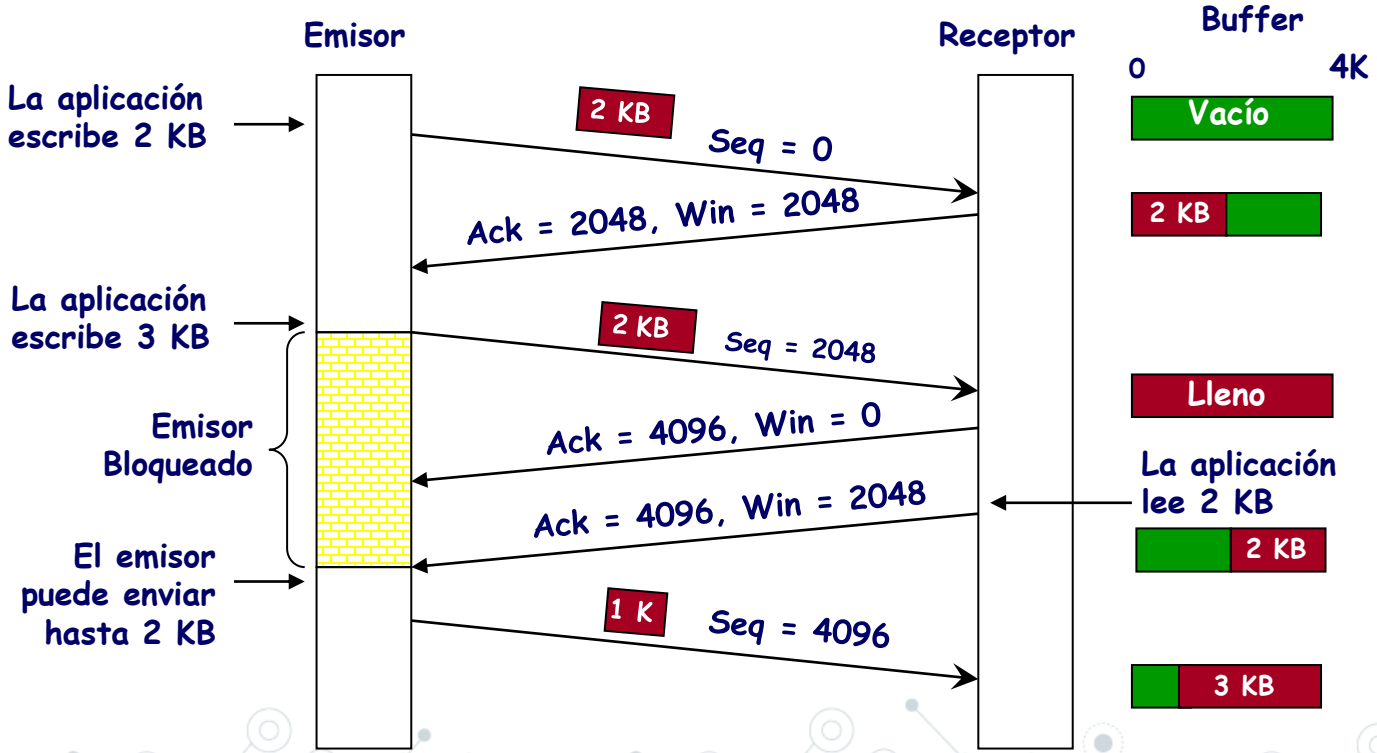


Se sigue un esquema de **Ventana Deslizante**

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 (WIN)**.
- Anunciando una **ventana cero** el **receptor puede bloquear al emisor**, y ejercer así **control de flujo**.
- La **ventana anunciada (**u **ofertada)** es un espacio que el **TCP receptor reserva** para esa comunicación en su ***buffer***.
- Tanto los números de secuencia como los **tamaños de ventana se indican en bytes**.

Control de flujo

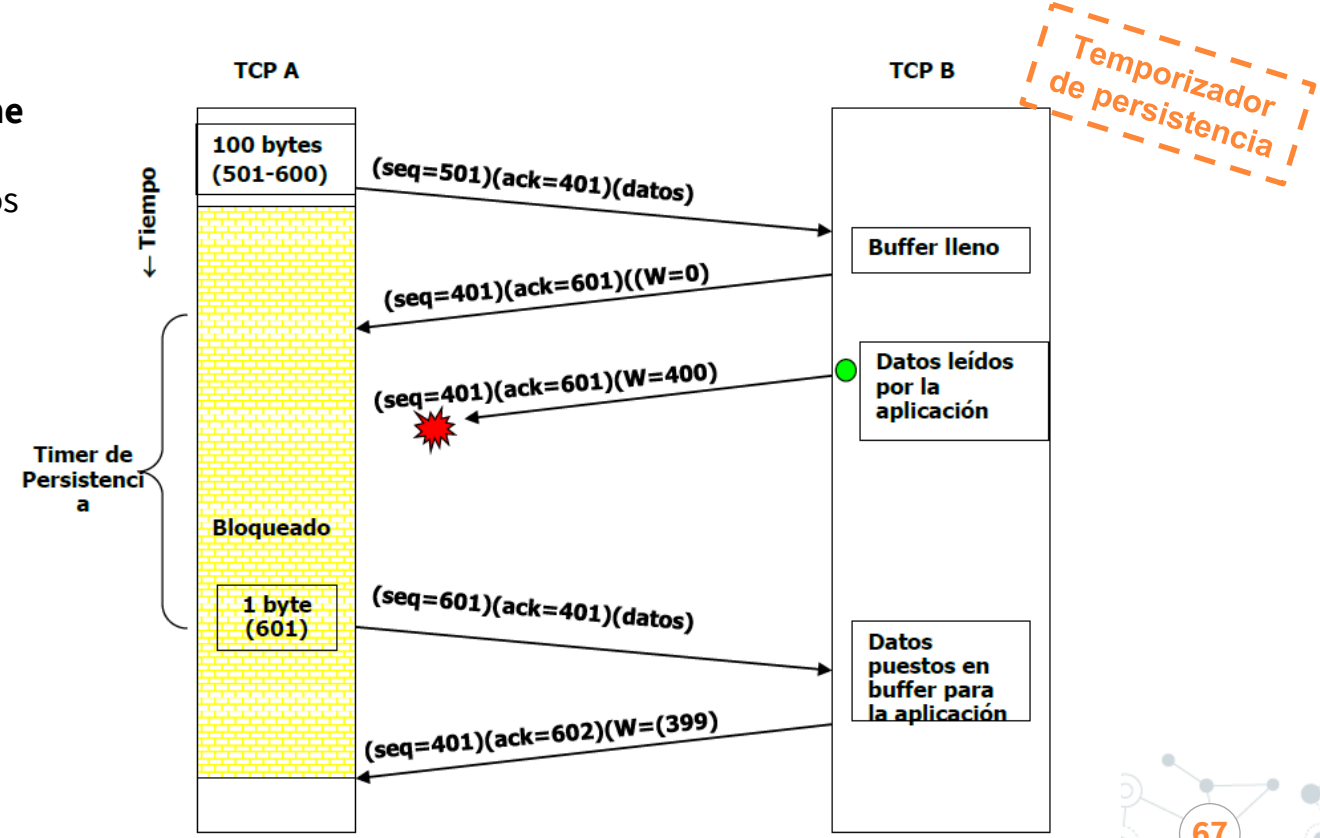


Control de flujo

- Si se perdiera el anuncio de la ventana disponible en el receptor, el emisor podría quedar bloqueado.

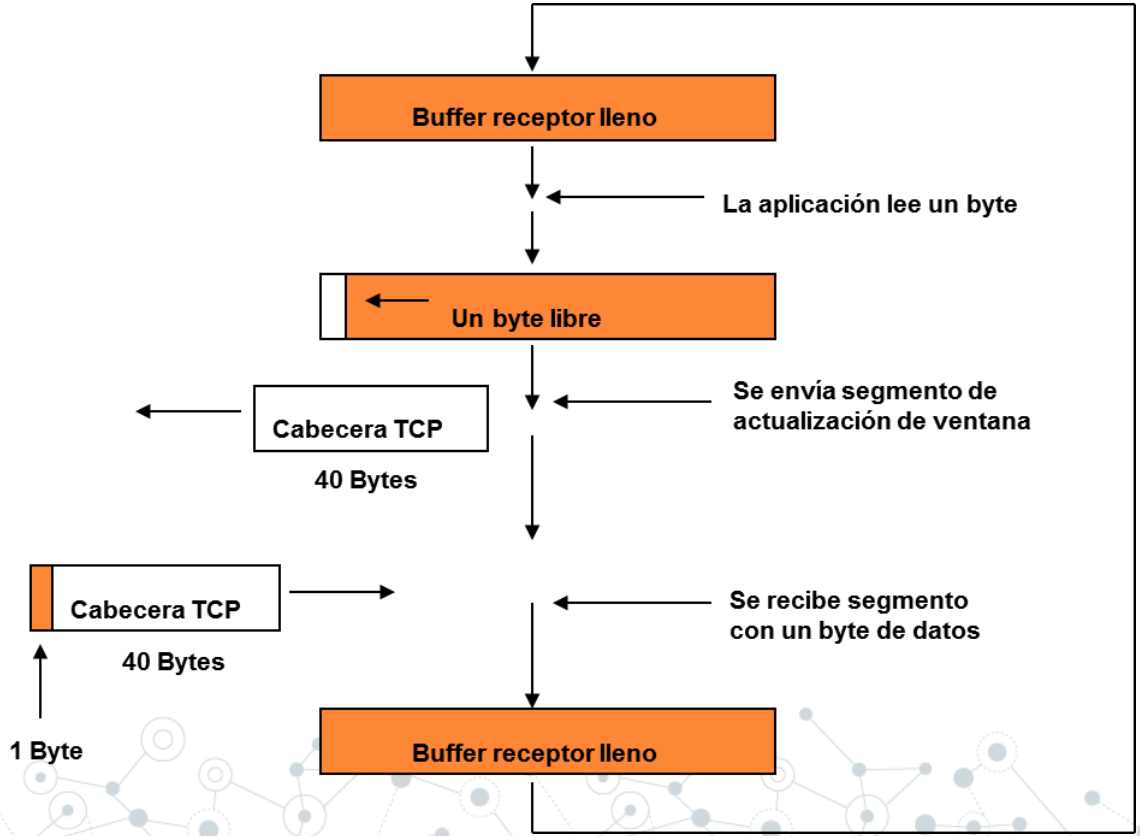
- Posible problema: **síndrome de la ventana tonta** (RFC 813) si se utilizan segmentos muy pequeños.

- Posible mejora: la **ventana optimista** (RFC 813) o solución de Clark.



Control de flujo

Síndrome de la ventana tonta (RFC 813)



Control de flujo

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.

TEMA 3. Capa de transporte en Internet

- 3.1. Introducción a los protocolos de Capa de Transporte
- 3.2. Protocolo de datagrama de usuario (UDP)
- **3.3. Protocolo de control de transmisión (TCP)**
  - Multiplexación/demultiplexación
  - Control de conexión
  - Control de errores y de flujo
  - **Control de congestión**
- 3.4. Extensiones TCP
- 3.5. Cuestiones y ejercicios

Control de congestión

- Adaptación a las características o rendimiento de la red (RFC 2001).
- Es un problema debido a la **insuficiencia de recursos** (la capacidad o velocidad de transmisión de las líneas y el buffer en routers y hosts no son infinitos).
- Es un **problema diferente al control del flujo**: el control de congestión es **para proteger a la red** debido a sus limitaciones.
- Los episodios de **congestión** se manifiestan en **retrasos en las ACKs y/o pérdidas de segmentos**, dependiendo del nivel de severidad del episodio.
- Solución extremo a extremo: en el **emisor limitar** de forma **adaptable el tráfico generado** para evitar pérdidas, pero siendo eficaz.
- La limitación se hace se hace mediante una aproximación conservadora: **limitando el tamaño de la ventana de emisión**.

Control de congestión

- Cuando hay **congestión** TCP debe de **reducir el flujo de datos**.
- 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 **se 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 **arranque lento** (***slow-start***) y fue diseñado por Van Jacobson en los años 80.

Control de congestión

SLOW START (PRUEBA Y ERROR)

- El emisor utiliza **dos ventanas** y un **umbral**.

Bytes_permitidos_enviar =
min{VentanaCongestion, VentanaDelReceptor}

VentanaDelReceptor: utilizada para el control de flujo (de tamaño variable) según el campo "ventana" recibido

VentanaCongestion:
Inicialmente VentanaCongestion = 1 · MSS

Inicio lento

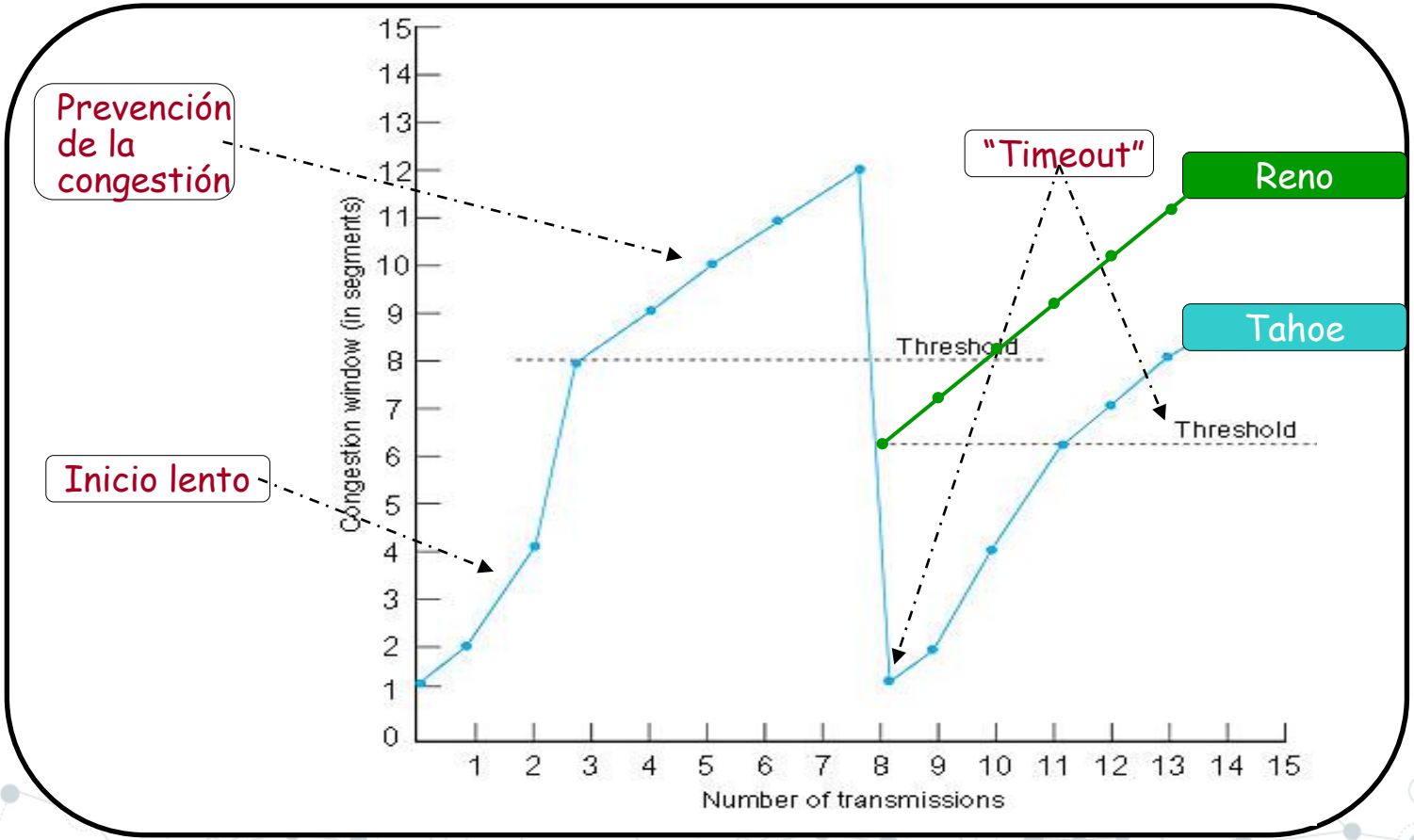
Si VentanaCongestion < umbral, por cada ACK recibido
VentanaCongestion += MSS **(crecimiento exponencial)**

Prevención de la congestión

Si VentanaCongestion > umbral, cada vez que se recibe **todos los ACKs pendientes**
VentanaCongestion += MSS (**crecimiento lineal**)

Si hay timeout entonces
umbral=VentanaCongestion/2 y VentanaCongestion = MSS

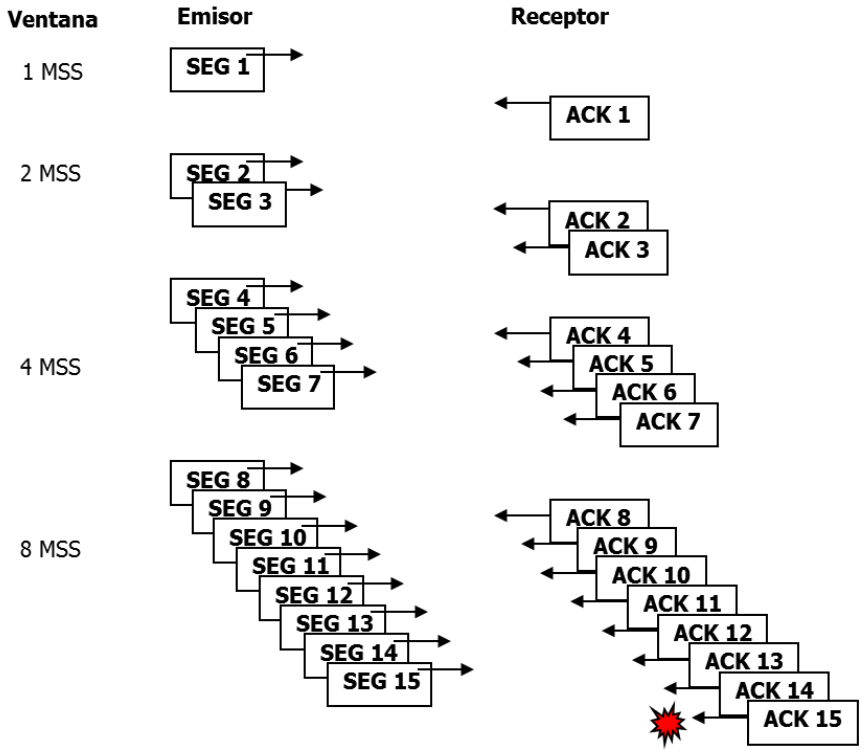
Control de congestión



Control de congestión

SLOW START (PRUEBA Y ERROR) – 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 en cada ACK).
- Si la ventana de congestión supera a la de control de flujo se aplica la restricción de ésta última con lo cual aquella deja de crecer .



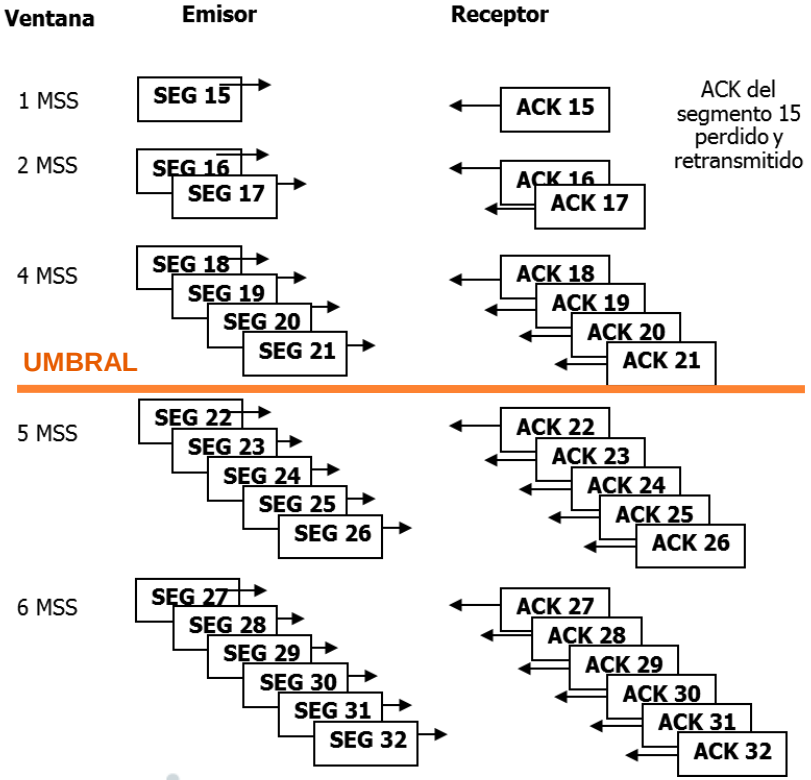
Control de congestión

SLOW START (PRUEBA Y ERROR) – SEGUNDA FASE

[Congestion Avoidance]

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 que se recibe un ACK.



TEMA 3. Capa de transporte en Internet

- 3.1. Introducción a los protocolos de Capa de Transporte
- 3.2. Protocolo de datagrama de usuario (UDP)
- 3.3. Protocolo de control de transmisión (TCP)
  - Multiplexación/demultiplexación
  - Control de conexión
  - Control de errores y de flujo
  - Control de congestión
- **3.4. Extensiones TCP**
- 3.5. Cuestiones y ejercicios

Variantes de 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**

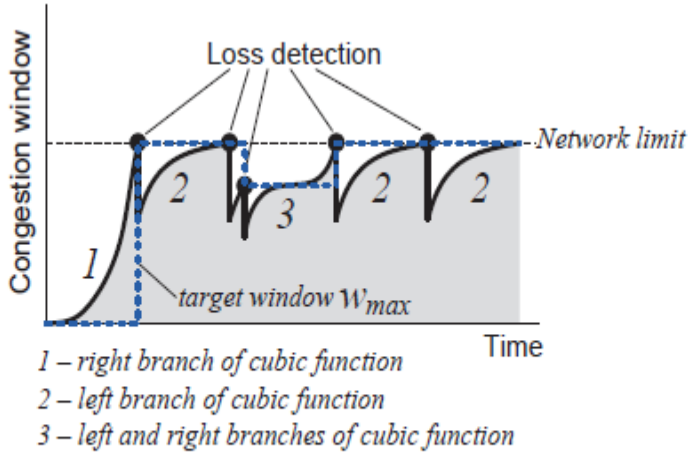
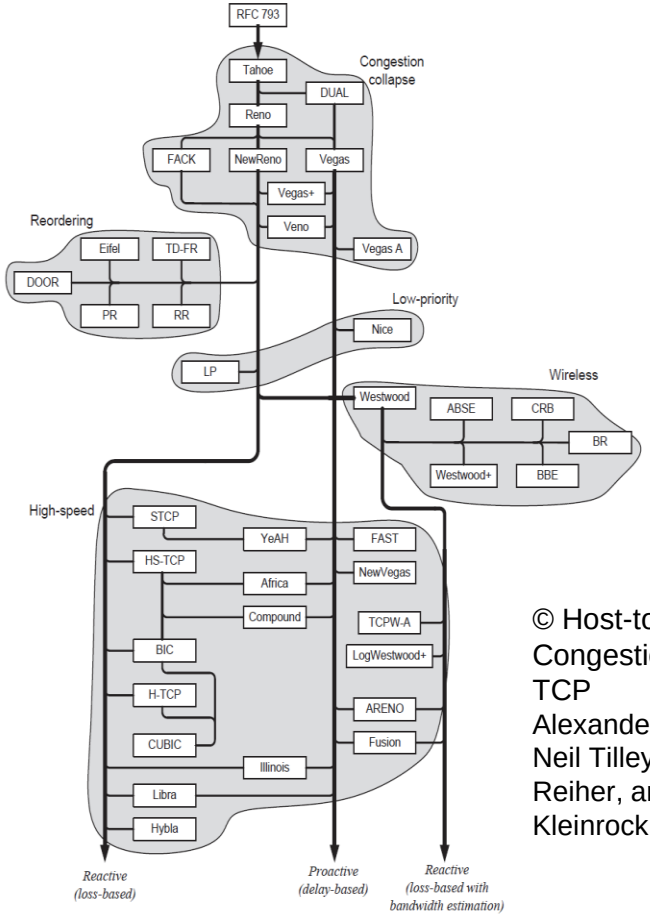


Fig. 50. Congestion window dynamics in CUBIC



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

Variantes de 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. Ventana selectiva.

¿Preguntas?

O comentarios, sugerencias, inquietudes