# Tema-3FR.pdf

mhm01

Fundamentos de Redes

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

TEMA 3: CAPA DE TRANSPORTE

## 1. Introducción a los Protocolos de Capa de Transportes

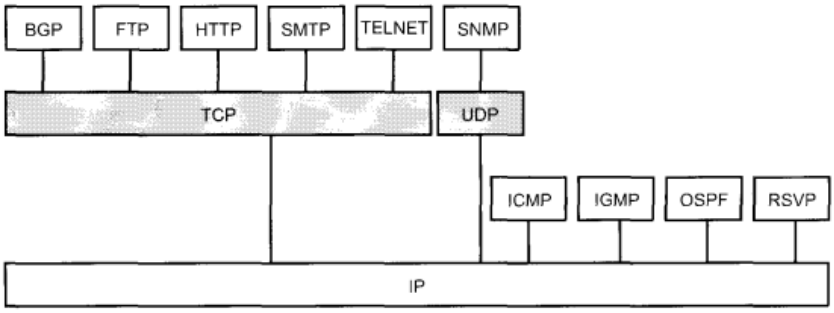
**---------- Capa de Transportes**

Las redes de datos en Internet nos dan soporte para establecer una comunicación continua y confiable entre los equipos. Es un único dispositivo, se pueden utilizar varios servicios (correo electrónico, acceso Web, mensajería instantáneas, 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.

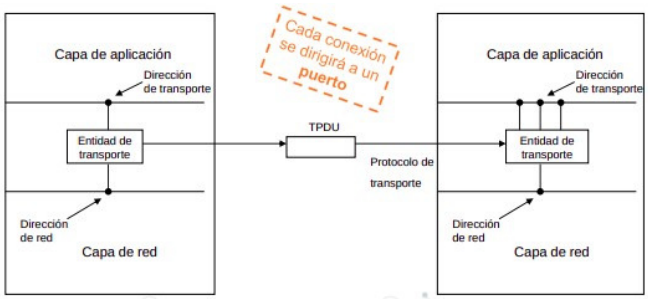
Las entidades de transporte es el hardware y/o software que se encargan de realizar este trabajo. La comunicación entre entidades de transporte es extremo a extremo (end to end), es decir, se produce entre el emisor/receptor finales, no teniendo en cuenta a ningún otro dispositivo intermedio de las subredes. El nivel de transporte mejora la calidad del servicio ofrecida por el nivel de red mediante: la multiplexación/demultiplexación y la introducción de redundancia en la información.

Hay dos protocolos: TCP y UDP. La capa de transporte es el enlace entre la capa de aplicación y la capa responsable de la transmisión en la red: red (en el modelo OSI) y Internet (en el 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.

Los protocolos de las capas de aplicación y transporte.

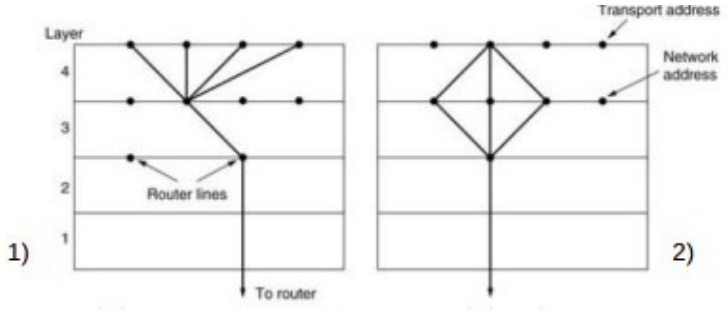



Permite realizar la multiplexión de comunicaciones (de aplicaciones).



**- Multiplexión/Demultiplexión:**

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 se quiere aumentar el caudal o reducirse el retardo en una conexión 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:

- 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.

- **Segmentación de datos (de aplicación) 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 encabexados 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 mas fáciles de administrar y controlar (errores, flujo, etc).

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

- **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 a este identificador número de puerto.

# KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

### ---------- 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.

#### ----- 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
- 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.

### ---------- 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

## 2. Protocolo de Datagrama de Usuario (UDP)

### ---------- Introducción

El protocolo UDP (User Datagram Protocol) se define en la RFC 768.

Proporciona un servicio de entrega de datagrama:

- **no orientado a conexión:** sin conexión previa (no hand-shaking). No hay retardo de establecimiento de la conexión. Cada TPDU (datagrama) es independiente.
- **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).
- multiplexión/demultiplexión (transportar el TPDU al proceso/aplicación correcto).

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

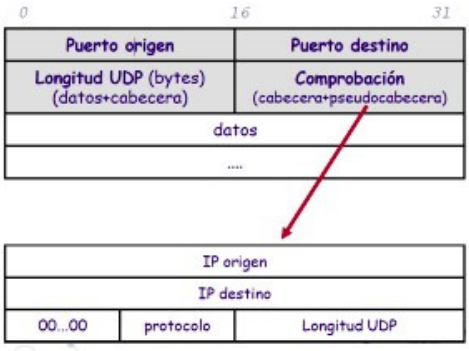
UDP suele utilizarse para:

- 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.

## ---------- 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.



- **Comprobación**:
  - 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).

## 3. Protocolo de Control de Transmisión (TCP)

---------- **Características del Transmission Control Protocolo (TCP)**

Es un servicio orientado a conexión que exigen un acuerdo entre emisor y receptor (hand shaking). Hace una entrega ordenada de las secuencias de bytes generadas por las aplicaciones (stream oriented). La transmisión es full duplex (se pueden enviar datos en ambos sentidos al mismo tiempo). El mecanismo de detección de errores ARQ (automatic Repeat reQuest) tiene confirmaciones positivas (ACKs) acumulativas, timeouts adaptables y incorporación de confirmaciones con los dados (piggybacking). Es un servicio fiable con control de congestión y control de flujo con ventanas deslizantes con tamaño máximo adaptable. Es un servicio punto a punto y no puede usarse para multicast.

----- **Servicios que ofrece TCP**

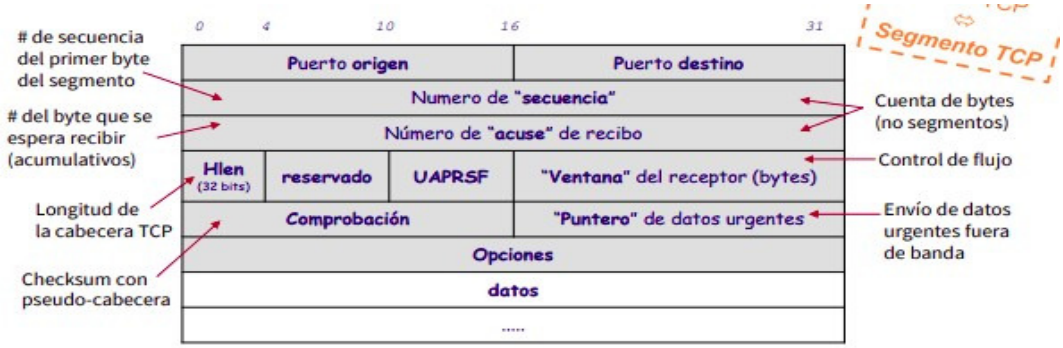
- **Establecimiento y Cierre de la Conexión:** al ser un protocolo orientado a conexión, dispone de mecanismos para establecer la conexión (antes de la transmisión de datos) y para cerrarla (al final de la transmisión).

- **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.

**----- Cabecera TCP**




Cada segmento TCP se encapsula en un datagrama IP.

### 3.3.1. Multiplexación/Demultiplexación

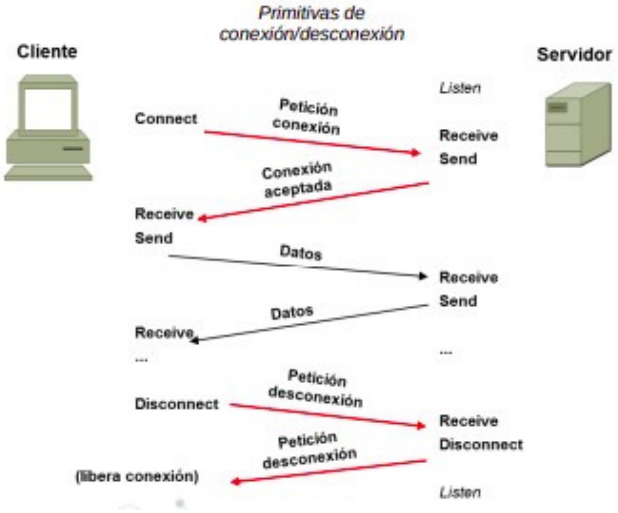
Consiste en trasportar los segmentos de aplicación correcta. Se realiza (al igual que en UDP) utilizando puertos asociados a cada aplicación. Los puertos TCP son diferentes que los UDP. Una conexión TCP se identifica por: IP y puerto origen y IP y puerto destino. Existen algunos puertos preasignados:

| 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 |

### 3.3.2. Control de Conexión

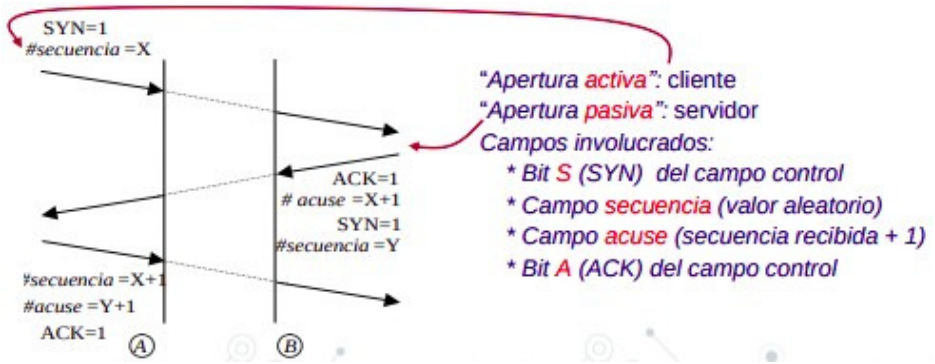
Es un mecanismo de sincronización entre emisor y receptor. Para garantizar la comunicación ordenada y sin errores. El TCP es orientado a conexión. El intercambio de información tiene tres fases:

- Establecimiento de la conexión (sincronizar # de secuencia y reservar recursos).
- Intercambio de datos (full-duplex).
- Cierre de la conexión (liberar recursos).




## ---------- 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.

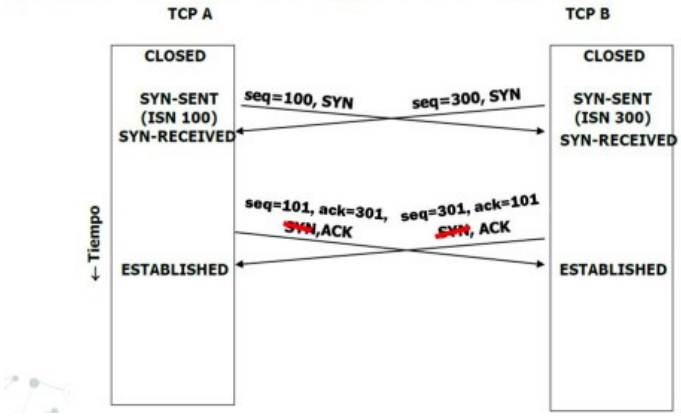



## ---------- Números de Secuencia

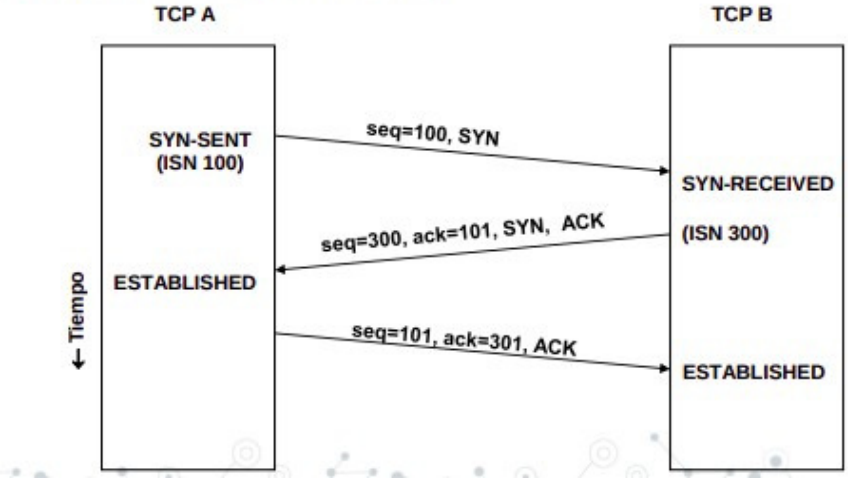
El número de secuencia es un campo de 32 bits que cuenta bytes en módulo $2^{32}$ (el contador se da la vuelta cuando llega al valor máximo). El número de secuencia no empieza normalmente en 0, sino en un valor denominado ISN (Initial Sequence Number) elegido "teóricamente" al azar; para evitar confusiones con transmisiones anterior. 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 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. El 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).

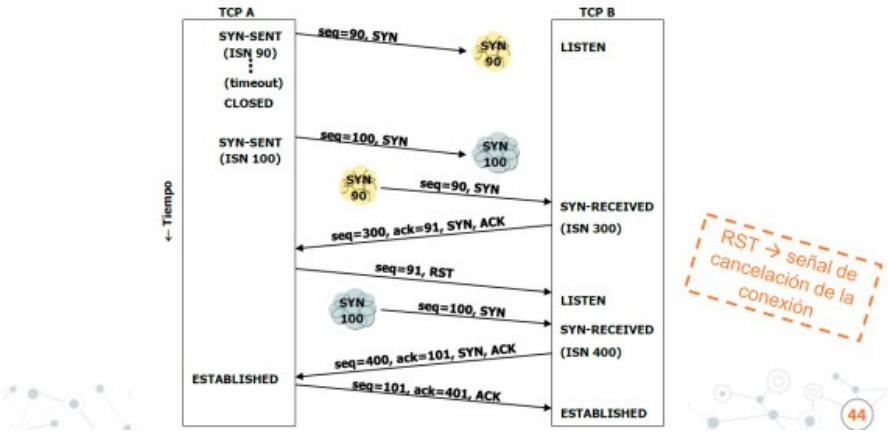
Ejemplo: ***three-way handshaking (Conexión simultánea)***




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

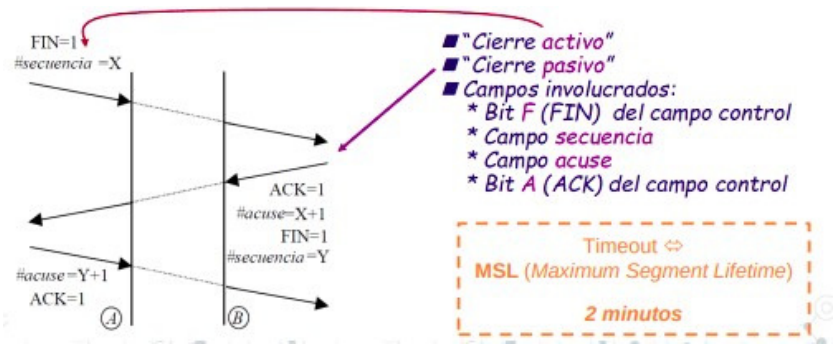


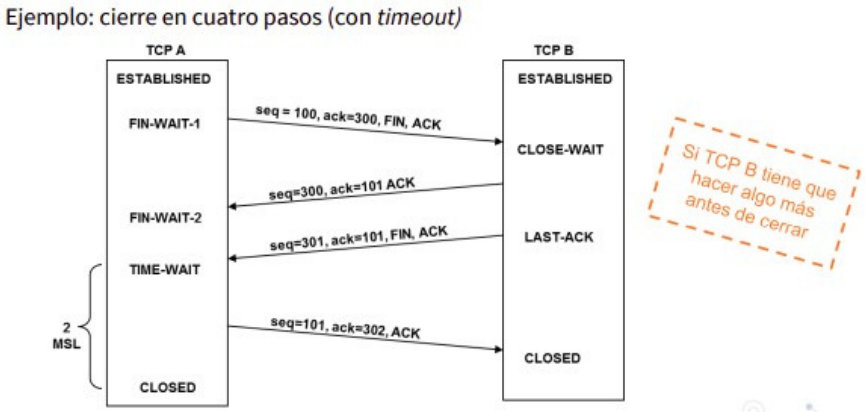

Ejemplo: ***three-way handshaking (SYN retrasados y duplicados)***

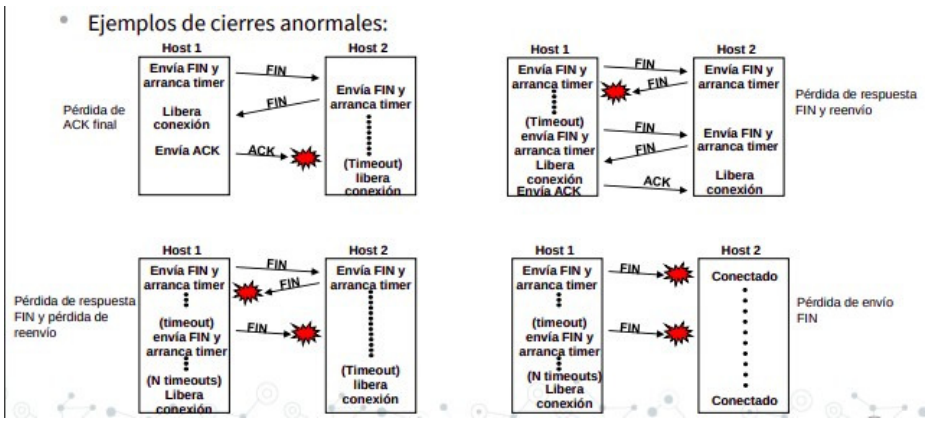



## ---------- Cierre

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




**---------- 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.

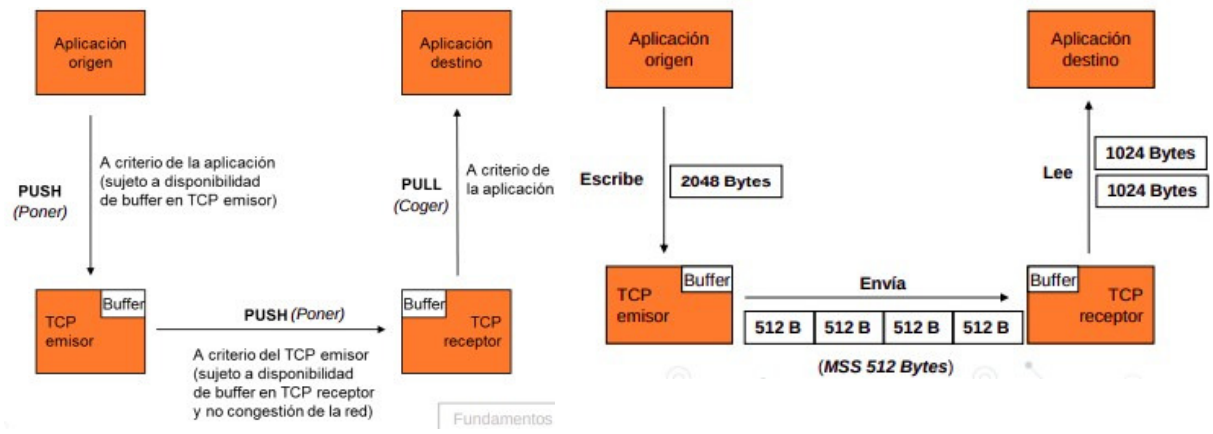
**- TCP → Aplicación:** la aplicación lee del buffer de recepción de TCP cuando quiere y cuanto quiere. Se hace una excepción con los datos urgentes.

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.

**----- TCP ↔ TCP**

El TCP emisor manda los datos cuando quiere. A excepción de los 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.

**----- TCP ↔ Aplicación y TCP ↔ TCP**




**----- 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.

### 3.3.3. Control de Errores y de Flujo

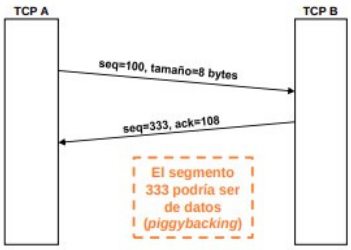
## ---------- Control de Errores

Para el control de errores se sigue un esquema ARQ (Automatic Repeat-reQuest) con confirmaciones positivas y acumulativas. Los campos involucrados son:

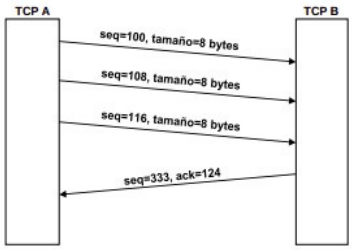
- El Campo Secuencia: offset (en bytes) dentro del mensaje.
- El Campo Acuse: número de bytes esperando en el receptor.
- El bit A (ACK) de Campo de Control.
- El Campo de Comprobación: checksum de todo el segmento y el uso de pseudo-cabecera.

Las confirmaciones se hacen 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.

### ----- ARQ: Funcionamiento habitual

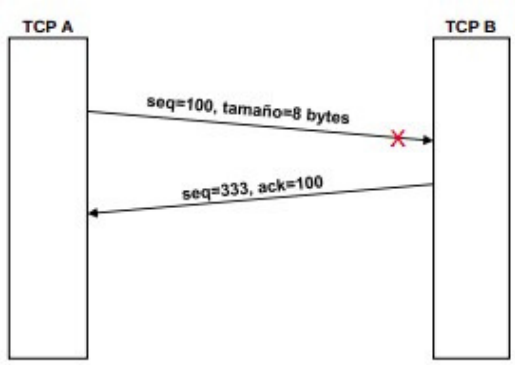


Confirmación simple

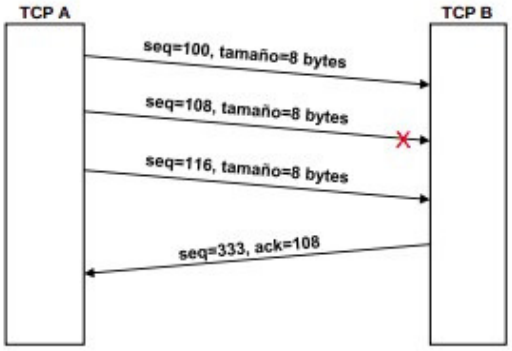


Confirmación acumulativa

### ----- ARQ: Error en Segmento

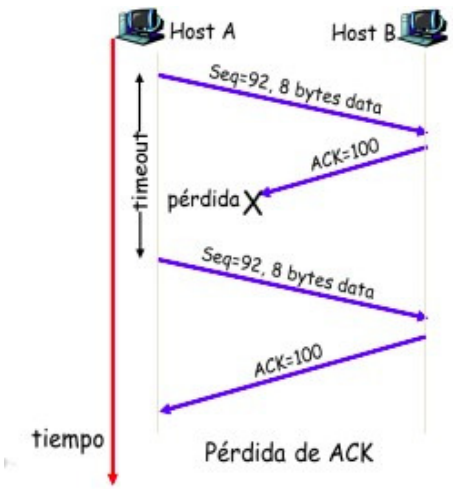


Solicitud de reenvío

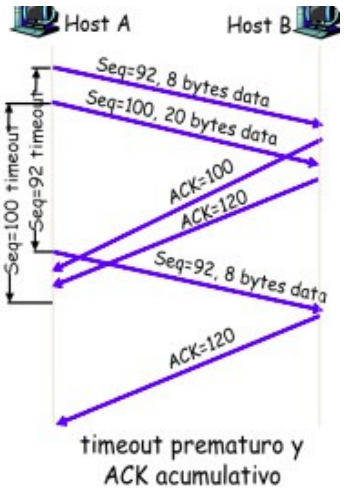


Solicitud de reenvío (desde el error)

### ----- ARQ: Retransmisión



Pérdida de ACK



timeout prematuro y ACK acumulativo

**----- ARQ: Protocolo de Generación de ACKS**

- Si hay una llegada ordenada de segmento, sin discontinuidad y con todo lo anterior ya confirmado: el TCP Receptor retrasará el ACK. Esperará al recibir al siguiente segmento hasta 500 ms. Si no llega, enviará el ACK.

- Si hay una llegada ordenada de segmento, sin discontinuidad y hay pendiente un ACK retrasado: el TCP Receptor enviará inmediatamente un único ACK acumulativo.

- Si hay una llegada es desordenada de segmento con un número de secuencia mayor que el esperado, con discontinuidad detectada: el TCP Receptorr enviará un ACK duplicado, indicando el número de secuencia del siguiente byte esperado.

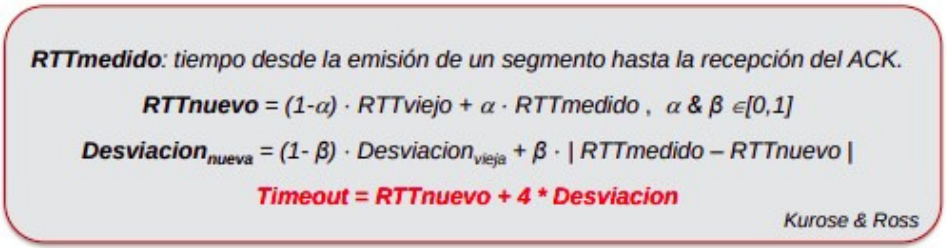
- Si hay una llegada de un segmento que completa una discontinuidad parcial o totalmente: el ACK Receptor confirma el ACK inmediatamente si el segmento comienza en el extremo inferior de la discontinuidad.

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

Debe ser mayor que el tiempo de ida y vuelta (RTT, Round Trip Time).

- Si es **demasiado pequeño:** los timeouts serán prematuros por lo que las retransmisiones serán innecesarias.
- Si es **demasiado grande:** habrá una reacción lenta a la pérdida de segmentos que provocará una baja eficiencia.

Para situaciones cambiantes, la mejor situación es adaptarse dinámicamente.




Si hay algún problema con los ACKs repetidos, tendremos una ambigüedad en la interpretación. Para solucionar esto usaremos el Algoritmo de Karn que actualiza el RTT sólo para los no ambiguos, pero si hay que repetir un segmento, hay que duplicar el timeout:

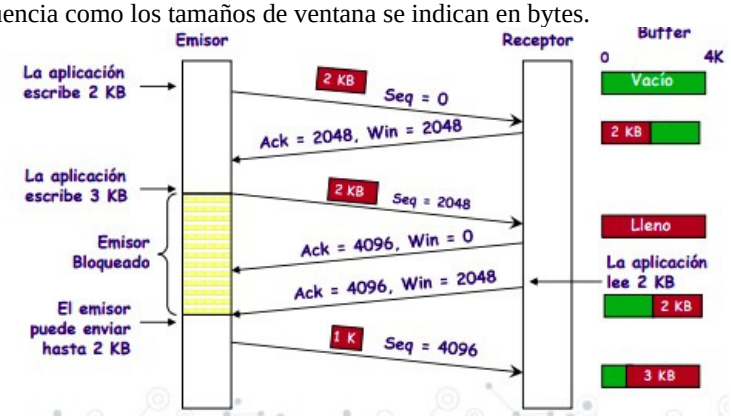
$$tout_{nuevo} = tout_{viejo} * y, \text{ siendo } y = 2$$

**---------- Control de Flujo**

Es el procedimiento para evitar que el emisor sature al receptor con el envío de demasiada información y/o demasiado rápido. Es un esquema crediticio, es decir, el receptor informa al emisor sobre los bytes autorizados a emitir sin esperar respuesta. Se utiliza el Campo Ventana:

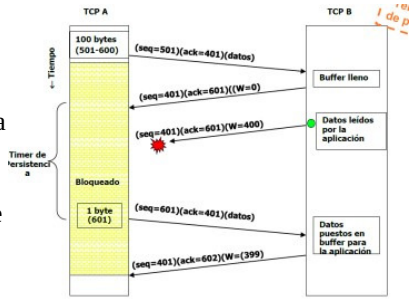
ventana útil emisor = ventana ofertada receptor – bytes en tránsito


Se sigue un esquema de ventana deslizante.
El TCP receptor informa en cada segmento al emisor del espacio que 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 número de secuencia como los tamaños de ventana se indican en bytes.




Si se perdiera el anuncio de la venta disponible en el recepto, el emisor podría quedar bloqueado.

Hay un posible problema que es el denominado Síndrome de la Ventana Tonta que ocurre si se utilizan segmentos muy pequeños. Para este problema, hay una posible solución, la ventaba Optimista o solución de Clark en la que el TCP receptor solo debe notificar una nueva ventana cuando tenga una cantidad razonable de espacio libre. Que sea razonable significa que o es un segmento de tamaño máximo (MSS) o la mitad del espacio disponible en el buffer.




### 3.3.4. Control de Congestión

Es la adaptación a las características o rendimiento de la red. 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. Hace uso de la solución de extremo a extremo: en el emisor hay que limitar de forma

adaptable el tráfico generado para evitar, pero siendo eficaz. La limitación se hace mediante una aproximación conservadora limitando el tamaño de la ventana de emisión.

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 TCP se denomina Arranque Lento (slow-start).

### ---------- Arranque Lento

El emisor utiliza dos ventanas y un umbral. Inicialmente la ventana de congestión tiene el tamaño de un MSS (Maximum Segment Size). Por cada segmento enviado con éxito la venta 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.

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.

## 4. Extensiones TCP

### ---------- Variantes de TCP

El TCP se define con múltiples "sabores". Los diferentes sabores no afectan a la interoperatividad entre los extremos. Desde cualquier versión de Linux con un kernel mayor que la 2.6.19 se usa por defecto TCP CuBIC.

**- Ventana escalada:** opción TCP en segmentos SYN: hasta $2^{14} * 2^{16}$ bytes = $2^{30}$ bytes = 1GB autorizados.

**- PAWS (Protect Against Wrapped Sequence numbers)**: Sello de tiempo y rechazo de segmentos duplicados.

**- SACK:** Confirmaciones selectivas. Ventana selectiva.