

TEMA 4. La capa de transporte

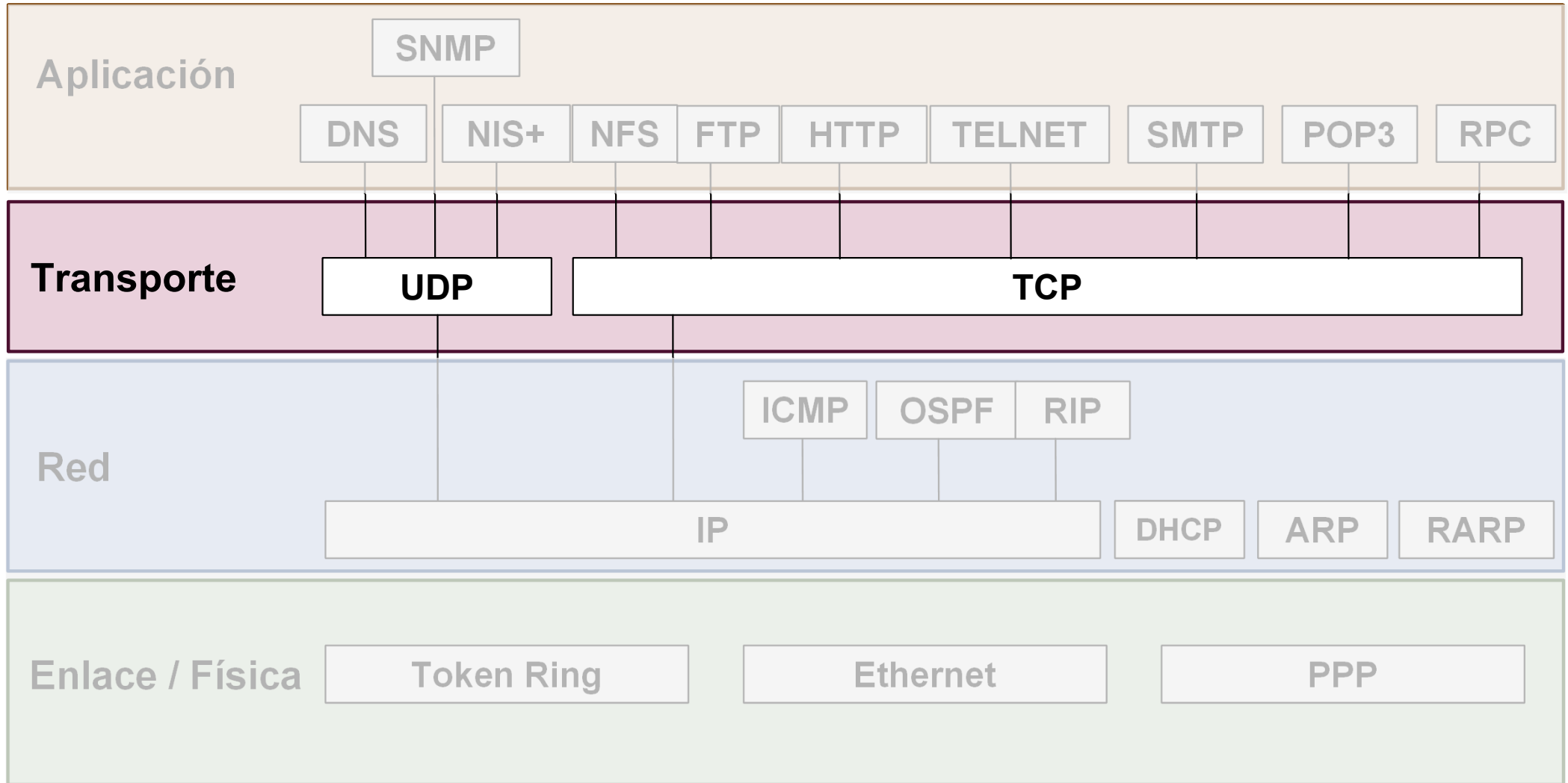
Protocolo TCP/UDP



Profesora: Guadalupe Miñana Ropero

Transparencias basadas en las de la asignatura de Redes de los grados de Informática de FDI

Arquitecturas TCP/IP



Protocolos de transporte en Internet: TCP y UDP

- El protocolo UDP (User Datagram Protocol)
 - Protocolo de transporte "sin conexión"
 - **No garantiza** un servicio extremo a extremo fiable
 - No controla la pérdida de paquetes, los errores o la duplicidad
 - Utilizado en aplicaciones en las que la rapidez en la entrega es más importante que la fiabilidad
 - DNS, SNMP, RIP, RTP, etc.

Protocolos de transporte en Internet: TCP y UDP

- **El protocolo TCP (Transmission Control Protocol)**
 - Protocolo de transporte "orientado a conexión"
 - Garantiza un servicio extremo a extremo fiable
 - Detecta segmentos de datos perdidos o erróneos y los retransmite
 - Detecta segmentos duplicados y los descarta
 - Ordena los segmentos en el destino y los entrega de forma ordenada a la capa de aplicación
 - Utilizado en aplicaciones en las que la fiabilidad en la entrega es más importante que la rapidez
 - Ejemplos: FTP, HTTP, Telnet, SMTP, etc.

Protocolos de transporte en Internet: TCP y UDP

Característica	TCP	UDP
Tipo de conexión	Orientado a conexión: <ul style="list-style-type: none">• Entrega ordenada de paquetes• Retransmisión de paquetes perdidos o erróneos• Detección de duplicados	Sin conexión: <ul style="list-style-type: none">• No garantiza la entrega ordenada• No retransmite paquetes perdidos o erróneos• No detecta duplicados
Unidad de transferencia	Segmento (20 bytes mínimo de cabecera)	Datagrama (8 bytes mínimo de cabecera)
Fases de la comunicación	<ol style="list-style-type: none">1. Establecimiento de conexión2. Transferencia de datos3. Cierre de conexión	Transferencia de datos (bloques individuales)
Control de errores/flujo	Método de tipo ventana deslizante <ul style="list-style-type: none">• Numeración de segmentos• Confirmación• Retransmisión	Sin control de errores ni flujo
Ejemplos	Telnet, FTP, HTTP, SMTP, POP3...	DNS, RIP, SNMP, DHCP...

El modelo cliente-servidor, puertos y sockets

El modelo cliente-servidor

- **El modelo cliente-servidor**

- Es el patrón de comunicación usado por la mayoría de las aplicaciones de Internet (tanto TCP como UDP)

- **Cliente**

- Es la **aplicación** o proceso **que inicia la conexión** o el intercambio de datos con la máquina remota (servidor)
- La aplicación cliente normalmente la arranca un usuario cuando quiere utilizar un servicio de la red
- Ejemplos: Navegador web (TCP), cliente de e-mail (TCP), consulta al servidor DNS (UDP)

- **Servidor**

- Es la **aplicación** o proceso (también denominado servicio) **que recibe y acepta la solicitud de conexión** o intercambio de datos del cliente
- Esta aplicación normalmente está ejecutándose continuamente en la máquina remota y está a la espera de solicitudes de clientes
- Ejemplos: Servidor web (TCP), servidor de correo electrónico (TCP), servidor DNS (UDP)

Puertos

- **Puertos**

- Toda aplicación o proceso de red (cliente o servidor) ejecutándose en un computador está identificada por un **número de puerto** único dentro de dicho sistema
 - El puerto es un identificador de 16 bits
 - Se usa tanto en aplicaciones basadas en TCP como en UDP

- **Puerto de un proceso cliente**

- Cuando el usuario arranca el proceso cliente, **el SO de la máquina cliente le asigna un número de puerto libre**
- Este número de puerto es siempre un número > 1023 (puertos efímeros)
(Los n° de puerto ≤ 1023 se usan para puertos servidores con privilegios de superusuario)

- **Puerto de un proceso servidor**

- Cuando un proceso cliente solicita una comunicación con un servidor, **el cliente debe conocer de antemano el número de puerto del servidor**
- Habitualmente, cada tipo de servidor utiliza un n° de puerto fijo, denominado puerto bien conocido (well-known port), que suele ser el mismo en todos los sistemas

Puertos

- **Puerto de un proceso servidor**
 - Ejemplos de puertos bien conocidos

Servidor	Puerto	Servidor	Puerto
FTP	20 y 21	DNS	53
SSH	22	HTTP	80
TELNET	23	POP3	110
SMTP	25	NTP	123

- En un sistema Linux, el archivo `/etc/services` contiene el puerto bien conocido asociado a cada tipo de servicio
- Los servicios clásicos de Internet suelen tener un puerto bien conocido ≤ 1023 , pero existen multitud de servidores con puertos > 1023

Sockets

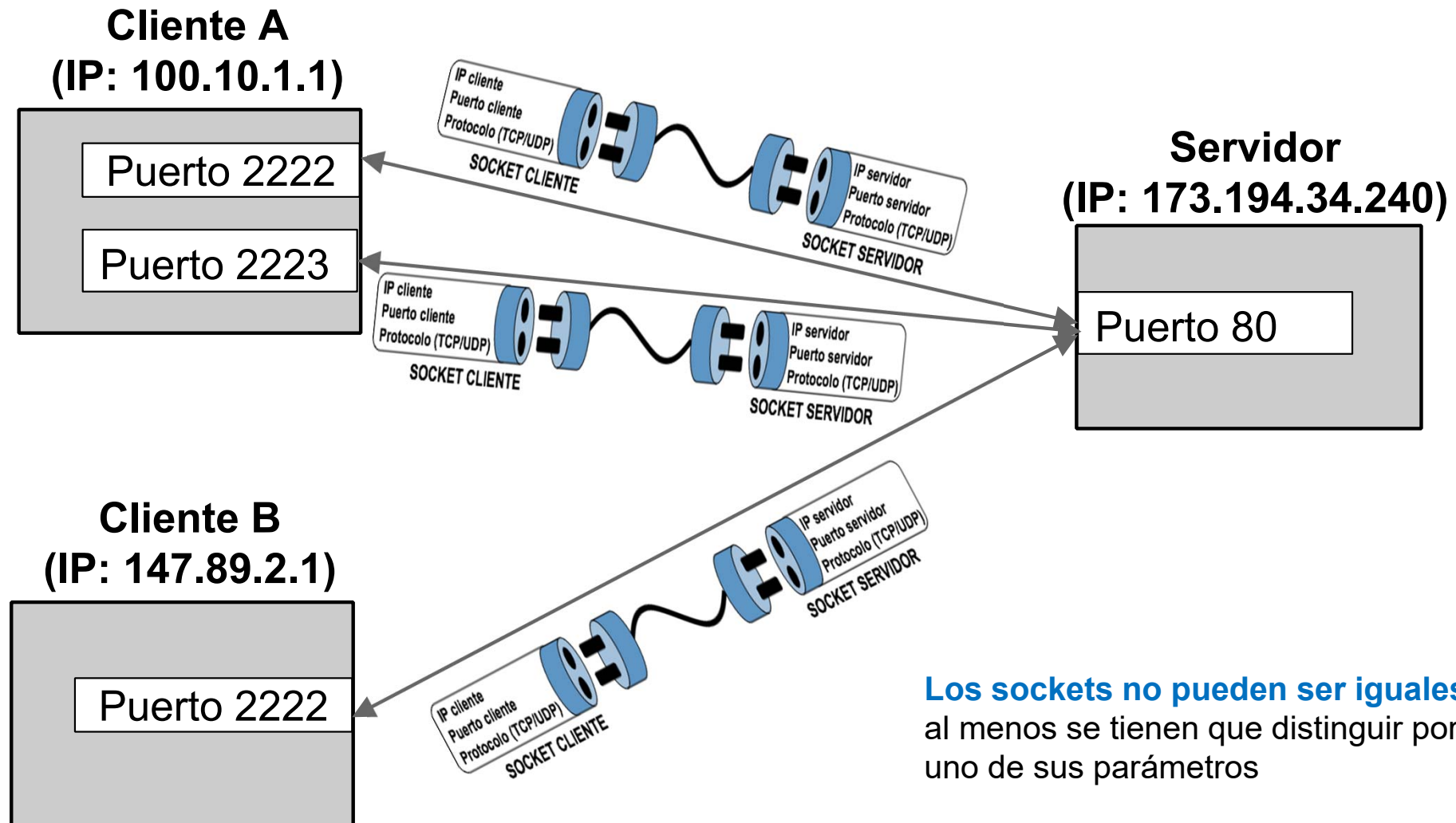
- Cuando **se establece un canal de comunicación** entre cliente y servidor (TCP o UDP) **los extremos de dicho canal se denominan “sockets”** (enchufe)
 - Un socket **se identifica (de forma única)** mediante **3 parámetros**
 - Dir. IP, nº de puerto y protocolo (TCP o UDP)
 - Una vez creados los sockets en ambos extremos, éstos **permiten el intercambio de datos bidireccional entre cliente y servidor**



- Protocolo (TCP o UDP, el mismo para ambos extremos)
- Dir. IP cliente
- Puerto cliente
- Dir. IP servidor
- Puerto servidor

Servidores concurrentes

- Es necesario **atender concurrentemente a múltiples usuarios**
- El servidor identifica cada comunicación de forma única por los parámetros del socket



El protocolo UDP

UDP: Características

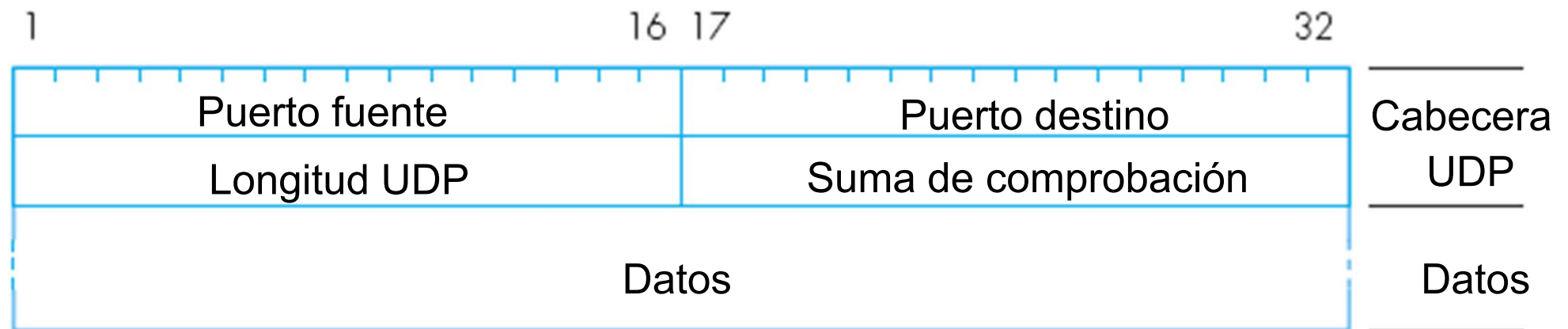
Características

- El protocolo UDP es un protocolo sin conexión y no fiable
- El receptor no envía confirmación de la recepción de los mismos
- UDP no garantiza:
 - la recuperación de datagramas perdidos o erróneos
 - la presentación ordenada de datagramas
 - la eliminación de duplicados

Desde el punto de vista de la aplicación

- Cualquier control de errores debe gestionarse en la capa de aplicación
- Si es necesario, el reensamblado también es necesario hacerlo en la capa de aplicación (mensaje aplicación > mensaje de red)

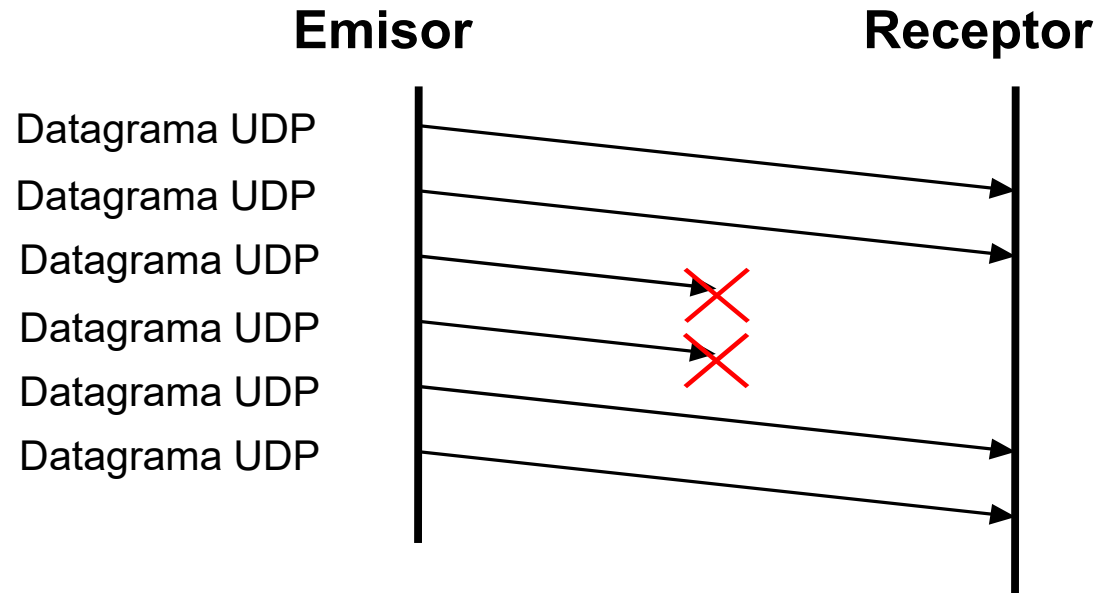
UDP: Formato del datagrama



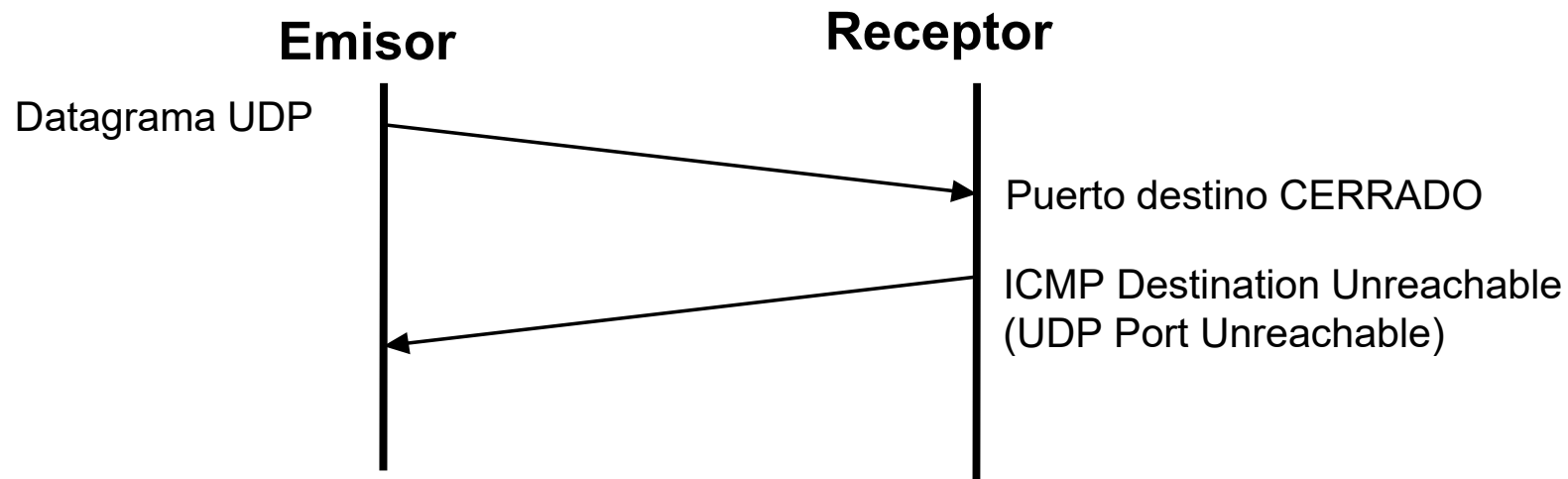
- **Puerto origen y destino:** Identifican a las aplicaciones que se comunican
- **Longitud UDP:** Longitud total en bytes ($2^{16} = 64\text{KB}$)
- **Suma de comprobación:** el complemento a 1 de la suma en complemento a 1 de todas las palabras de 16 bits que forman parte de:
 - El datagrama UDP (datos + cabecera)
 - Una pseudo-cabecera formada por información de la cabecera IP:
 - dirección IP fuente y destino
 - campo Protocolo
 - longitud UDP

El protocolo UDP: Mecanismos de transmisión

- Envío de paquetes UDP a un **puerto** de servicio **ABIERTO**



- Envío de paquetes UDP a un **puerto** de servicio **CERRADO**



El protocolo TCP

El protocolo TCP: Características

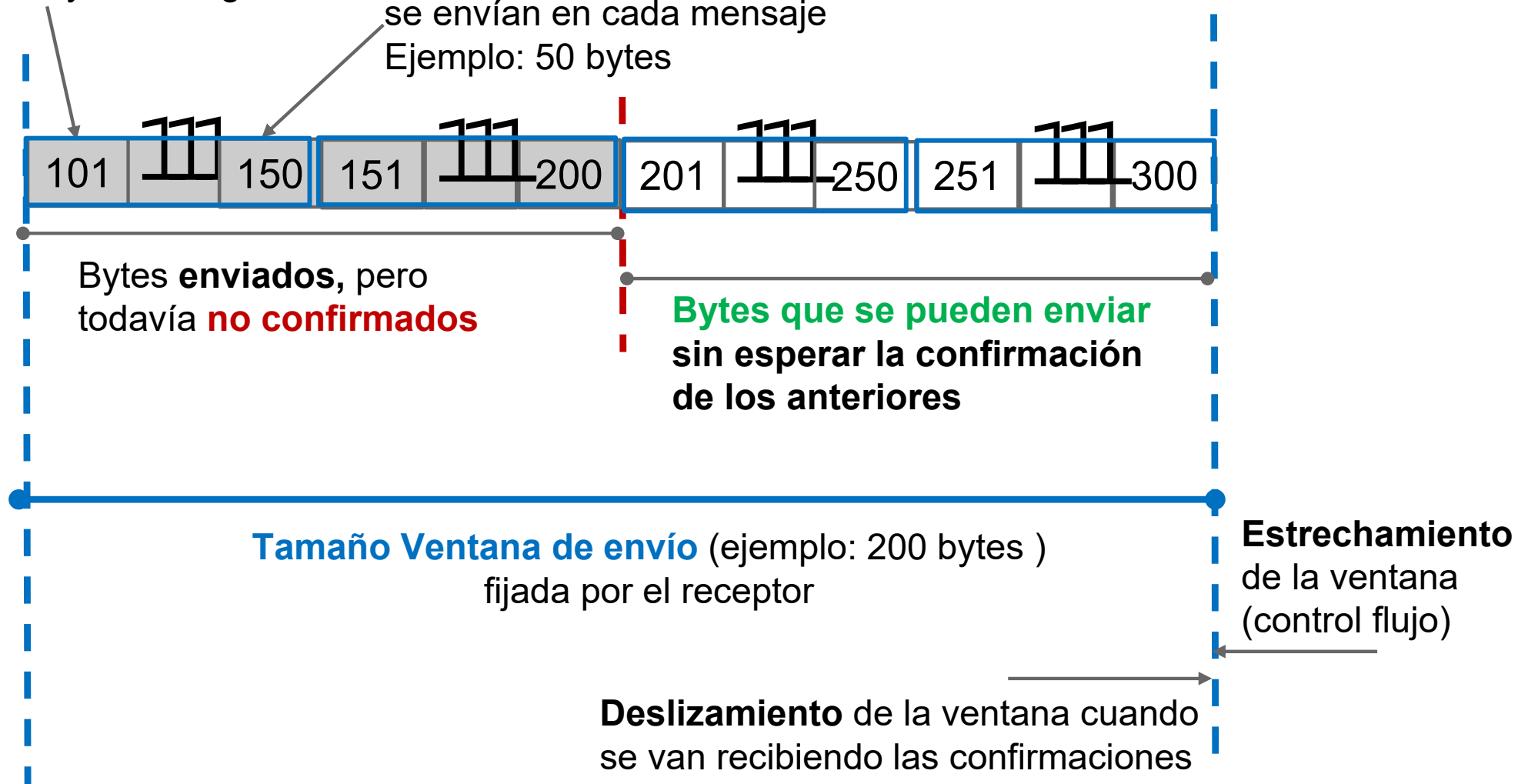
- **Unidad de transferencia:** **Segmento** TCP
- **Fases en una transmisión:**
 1. Establecimiento de conexión
 2. Transferencia de datos
 3. Cierre de conexión
- **Mecanismos de control de errores:** **Tipo ventana deslizante**
 - Numeración de segmentos
 - Cada segmento lleva un **número de secuencia** de 32 bits
 - Indica la posición que tiene el primer byte de datos que envía el segmento dentro del mensaje
 - Confirmaciones del receptor
 - Cuando el receptor recibe un segmento de datos correcto y sin errores, envía una **confirmación** al emisor
 - Retransmisión de segmentos
 - Si transcurrido un tiempo desde que se envió el segmento, el emisor no recibe confirmación, entonces retransmite de nuevo el segmento

Mecanismos de control de errores: Tipo ventana deslizante

Ventana deslizante (envío)

Nº de secuencia = nº de byte en el flujo de datos.
Primer byte del segmento

Segmento, grupo de bytes que se envían en cada mensaje
Ejemplo: 50 bytes

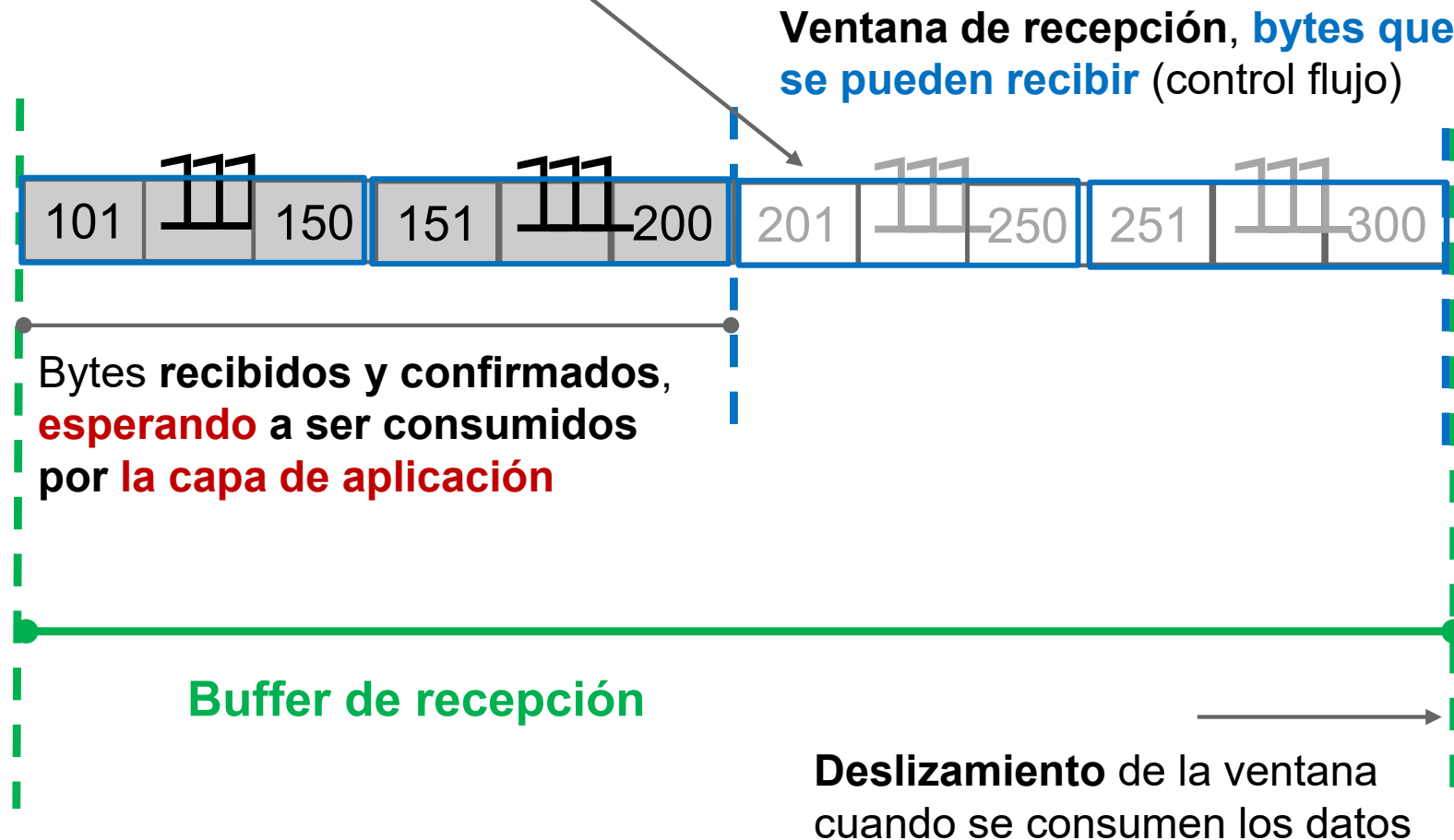


Mecanismos de control de errores: Tipo ventana deslizante

Ventana deslizante (recepción)

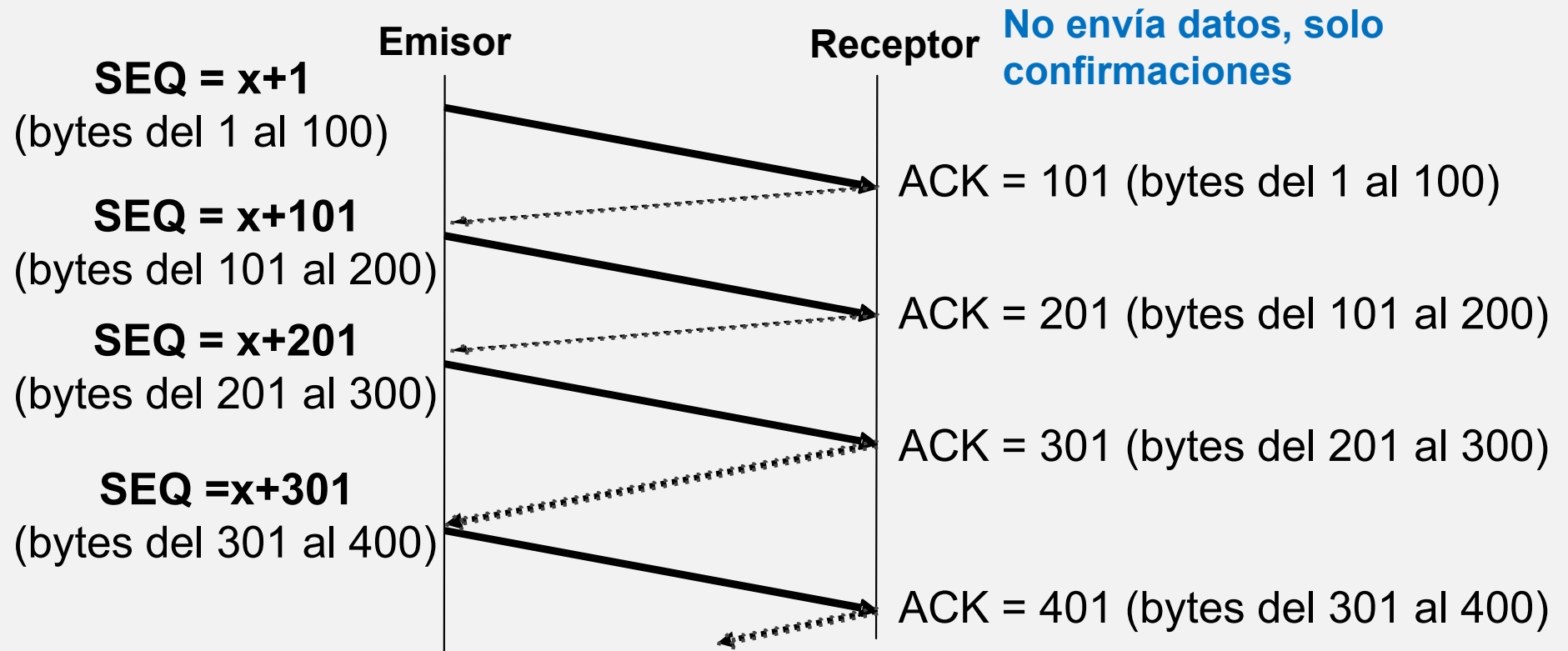
Números de confirmación (ACK), número del primer byte en el flujo de datos **que se espera recibir**

- Acumulativos, confirman todos los bytes anteriores al de ACK
- *Piggybacking*, se solapan con el envío de datos



Mecanismos de control de errores: Tipo ventana deslizante

Ejemplo: Transmisión sin errores. Tamaño de la ventana = Tamaño del segmento = 100 bytes. Luego no se puede enviar el siguiente segmento hasta que no se recibe confirmación

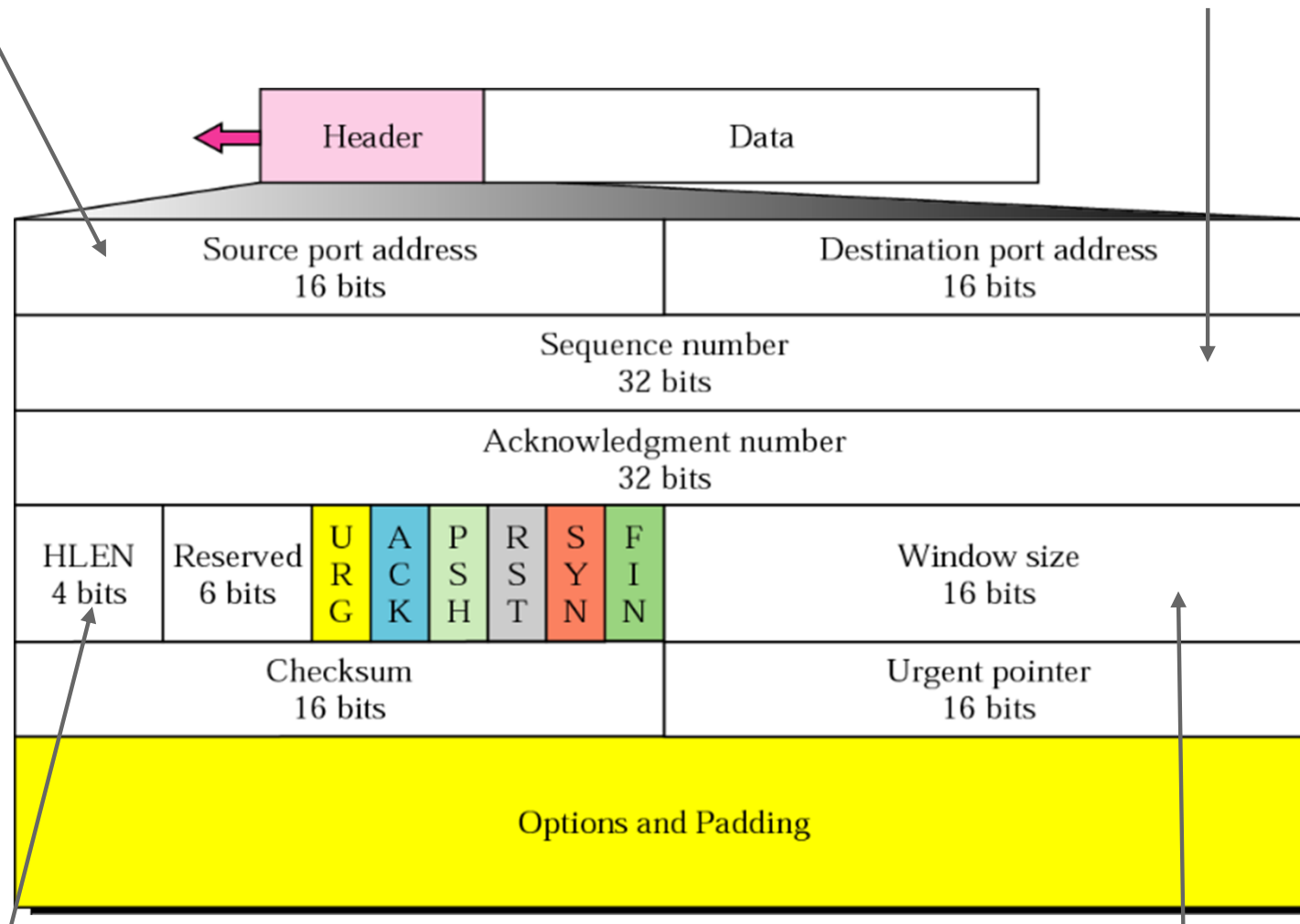


¿Qué cambiaría si Tamaño de la ventana = 100 bytes y Tamaño del segmento = 50 bytes?

TCP: Formato del Segmento

Puertos, identifican los extremos de la conexión

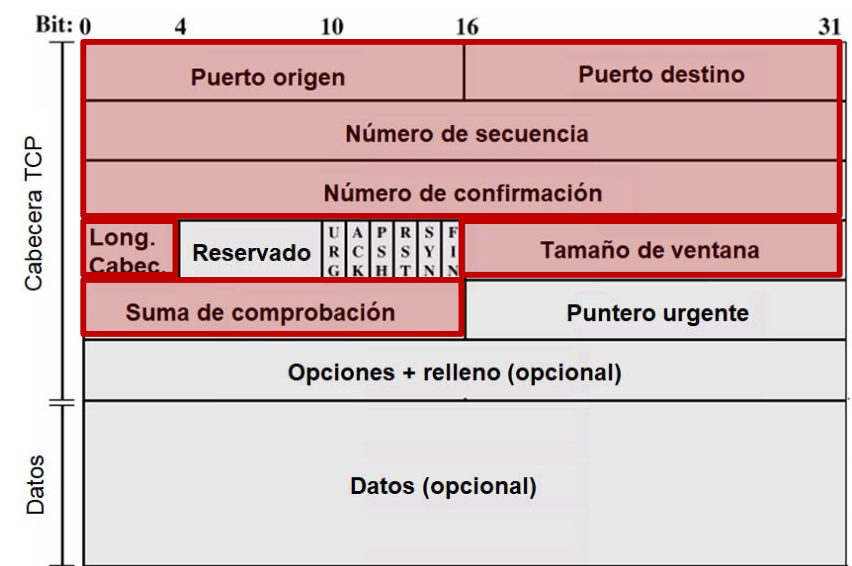
Números de secuencia y confirmación, expresados en bytes en el flujo de datos



Longitud de la cabecera en palabras de 32 bits (20-60 bytes)

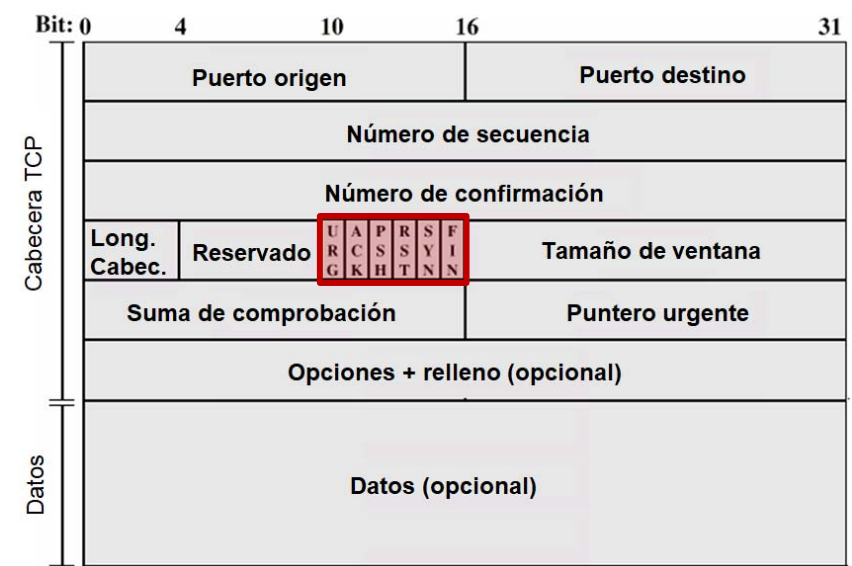
Tamaño de la ventana expresado en bytes (control de flujo)

TCP: Formato del segmento



- **Puerto origen y destino**
 - Identifican los extremos de la conexión
- **Nº de secuencia**
 - Se usa para determinar la posición del primer byte de datos que envía el segmento con respecto al mensaje original
- **Nº de confirmación**
 - Para enviar confirmaciones superpuestas en sentido contrario
 - Indica el nº de secuencia del siguiente byte que se espera recibir
- **Longitud de la cabecera**
 - Medida en palabras de 32 bits
- **Tamaño de ventana**
 - Permite anunciar el tamaño de la ventana de recepción durante la conexión TCP
 - El valor del campo ventana indica la cantidad de bytes (relativos al nº de byte indicado en el campo nº de confirmación) que el receptor es capaz de aceptar
- **Suma de comprobación:** se calcula cómo en UCP

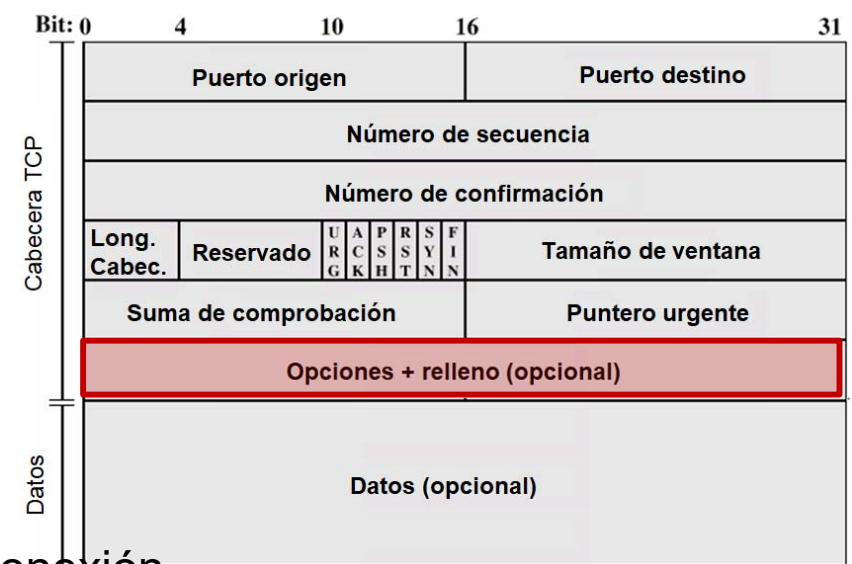
TCP: Formato del segmento



● Flags

- **URG:** indica si el segmento transporta datos urgentes al principio
 - El **puntero urgente** indica en este caso el último byte de datos urgente (desde el número de secuencia)
- **ACK:** indica si el segmento lleva un número de confirmación válido
 - Todos los segmentos de una conexión TCP, excepto el primero, transportan un número de confirmación válido (con ACK=1)
- **PSH:** indica si los datos deben ser pasados inmediatamente a la aplicación
 - Si no se activa, los datos se pueden almacenar en un buffer de recepción y éstos se pasan a la aplicación cuando el buffer se llena
- **RST:** utilizado para abortar una conexión
- **SYN:** utilizado en el establecimiento de la conexión
 - Significa que los extremos deben sincronizar los números de secuencia iniciales de la transmisión
- **FIN:** utilizado en la finalización de la conexión

TCP: Formato del segmento

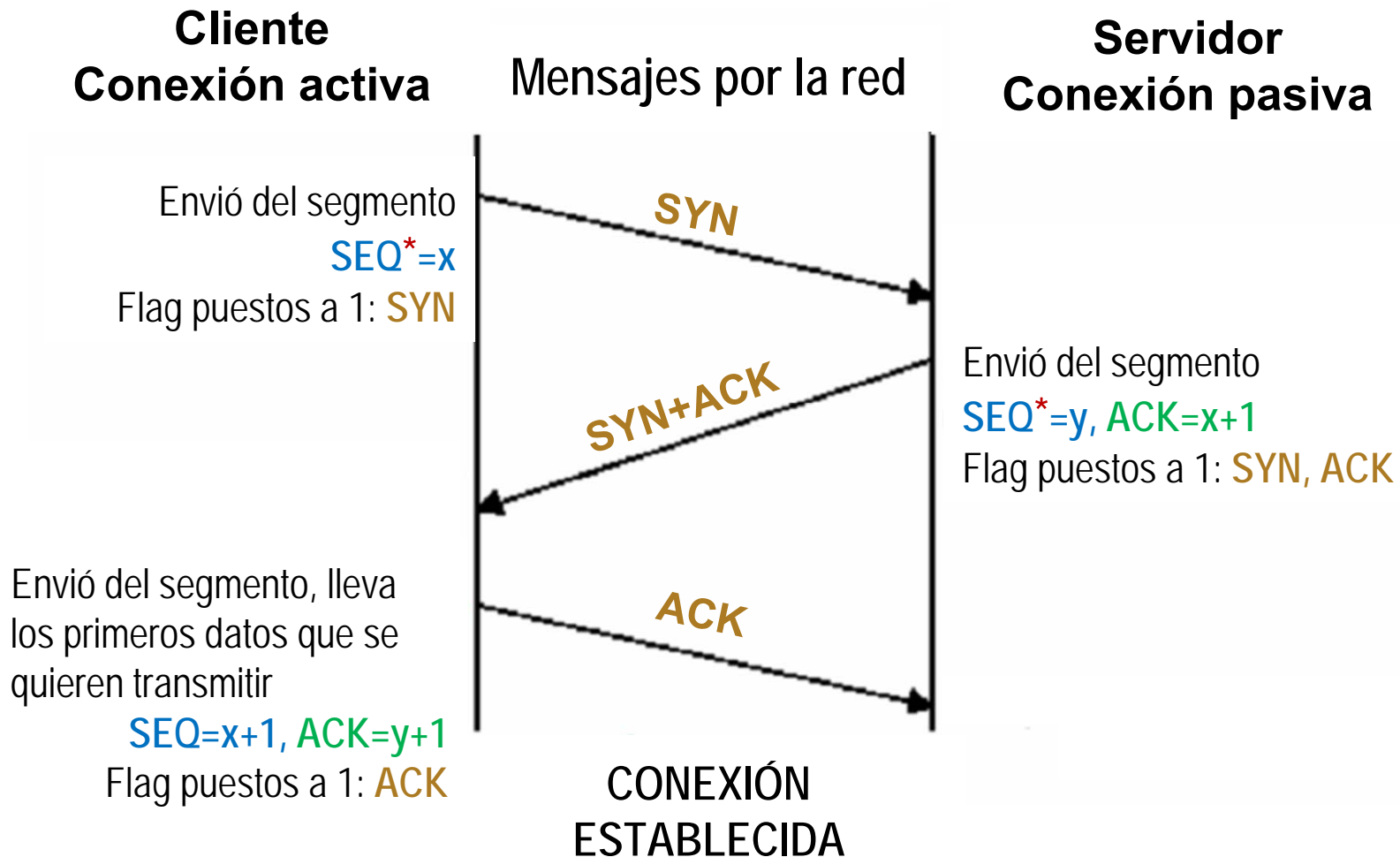


- **Opciones**

- Permite negociar parámetros adicionales de la conexión
- Principales opciones
 - **Tamaño máximo del segmento** (MSS, Maximum Segment Size)
 - Permite negociar el tamaño máximo del bloque de datos que contienen los segmentos que se envían al destinatario
 - Si no se negocia el MSS al inicio de una conexión, se establece un valor por defecto de 536 bytes
 - Factor de escala de ventana
 - Permite trabajar con ventanas mayores de 2^{16} bytes
 - Sello de tiempo (timestamp)
 - Permite introducir la hora de envío de un segmento
 - Se utiliza para calcular el tiempo de ida y vuelta de los segmentos y sus correspondientes confirmaciones
 - Opción de confirmación selectiva (SACK, Selective ACK)
 - Permite confirmar segmentos fuera de orden

Protocolo TCP. Fases en una transmisión

Establecimiento de conexión de 3 vías (3-way handshake)

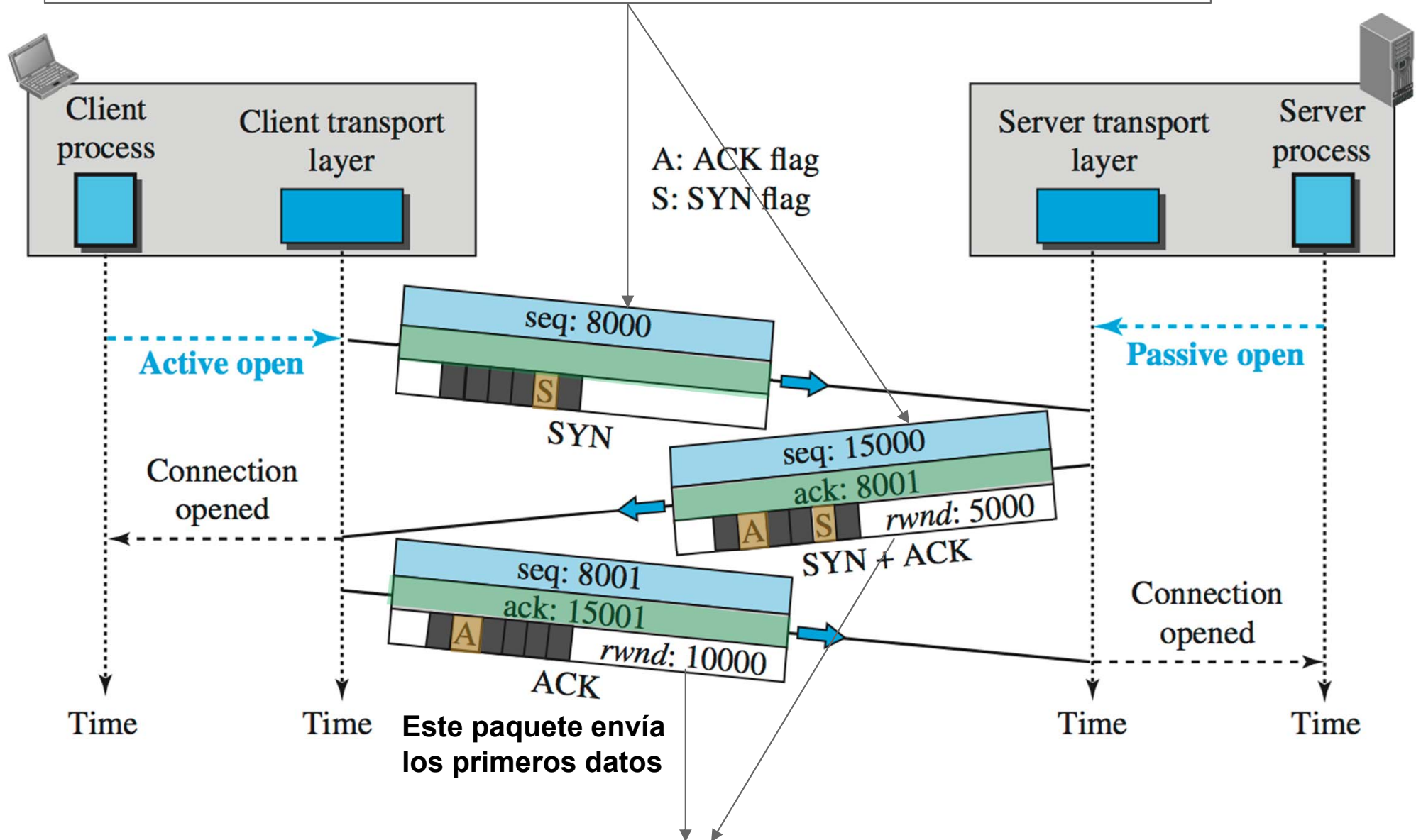


NOTA: Si en los segmentos:

- **NO se envían datos** -> SEQ^* = N° de secuencia inicial
- **SI se envían datos** -> SEQ = N° de secuencia inicial + primer byte del conjunto de datos que se envían

Establecimiento de conexión de 3 vías

Números de secuencia iniciales, sincronizan ambos extremos



Tamaños de ventana, junto con confirmaciones en ambos sentidos

Envío de datos

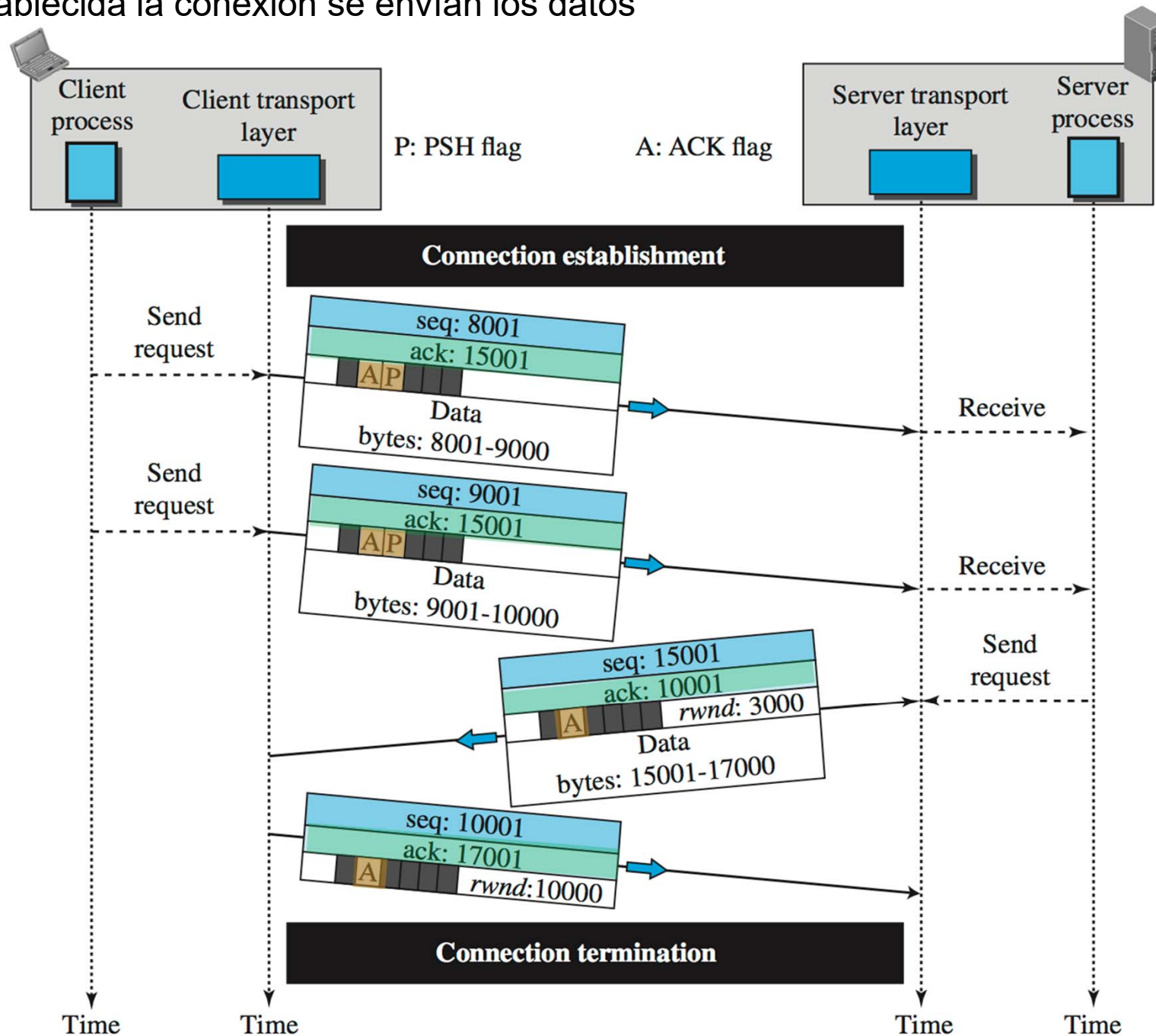
Mecanismos de transmisión

- El **tamaño máximo del segmento** es fijado de forma independiente por cada extremo en el campo Options
- El emisor envía todos los segmentos numerados
 - El campo n° de **secuencia** indica la **posición del primer byte del segmento** con respecto al inicio de la conexión
- El receptor envía una confirmación por cada segmento recibido
 - El campo n° de **confirmación** siempre **contiene el identificador del siguiente byte que se espera recibir**
 - Son **acumulativas** y se **solapan con** los envíos de **datos** (piggyback)
 - Si no hay datos que enviar, las confirmaciones de segmentos en orden se retrasan un máximo de 500 ms para solaparlas con un envío de datos (dependerá del tamaño de la ventana deslizante del emisor)
 - Los segmentos fuera de orden que provocan huecos se confirman inmediatamente (esto provocará una retransmisión rápida, como veremos)
 - Los segmentos fuera de orden que completan huecos se confirman inmediatamente

Envío de datos

Una vez establecida la conexión se envían los datos

- En este ejemplo en cada paquete
- El cliente está enviando **1000 Bytes**
 - El servidor está enviando **2000 Bytes**

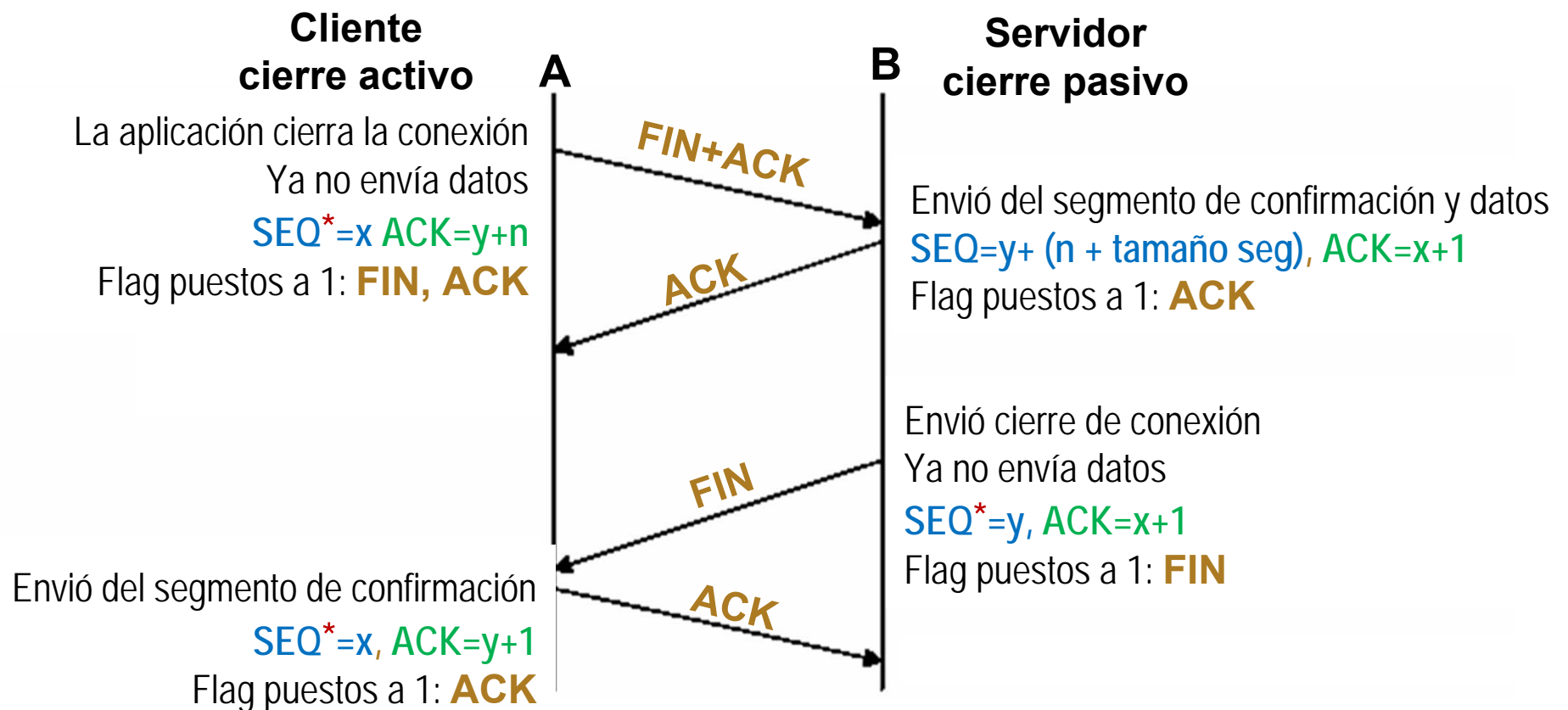


TCP: Cierre de conexión de 4 vías

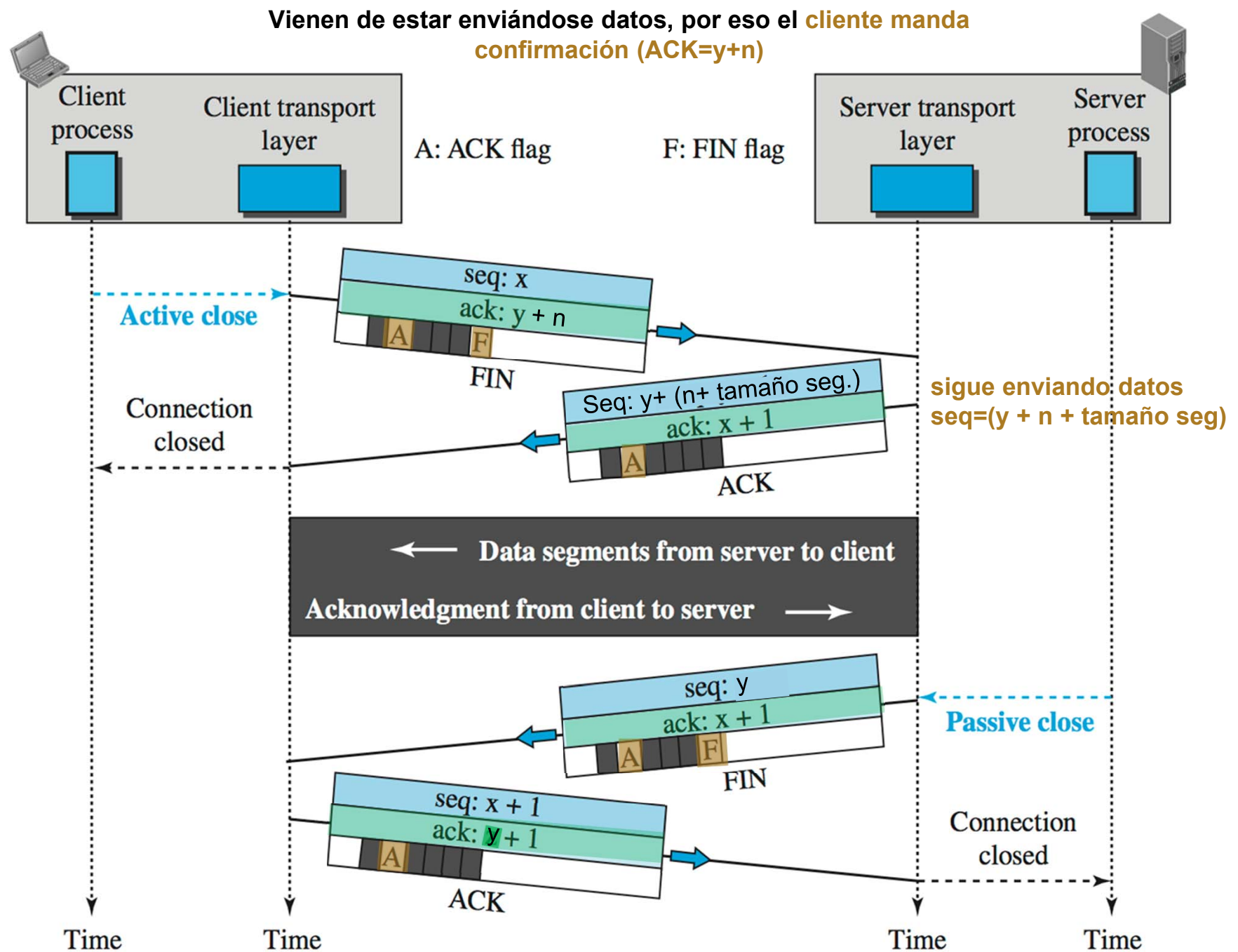
Mecanismo de cierre de conexión: Protocolo de 4 vías

1. El cliente ha terminado de transmitir y cierra primero la conexión, pero puede recibirlos y devolver confirmaciones
2. El servidor confirma el mensaje, pero no inicia la finalización de su extremo porque le quedan datos por enviar
3. Cuando el servidor acaba de enviar datos cierra la conexión
4. El cliente confirma el mensaje

Vienen de estar enviándose datos, por eso el
cliente manda confirmación ($ACK=y+n$)



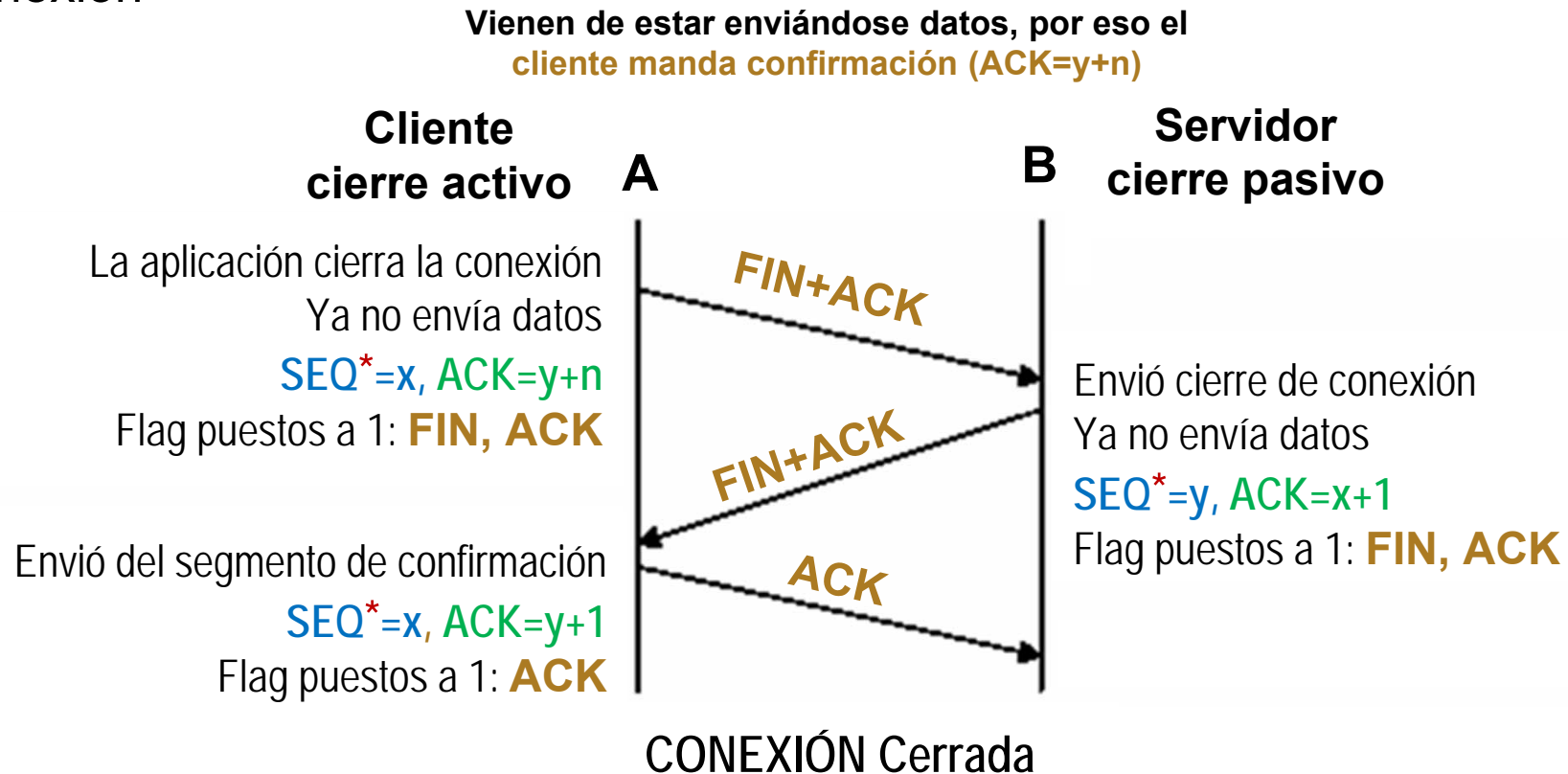
TCP: Cierre de conexión de 4 vías



TCP: Cierre de conexión de 3 vías

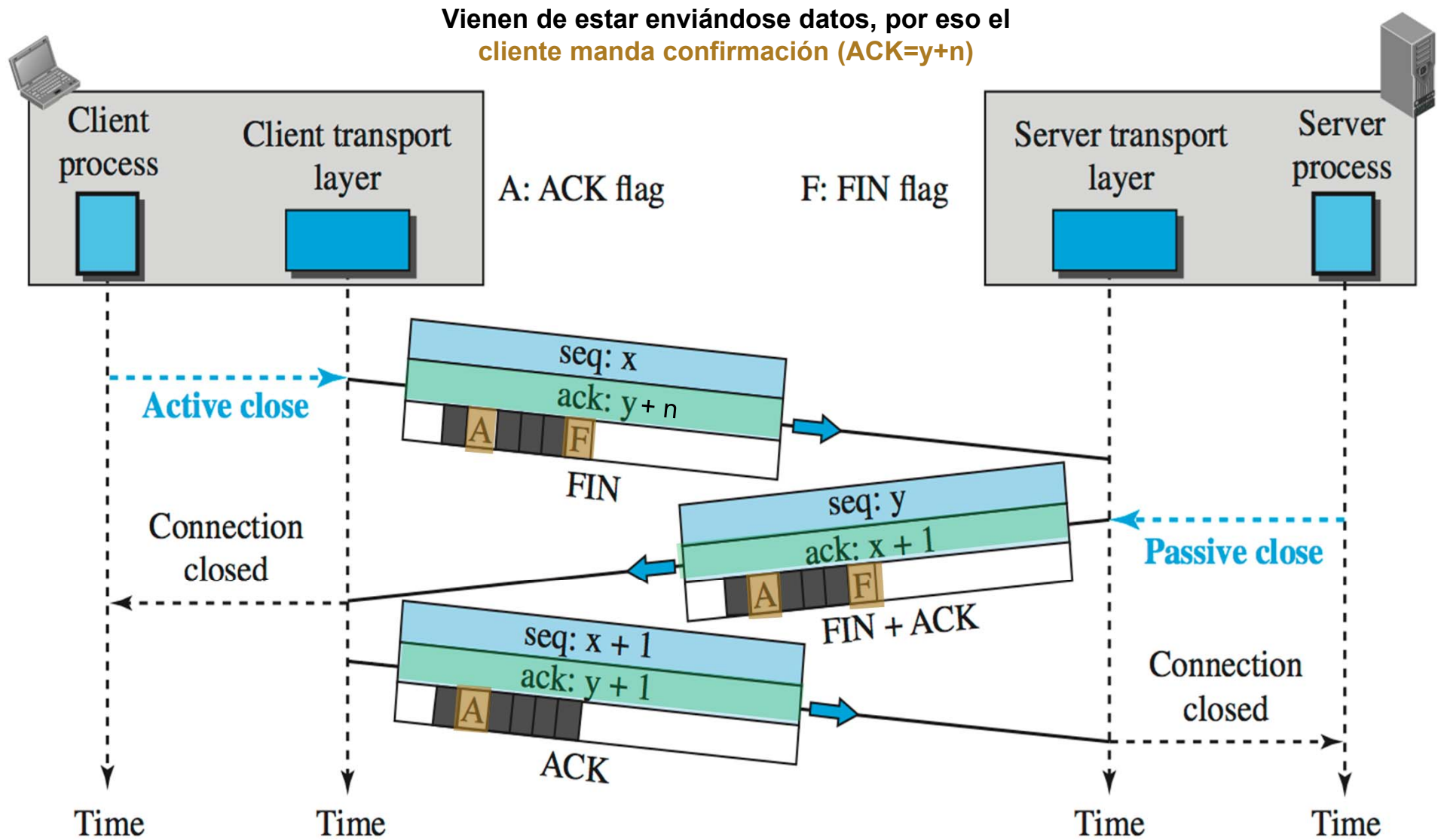
Mecanismo de desconexión: Protocolo de 3 vías

- Ambos extremos han terminado de transmitir y están de acuerdo en cerrar la conexión



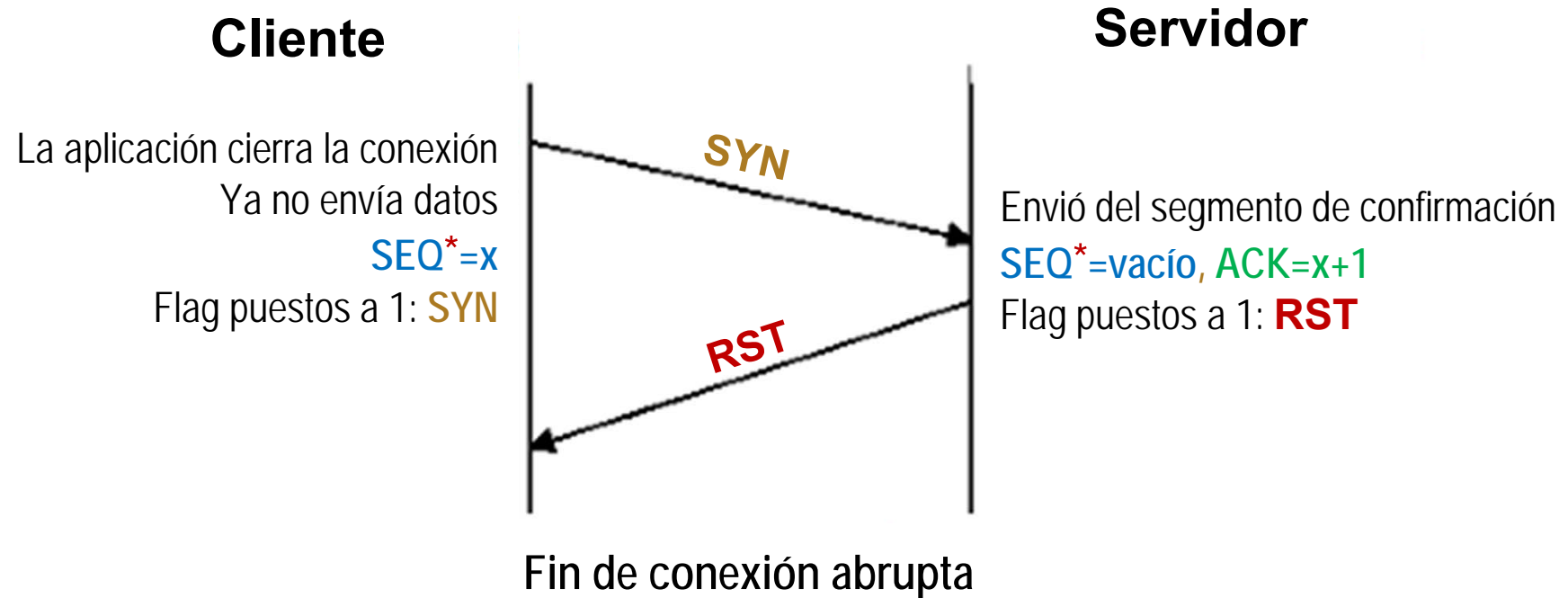
NOTA: En estos paquetes ya no se están enviando datos -> SEQ^* = Número de secuencia inicial

TCP: Cierre de conexión de 3 vías



TCP: Desconexión abrupta

- Comunicación abortada



Protocolo TCP. Control de errores

Mecanismos de control de errores

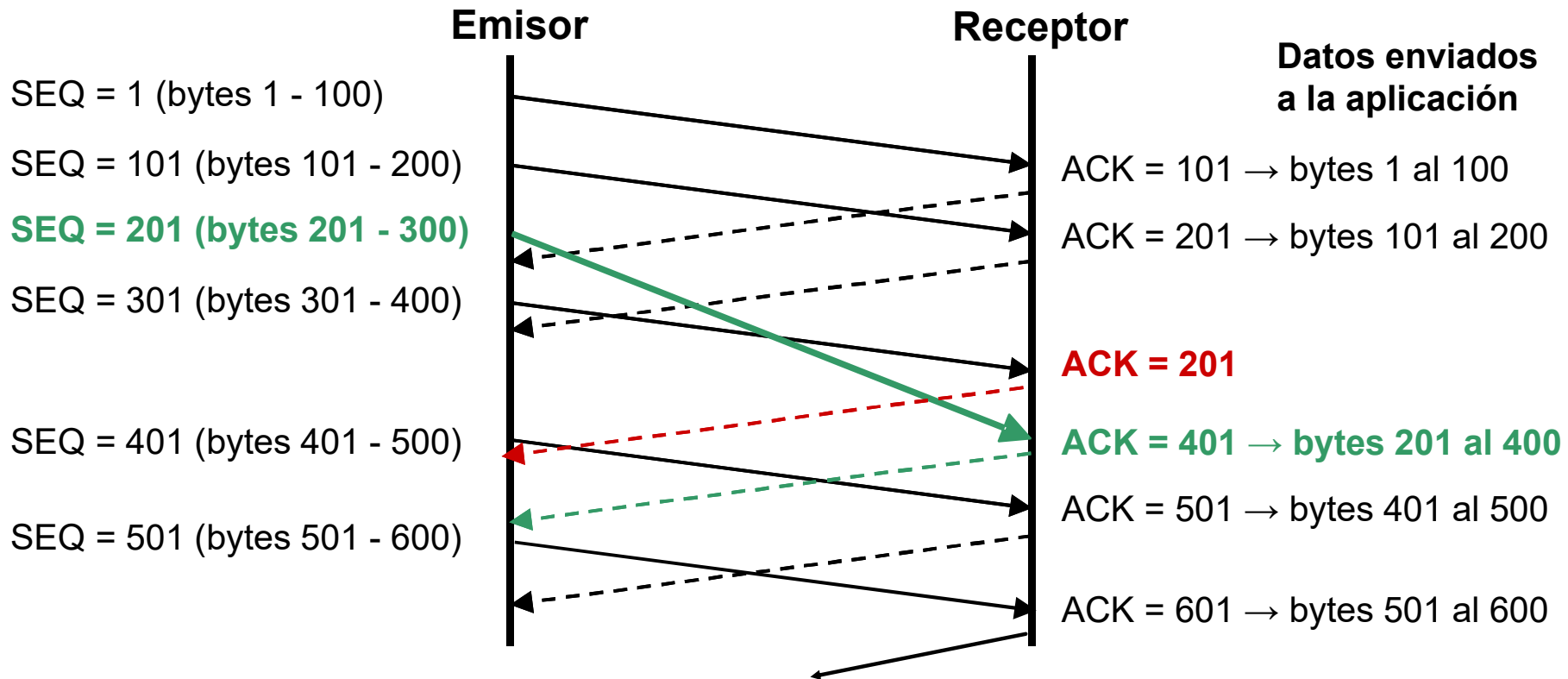
- **Simplificaciones adoptadas en los siguientes ejemplos:**
 - Sólo se transmiten datos en **un sentido** (Emisor → Receptor).
 - En el caso general, la transmisión es bidireccional
 - Se supone que todos los **segmentos** transportan **100 bytes** de datos
 - En el caso general, los segmentos pueden transportar diferente número de bytes, hasta un máximo fijado por el valor del MSS (Maximum Segment Size)
 - Se supone que el **número de secuencia inicial** del emisor es **SEQ=1**
 - En el caso general, el valor de los números de secuencia iniciales se acuerda durante el proceso de establecimiento de conexión TCP

Mecanismos de control de errores

- Ejemplo 1: **Retardo** en la transmisión **de un segmento**
 - El receptor recibe segmentos fuera de secuencia
 - Devuelve un ACK indicando el siguiente byte en secuencia que espera recibir
 - El emisor recibe un ACK fuera de secuencia
 - **No** inicia la **retransmisión** del segmento **inmediatamente** después de recibir el ACK
 - Un segmento sólo se retransmite **cuando se reciben tres ACK duplicados** para ese segmento
 - El receptor recibe el segmento retardado que le faltaba
 - Envía una **confirmación de toda la secuencia completa**

Mecanismos de control de errores

- Ejemplo 1: **Retardo** en la transmisión de un **segmento**

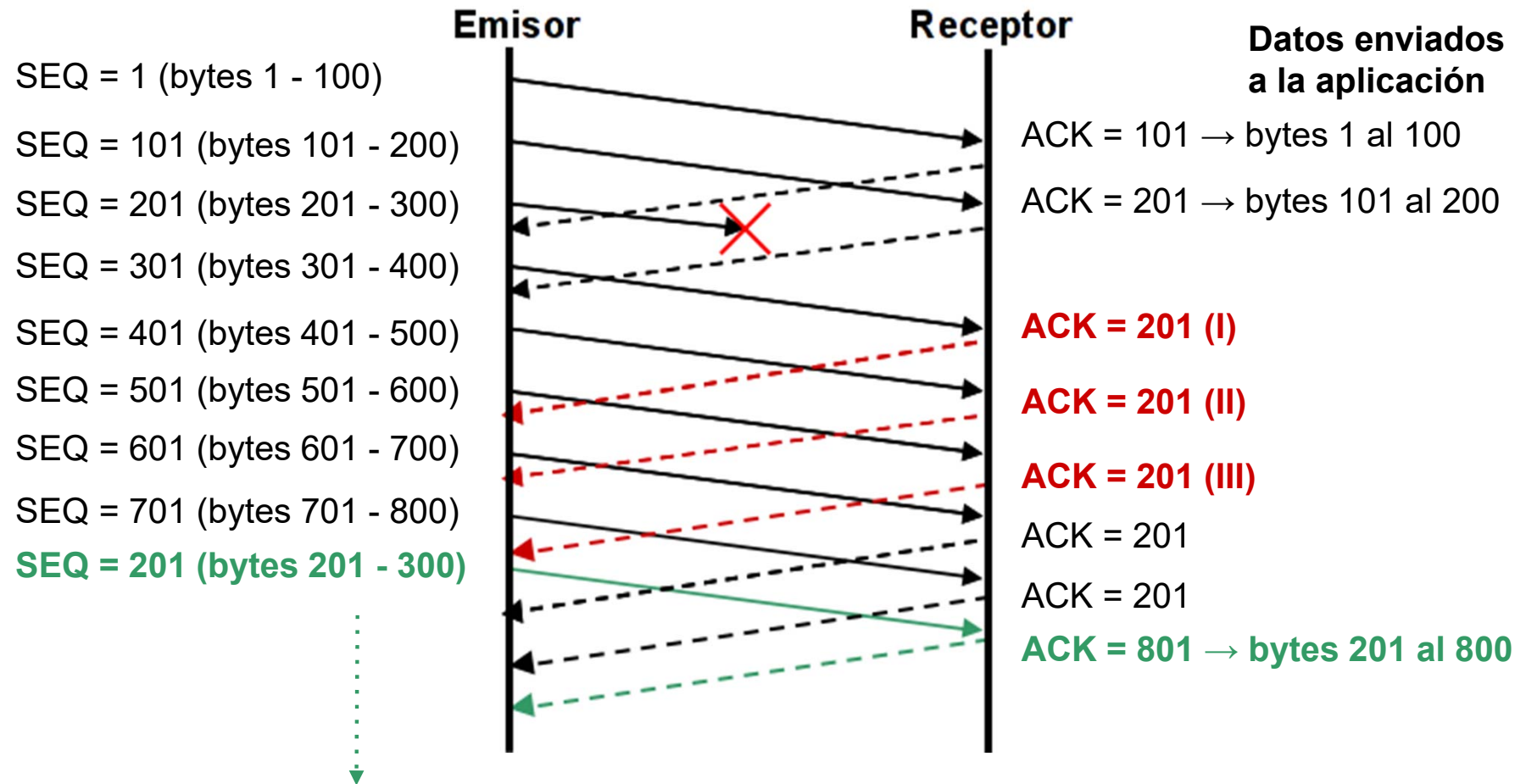


Mecanismos de control de errores

- Ejemplo 2: **Pérdida de un segmento**
 - El receptor recibe los segmentos fuera de secuencia:
 - Devuelve un ACK indicando el siguiente byte en secuencia que espera recibir
 - El emisor recibe **tres ACKs duplicados** con el mismo identificador:
 - Retransmite el segmento pendiente de confirmación
 - El receptor recibe el segmento retardado que le faltaba
 - Envía una **confirmación de toda la secuencia completa**

Mecanismos de control de errores

- Ejemplo 2: **Pérdida de un segmento**



Al recibir la 3ª confirmación duplicada con ACK=201, retransmite dicho segmento

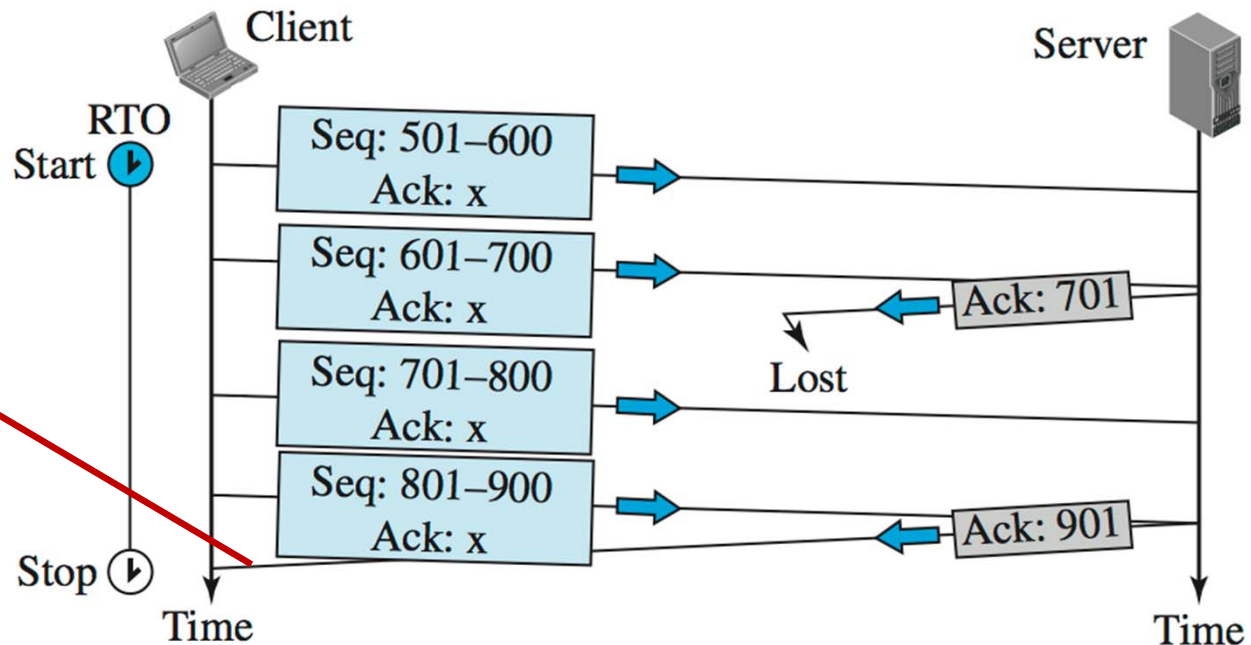
Mecanismos de control de errores

- Ejemplo 3: Temporizador de retransmisión

- Para contemplar la **posibilidad de que el emisor no reciba confirmaciones**
- Por cada nuevo segmento transmitido **se inicia un temporizador de retransmisión**
- El segmento **se retransmite si ha expirado el temporizador**

Caso 1: **Se pierde una confirmación** pero llega la siguiente confirmación antes de que expire el temporizador

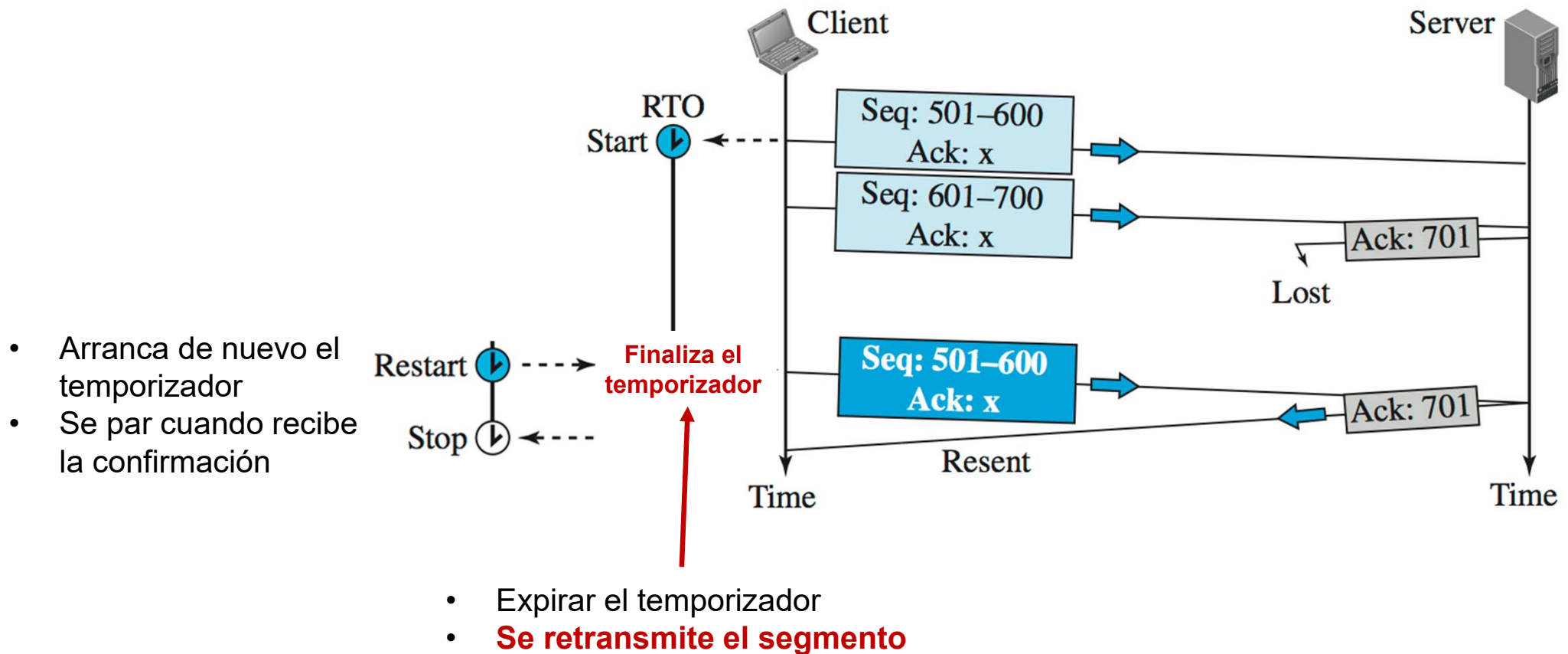
- Antes de expirar el temporizador le llega la confirmación Ack:901 (esto confirma que se ha recibido todo lo anterior)
- **El temporizador se para**
- **NO se retransmite el segmento**



NOTA: en este ejemplo se manda confirmación cada dos segmentos recibidos

Mecanismos de control de errores

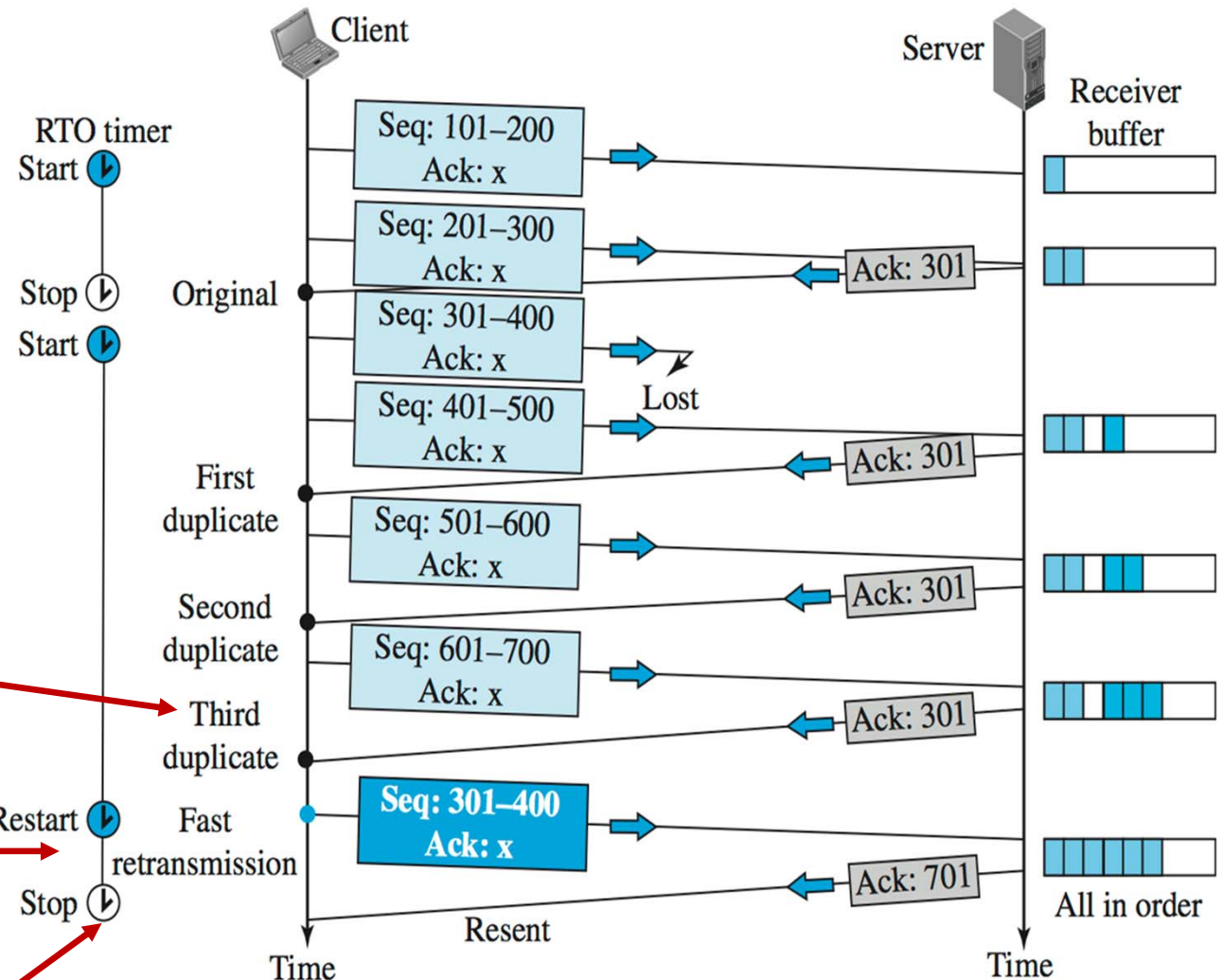
Caso 2: Temporizador de retransmisión expirado



Mecanismos de control de errores

Caso 3: **Se pierde un segmento** y llegan las tres confirmación duplicadas antes de que expire el temporizador

- Antes de expirar el temporizador le llegan tres confirmaciones Ack:301
- **Se se retransmite el segmento**
- Se reinicia el temporizador
- Se para el temporizador porque llega la confirmación Ack: 701



Mecanismos de control de errores: Temporizador

- TCP utiliza un mecanismo adaptativo
- La elección del tiempo de vencimiento del temporizador de retransmisión (timeout) está basada en los retardos observados en la red
- Los retardos en la red pueden variar dinámicamente, por tanto, los timeouts deben adaptarse a esta situación

Métodos para fijar los temporizadores de retransmisión

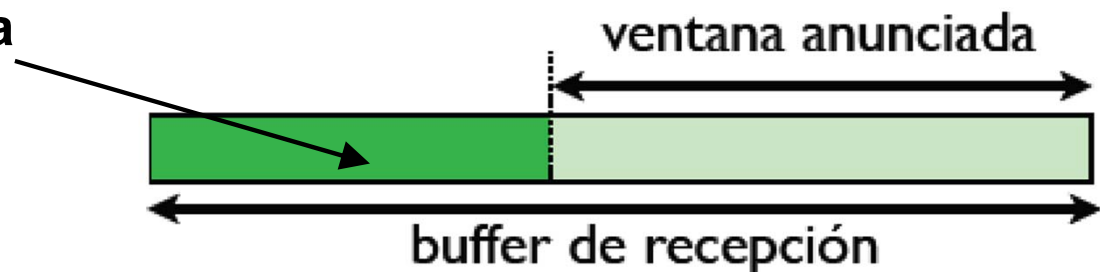
- **Método de la media ponderada (algoritmo de Jacobson)**
 - Mide el tiempo de ida-y-vuelta (round-trip) de cada segmento
 - El temporizador se ajusta a 2 veces el tiempo de ida-y-vuelta estimado
- **Método de la varianza (algoritmo de Jacobson/Karels)**
 - Corrige el método de Jacobson para situaciones de gran variabilidad
- **Algoritmo de Karn**
 - Como el anterior, pero descarta el uso de segmentos retransmitidos para el cálculo del tiempo de ida-y-vuelta

Protocolo TCP. Control de flujo

Mecanismo de Control de flujo

- El control de flujo en TCP se realiza mediante la **ventana de recepción**
- El **tamaño de esta ventana** indica el número de bytes que puede aceptar el receptor en un instante dado
 - Este tamaño viene determinado por el espacio libre disponible en el buffer de recepción utilizado para esa conexión TCP
 - El buffer de recepción almacena aquellos datos recibidos a través de la conexión TCP, hasta que estos son leídos por parte de la aplicación receptora
- El **tamaño de la ventana varía** a lo largo de una conexión
 - El tamaño de la ventana de recepción se anuncia en cada segmento de confirmación
 - Si el receptor anuncia una ventana de tamaño 0, el emisor no puede enviar más datos hasta que se anuncie un nuevo tamaño de ventana mayor

**Datos recibidos
(pendientes de lectura
por la aplicación)**



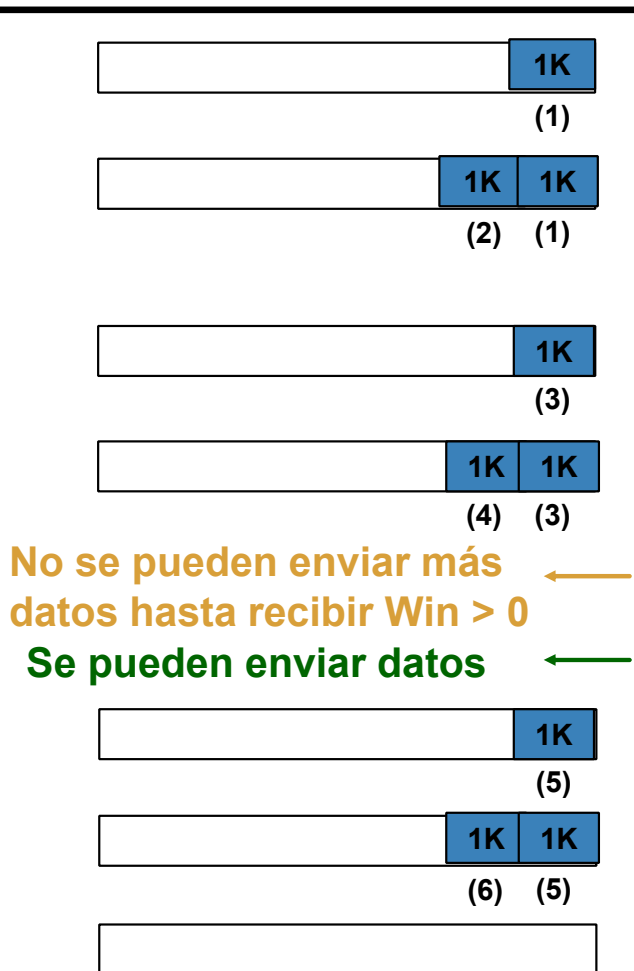
Mecanismo de Control de flujo

Ejemplo: El receptor es lento leyendo los datos (pasando los datos a la capa de aplicación)

- Buffer de transmisión de 8K, de recepción de 4K y segmentos de 1K
- Envío de ACK's cada dos segmentos

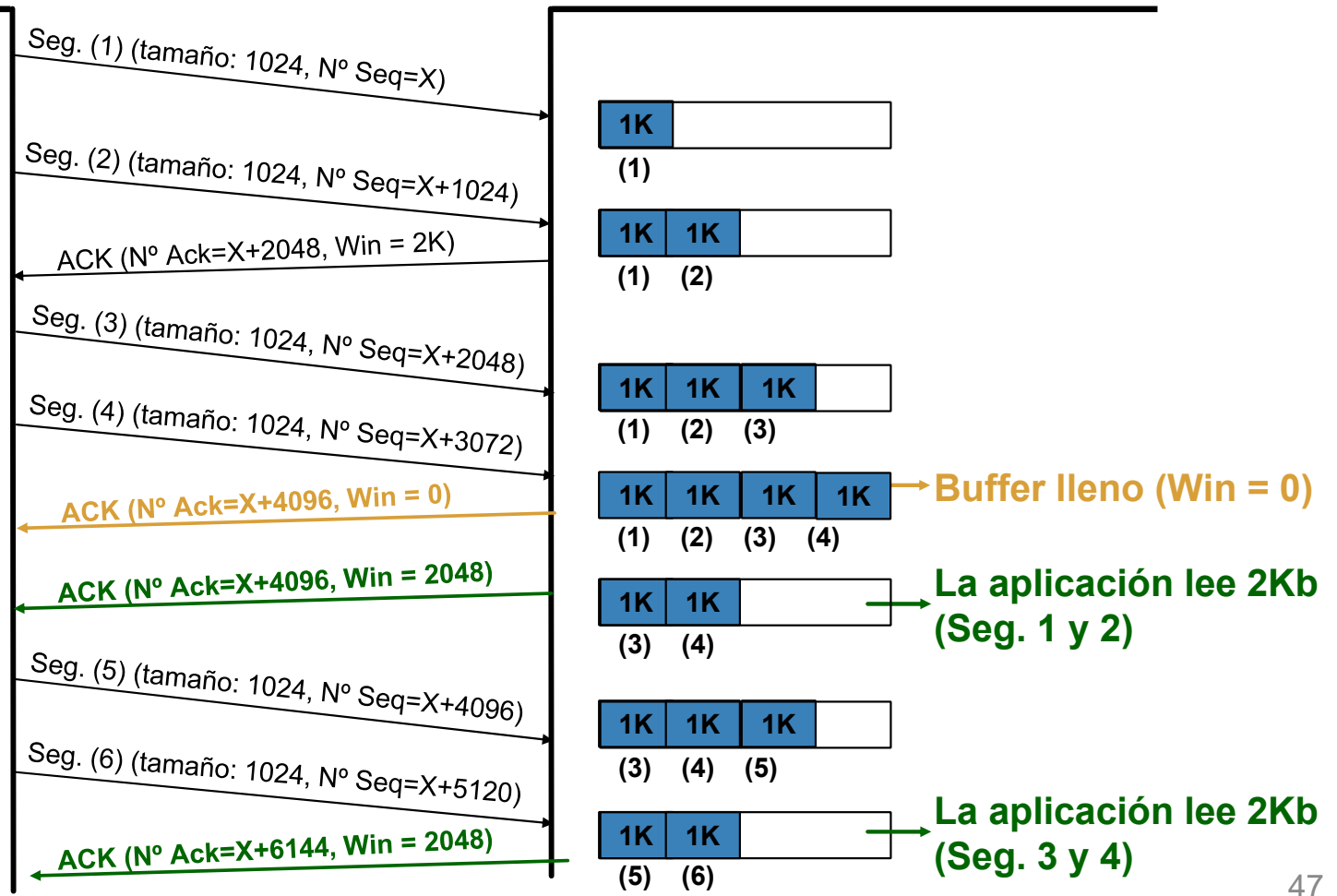
Buffer de transmisión (8K)

(segmentos enviados pdte. de confirmación)



Buffer de recepción (4K)

(segmentos recibidos pdte. de lectura)



TCP vs UDP para videojuegos es red

- ¿Qué protocolo es más utilizado?
- ¿Qué tipo de videojuegos se basan en cada protocolo?
- ¿Qué ventajas e inconvenientes presentan?

https://gafferongames.com/post/udp_vs_tcp/