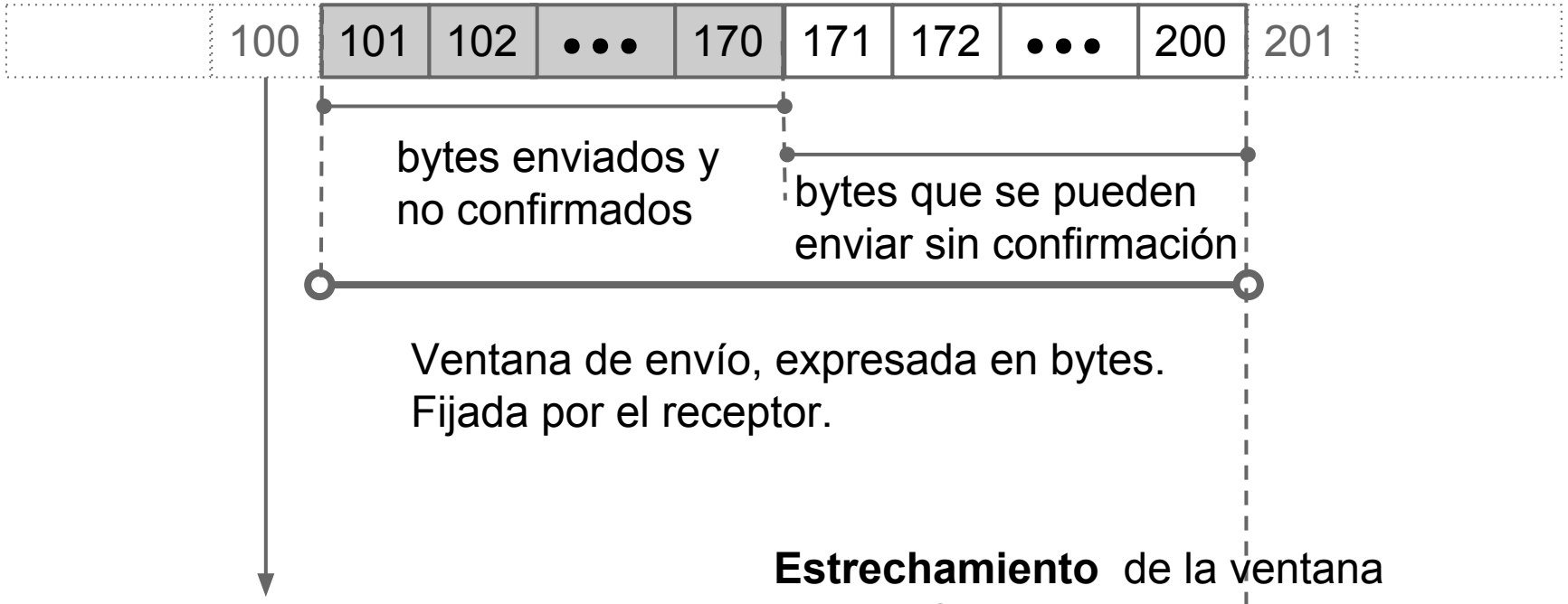
TEMA 3. Conceptos Avanzados del Protocolo TCP

PROFESORES:
Rafael Moreno Vozmediano
Rubén Santiago Montero
Juan Carlos Fabero Jiménez

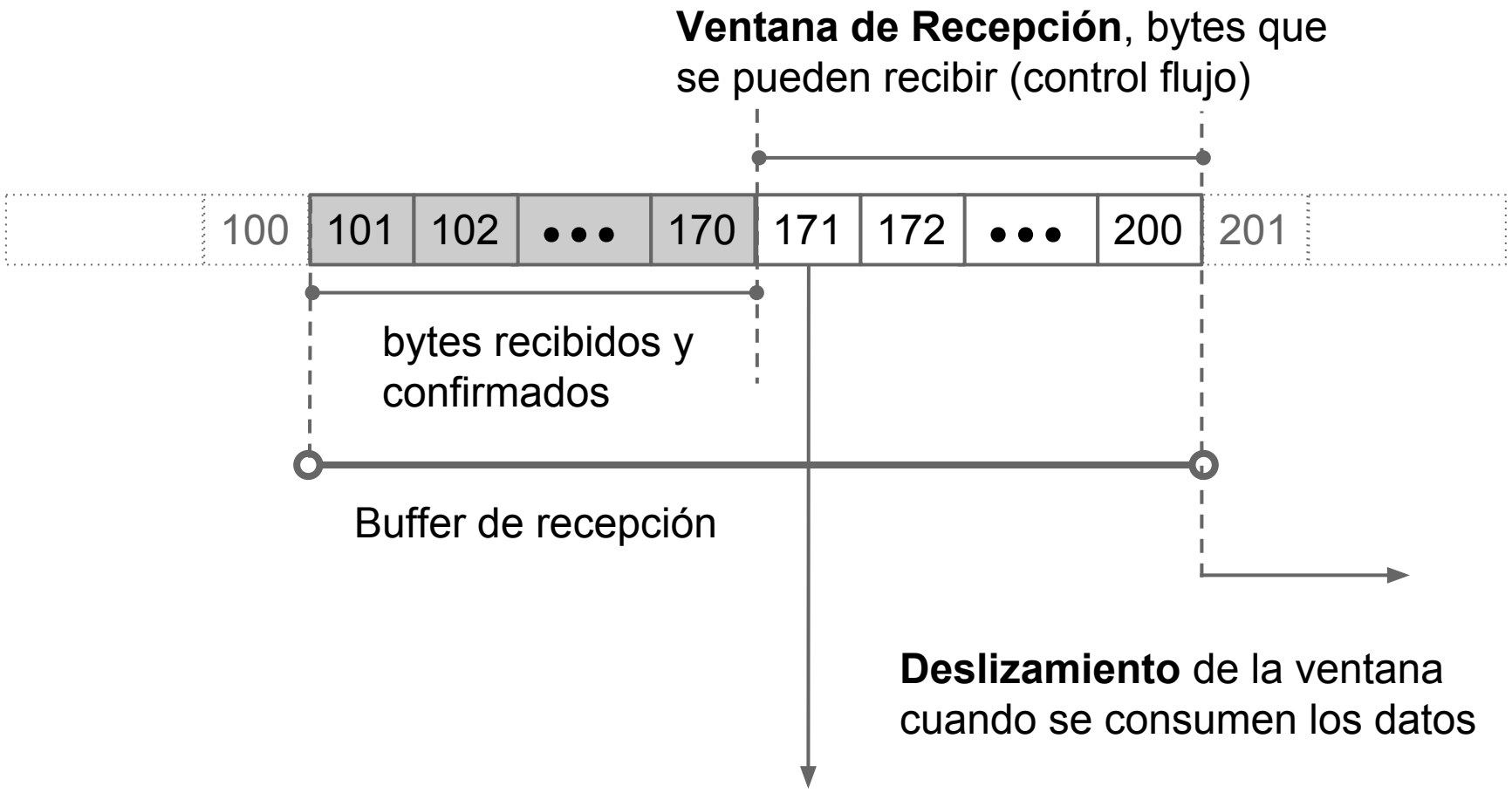
El protocolo TCP: Características

- Unidad de transferencia: Segmento TCP
- Orientado a conexión y fiable. Define las siguientes fases para la transmisión:
  - Establecimiento de conexión
  - Transferencia de datos
  - Cierre de conexión
- Mecanismos de control de errores de tipo ventana deslizante con:
  - Códigos de comprobación (*checksum*)
  - Numeración de segmentos
  - Confirmaciones selectivas y acumuladas, superpuestas del receptor
  - Retransmisión de segmentos perdidos o erróneos
  - Temporizadores
- Servicios ofrecidos por TCP:
  - Comunicación lógica proceso-proceso, usando números de puerto
  - Flujo de datos (stream) para el envío y recepción
  - Transmisión orientado a conexión y fiable
  - Full-duplex y multiplexación

Ventana Deslizante: La ventana de envío



Ventana Deslizante: La ventana de recepción

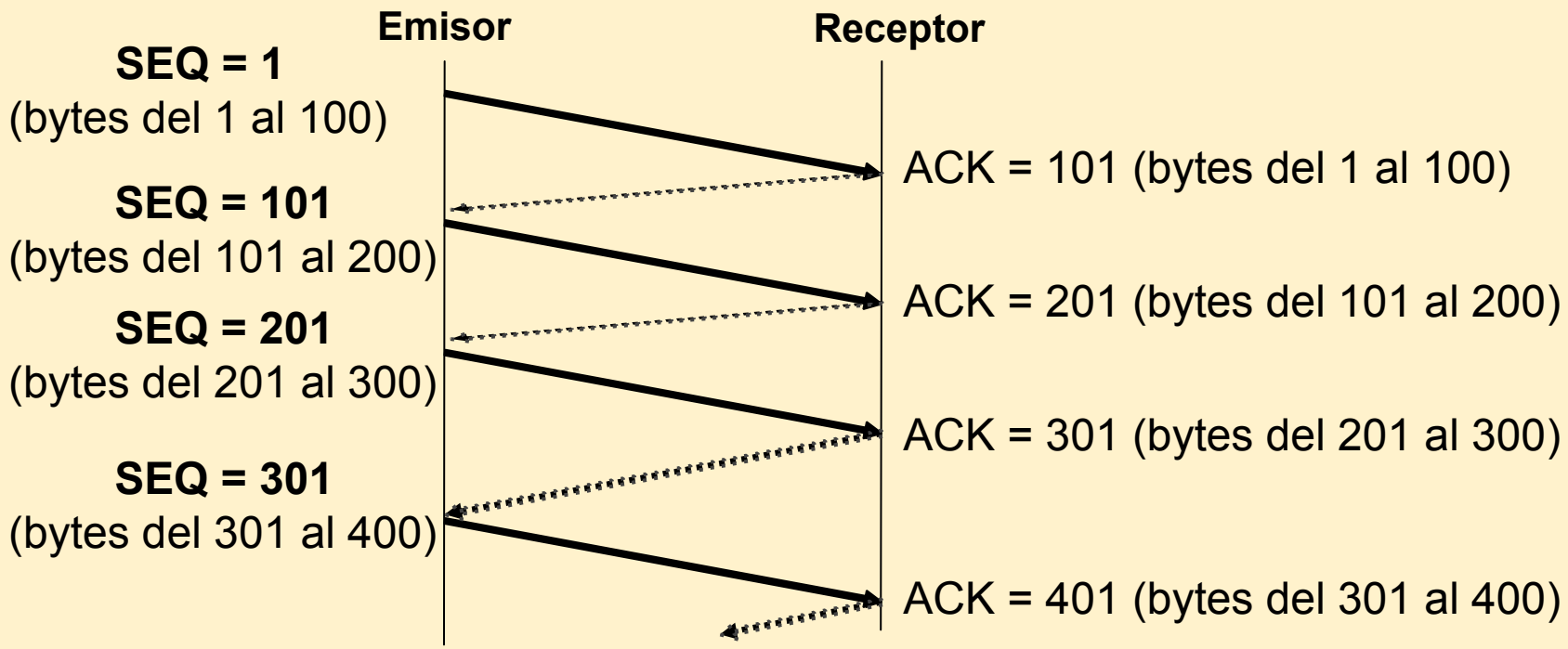


Números Confirmación (ACK), número del primer byte en el flujo de datos que se espera recibir.

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

Ventana Deslizante: Funcionamiento

Ejemplo: Transmisión sin errores. Tamaño de la ventana 100 bytes, Tamaño del segmento 50 bytes.

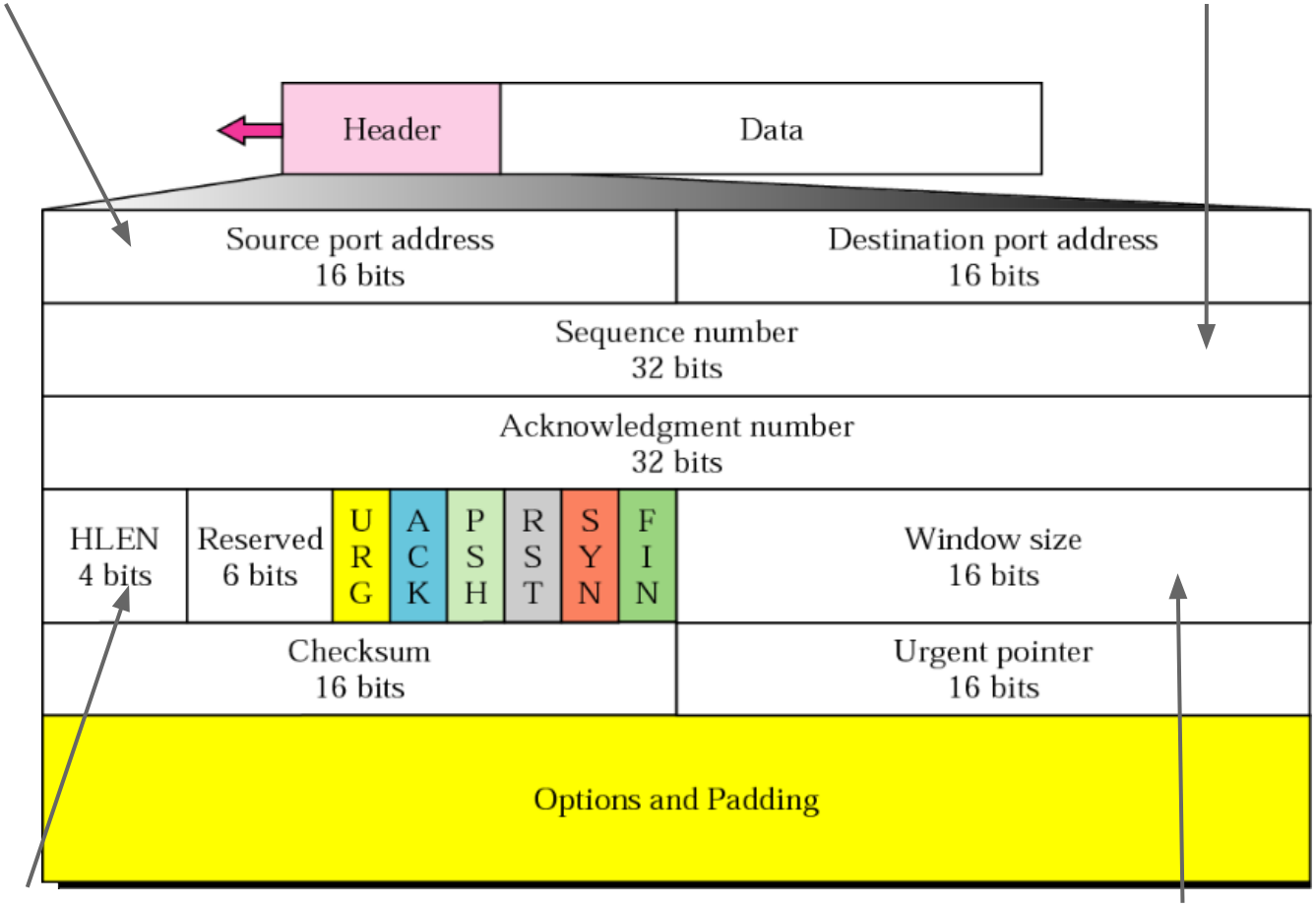


Ejemplo: ¿Cuál serían los números de secuencia y confirmación para un tamaño de segmento de 50 bytes? (Nota, las confirmaciones en TCP no se realizan de forma inmediata)

Formato del Segmento TCP

Puertos, Identifican los extremos de la conexión

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



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

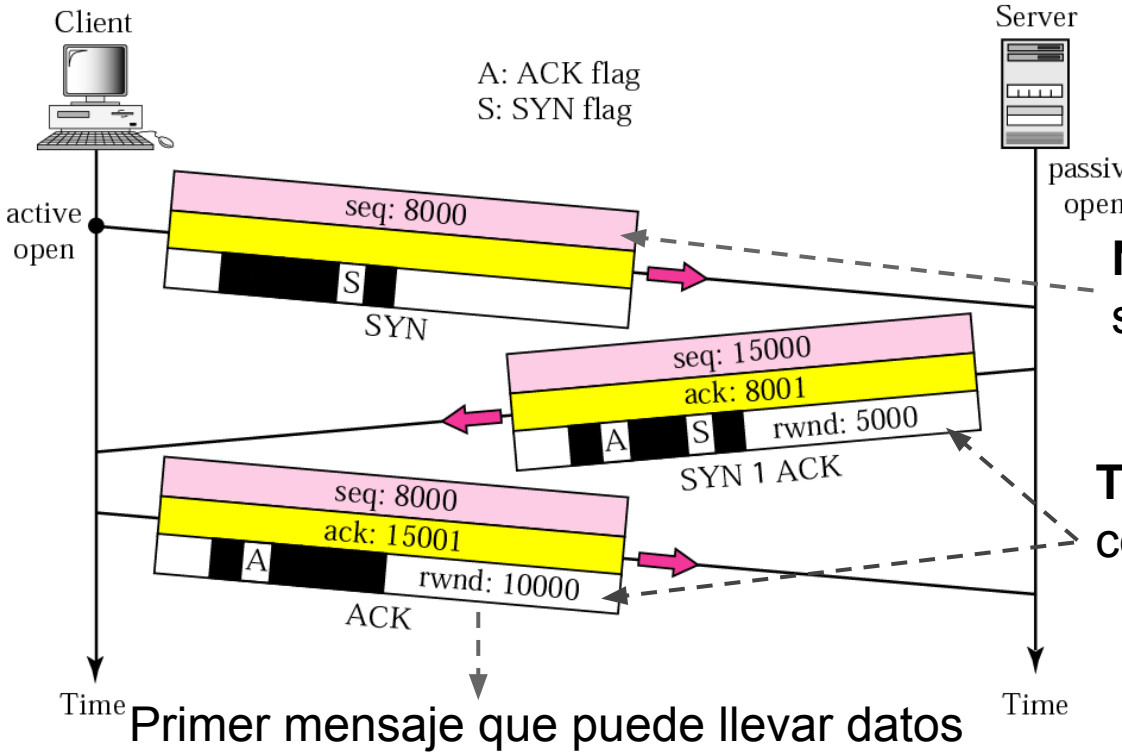
Tamaño de la ventana expresado en bytes (control de flujo)

Formato del Segmento TCP

Flags del campo de control (6 bits)

- **URG** y puntero urgente. (URG=1) El segmento transporta datos urgentes a partir del nº de byte especificado en el campo puntero urgente.
- **ACK**: (ACK=1) El segmento un número de confirmación válido. **Todos los segmentos** de una conexión TCP, excepto el primero, llevan ACK=1.
- **PUSH**: Los datos deben ser enviados inmediatamente a la aplicación (PUSH=1), o pueden almacenarse en el buffer (PUSH=0).
- **RST**: Utilizado para abortar una conexión
- **SYN**: Utilizado en el establecimiento de la conexión y sincronizar los números de secuencia iniciales
- **FIN**:Utilizado en la finalización de la conexión

Fases de la Conexión: Establecimiento (3-way)



Números de secuencia inicial, sincronizan ambos extremos.

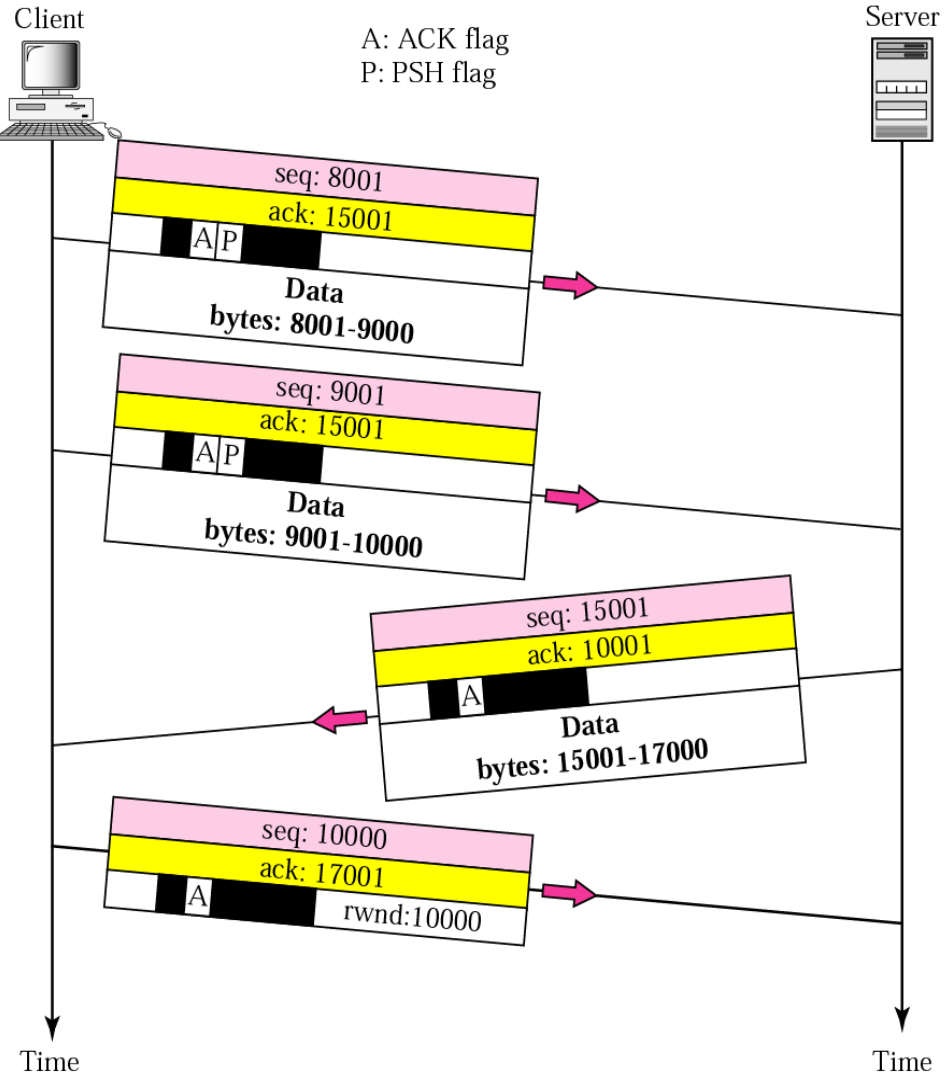
Tamaños de ventana, junto con confirmaciones en ambos sentidos

Primer mensaje que puede llevar datos

SYN Flood

- Vulnerabilidad en el protocolo que consiste en enviar una gran cantidad de segmentos TCP con el flag SYN activado.
- Satura el servidor (DoS) que asigna recursos a cada intento de conexión. Alternativas:
  - Limitar el número de conexiones
  - Aceptar conexiones sólo de IP's confiables
  - Retrasar la asignación de recursos usando cookies

Fases de la Conexión: Transferencia



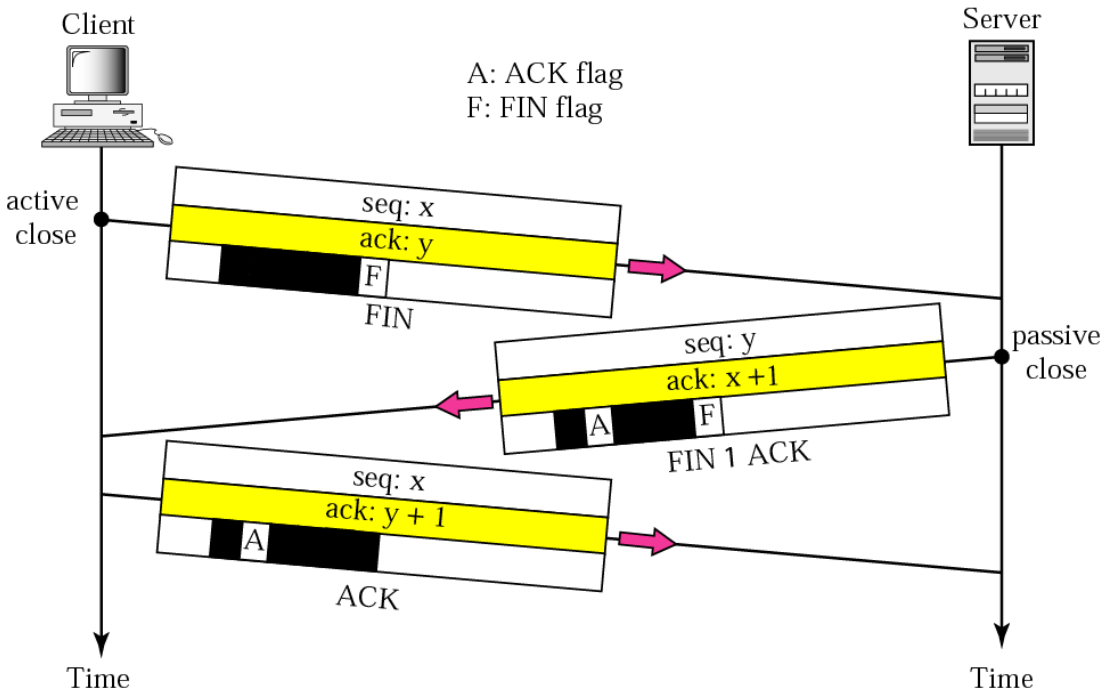
Tamaño del segmento, fijado independientemente por cada extremos.

- **PUSH** (Emisor). Crea un segmento inmediatamente y lo envía (sin esperar MSS).
- **PUSH** (Receptor) Pasa los datos a la aplicación inmediatamente.
- TCP actúa orientado a fragmento y no a byte

URG, desde el primer byte hasta el marcado por el puntero de urgente.

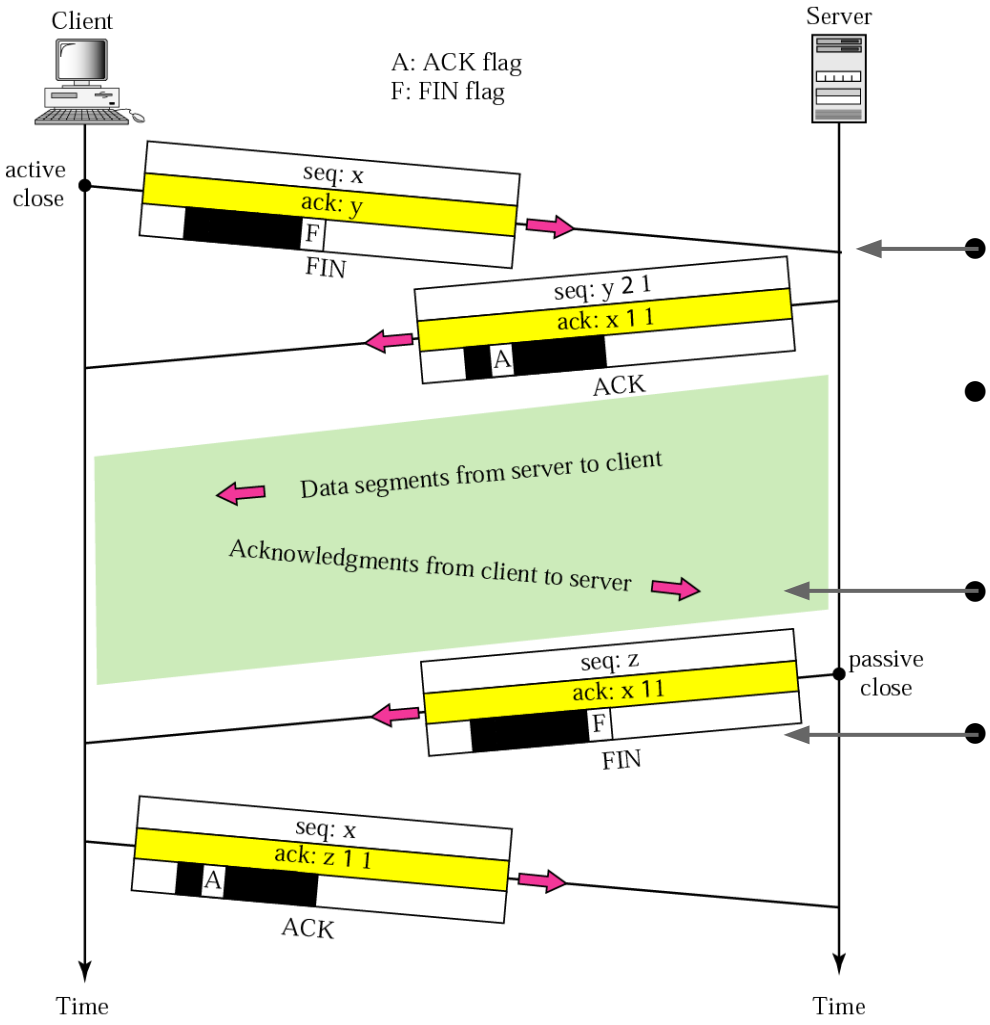
- TCP notifica a la aplicación de los datos urgentes (SIGURG).
- El tratamiento de *urgencia* corresponde a la aplicación no a TCP.

Fases de la Conexión: Finalización (3-way)



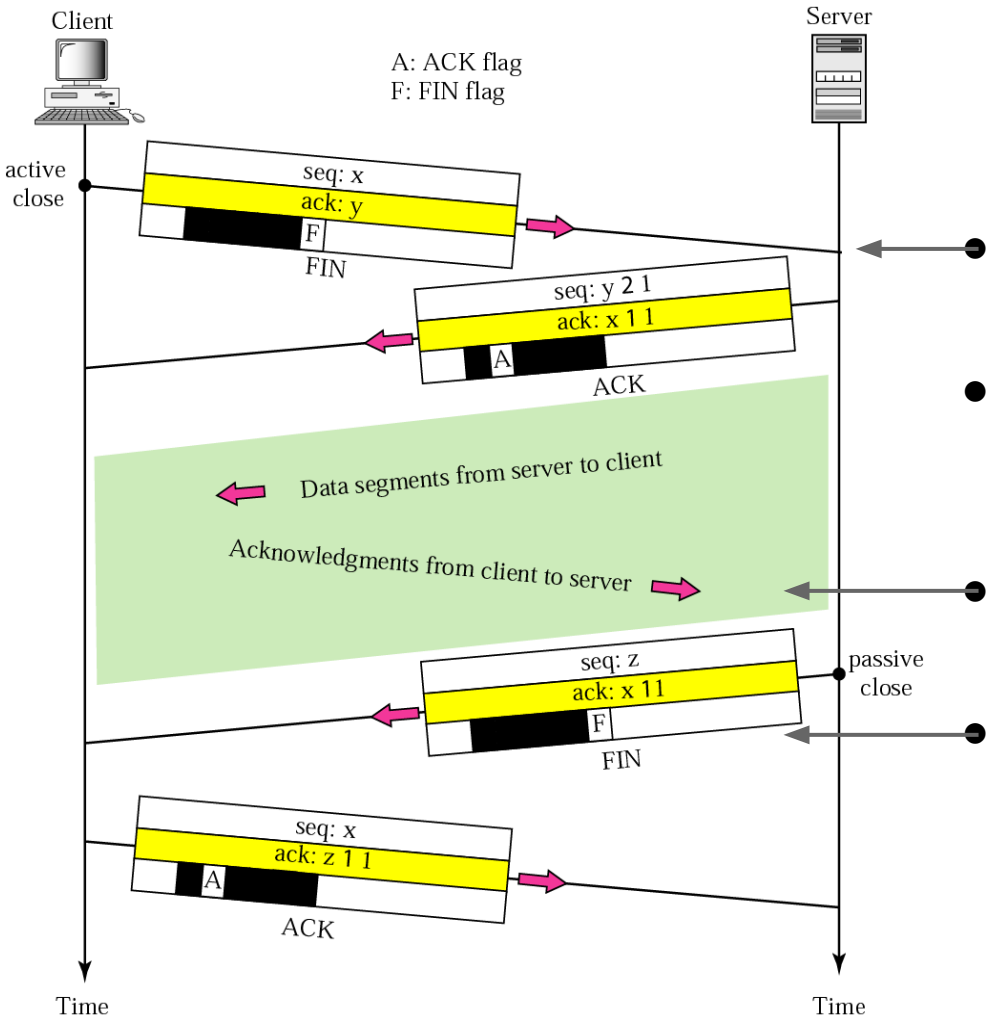
- Ambos extremos dejan de enviar información
- Los mensajes de FIN pueden contener datos. Siempre consume un número de secuencia como mínimo ya que deben ser confirmados
- El último ACK no lleva datos

Fases de la Conexión: Finalización (4-way)



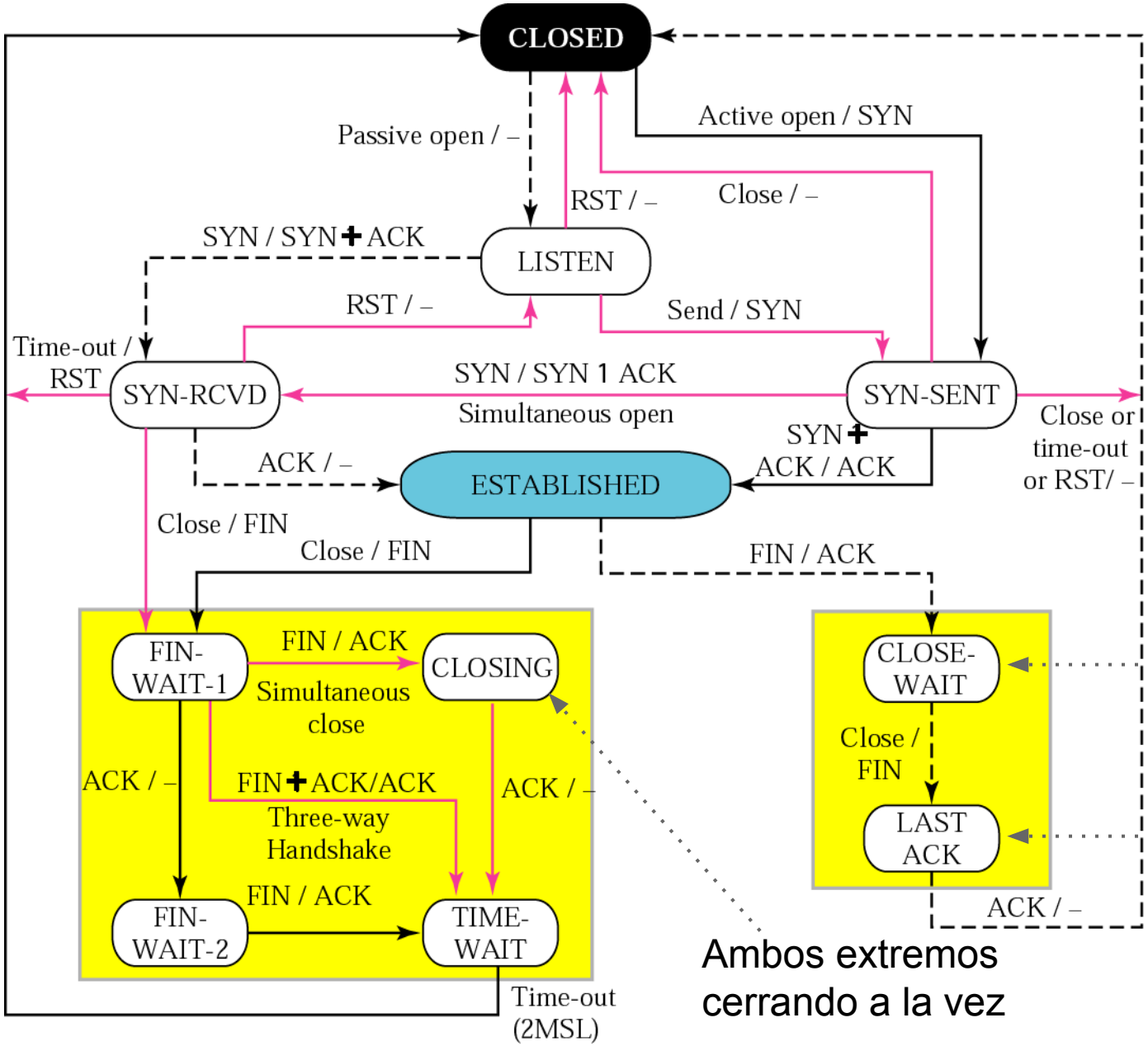
- Termina el envío de datos pero se siguen recibiendo.
- El cliente inicia el cierre, el servidor confirma el FIN pero no inicia la finalización de su extremo.
- El servidor continúa enviando datos al cliente
- Termina la conexión con un mensaje FIN que debe ser confirmado.

Fases de la Conexión: Finalización (4-way)



- Termina el envío de datos pero se siguen recibiendo.
- El cliente inicia el cierre, el servidor confirma el FIN pero no inicia la finalización de su extremo.
- El servidor continúa enviando datos al cliente
- Termina la conexión con un mensaje FIN que debe ser confirmado.

Fases de la Conexión: Máquina de Estados

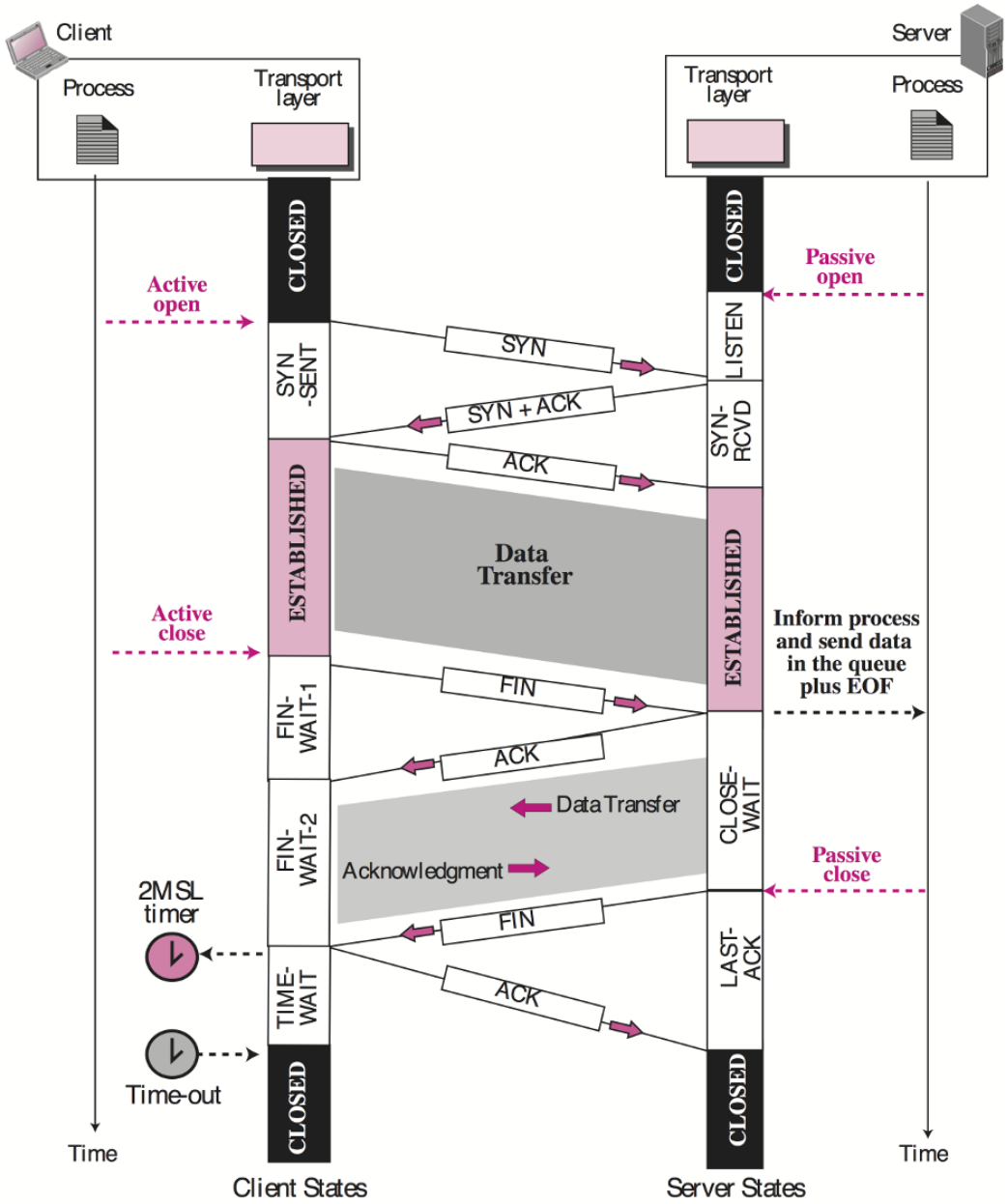


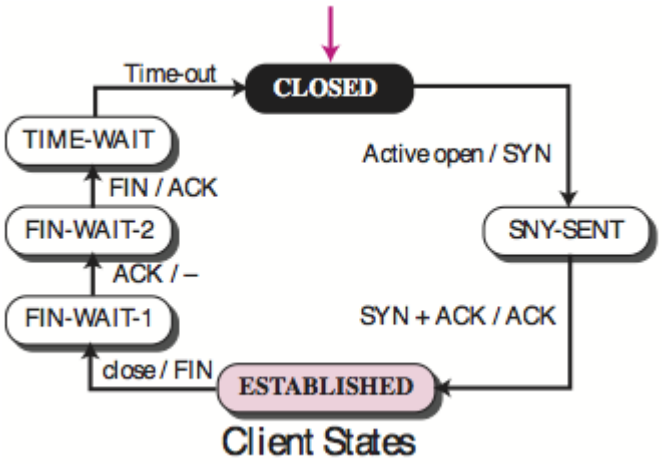
Esperando el cierre de la aplicación

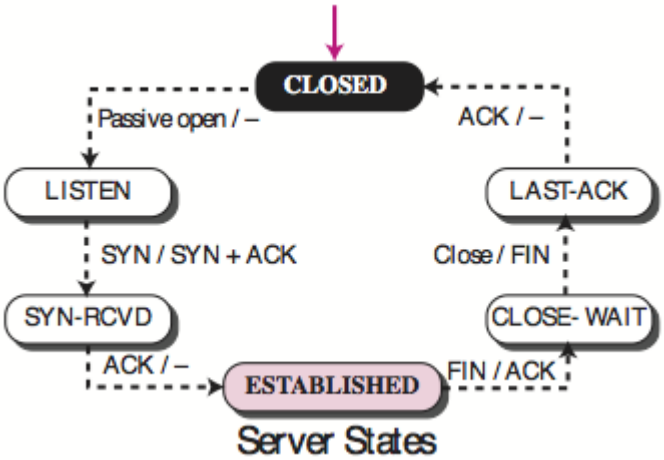
2º FIN enviado

Ambos extremos cerrando a la vez

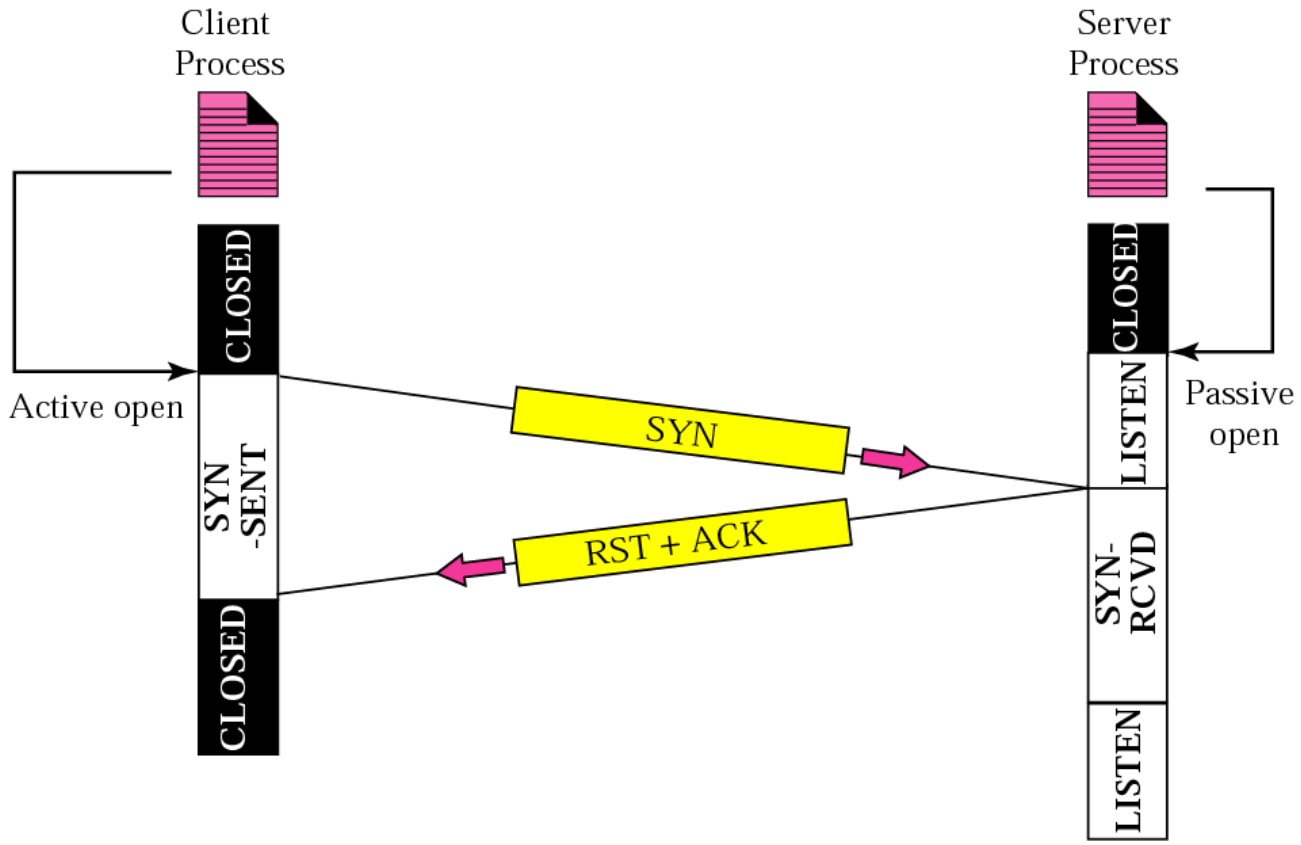
Fases de la Conexión: Máquina de Estados







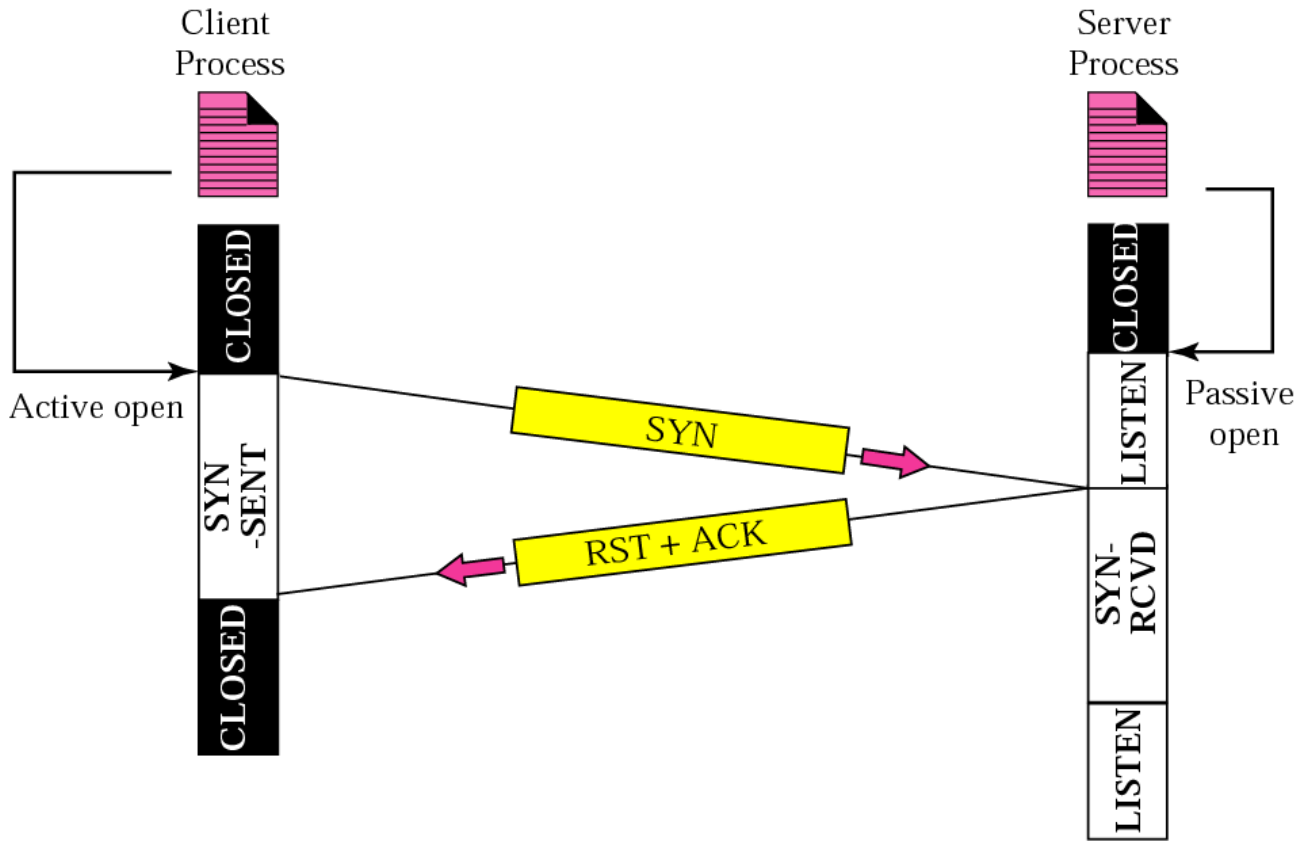
Fases de la Conexión: Máquina de Estados



Ejemplos: Describir la secuencia de estados para el cliente y servidor durante:

- El cierre de tres vías
- El cierre simultáneo

Fases de la Conexión: Máquina de Estados



Ejemplos: Describir la secuencia de estados para el cliente y servidor durante:

- El cierre de tres vías
- El cierre simultáneo

Control de Errores: Confirmaciones

- El control de errores se realiza usando el mecanismo de ventana deslizante que permite gestionar:
  - La recepción de paquetes duplicados
  - La retransmisión de paquetes erróneos o perdidos
  - La recepción de paquetes fuera de línea

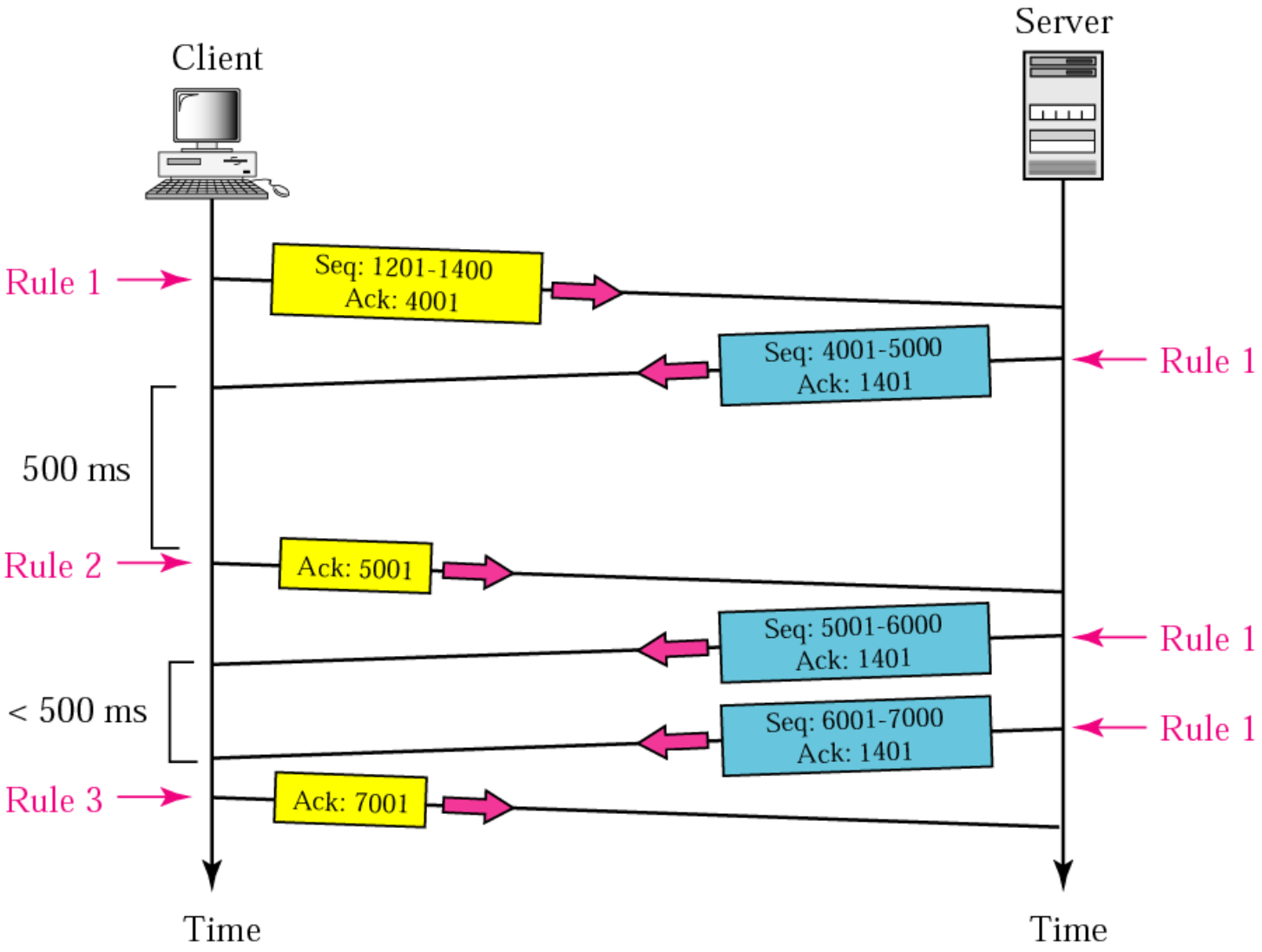
Confirmaciones

- Acumulativas, siguiente byte que espera recibir y solapadas con los envíos (piggyback)
- Las confirmaciones de paquetes *en-orden* se retrasan para solaparla con un envío un máximo de 500ms.
- Sólo se retrasan un máximo de dos confirmaciones en orden.
- Los paquetes fuera de orden se confirman con el siguiente byte que se espera recibir.
- Los paquetes duplicados se confirman para prevenir pérdidas de ACK's.
- (opcional) SACK, confirmaciones selectivas de paquetes fuera de orden
  - No reemplazan los ACK, informativos para el emisor
  - Implementados como opción TCP

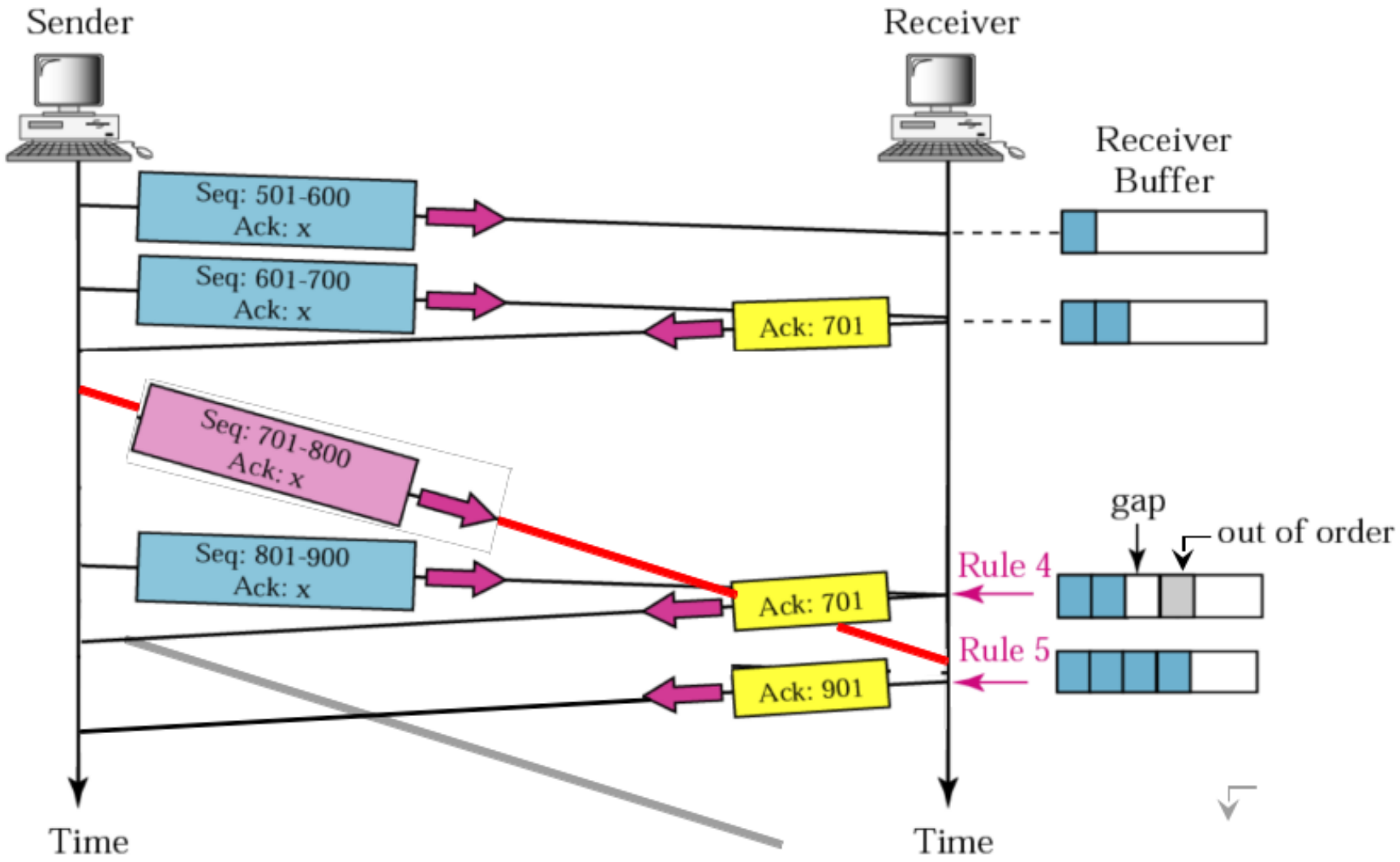
Control de Errores: Retransmisión

- La capacidad para retransmitir un segmento TCP cuando no se recibe o se recibe erróneamente es el núcleo del control de errores.
- TCP dispone de dos mecanismos de retránsmisión:
  - **Temporizador de Retransmisión (RTO, *Retransmission Time-Out)***
    - Cada conexión tiene asociado un único temporizador
    - Cuando el RTO expira se envía el primer segmento sin confirmar de la ventana
    - Existen diversos algoritmos para fijar RTO que es dinámico y debe ser mayor que el RTT (round-trip time)
  - **Retransmisión por recepción de 3 ACKs duplicados**
    - Retransmisión rápida, no requiere que expire el RTO

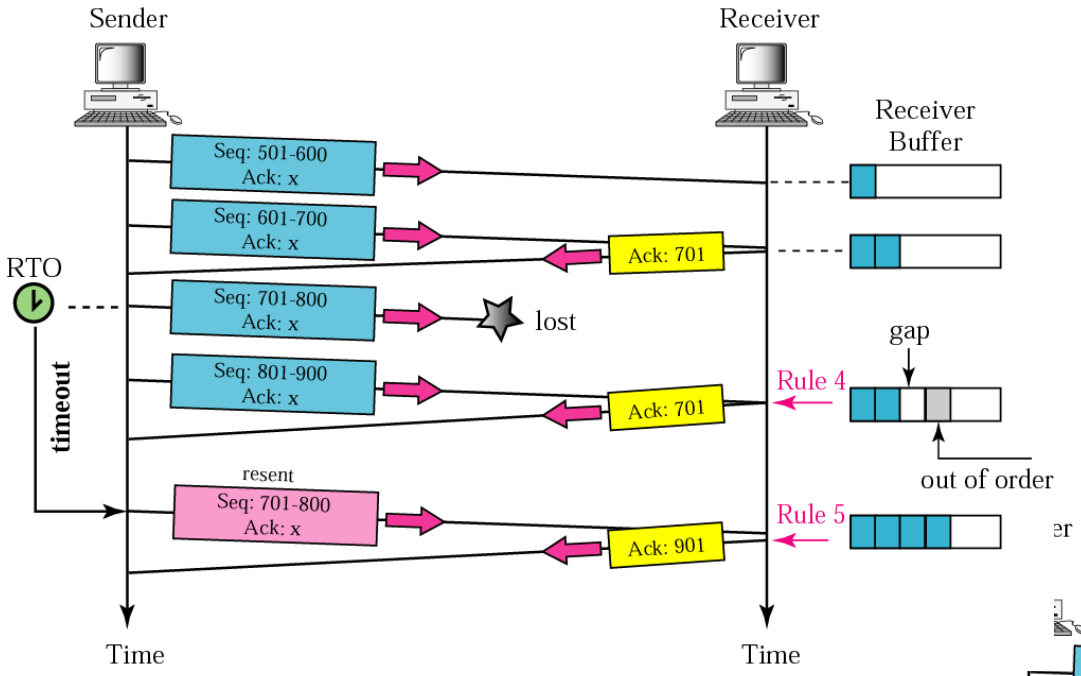
Control de Errores: Transmisión sin Error



Control de Errores: Recepción fuera de orden

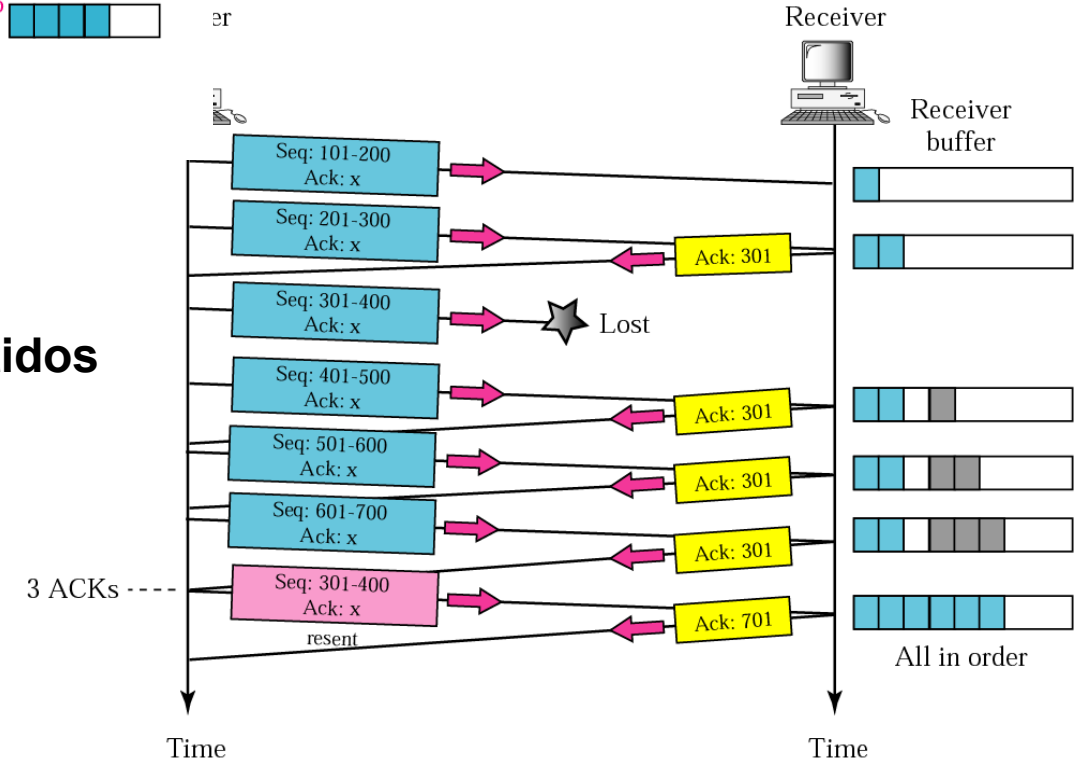


Control de Errores: Perdida de un Segmento

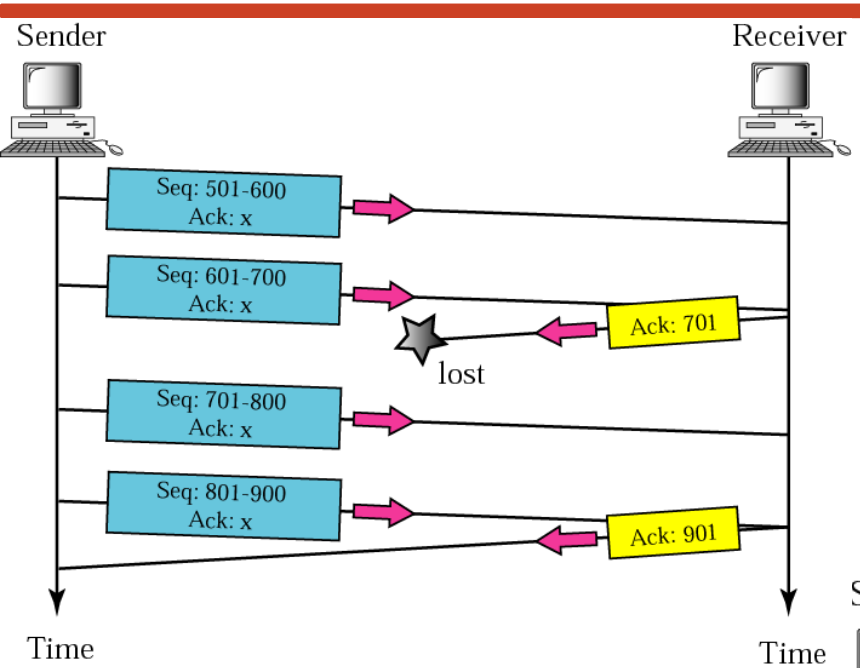


Temporizador RTO expirado

Recepción de 3 ACK's repetidos
(Fast-retransmission)

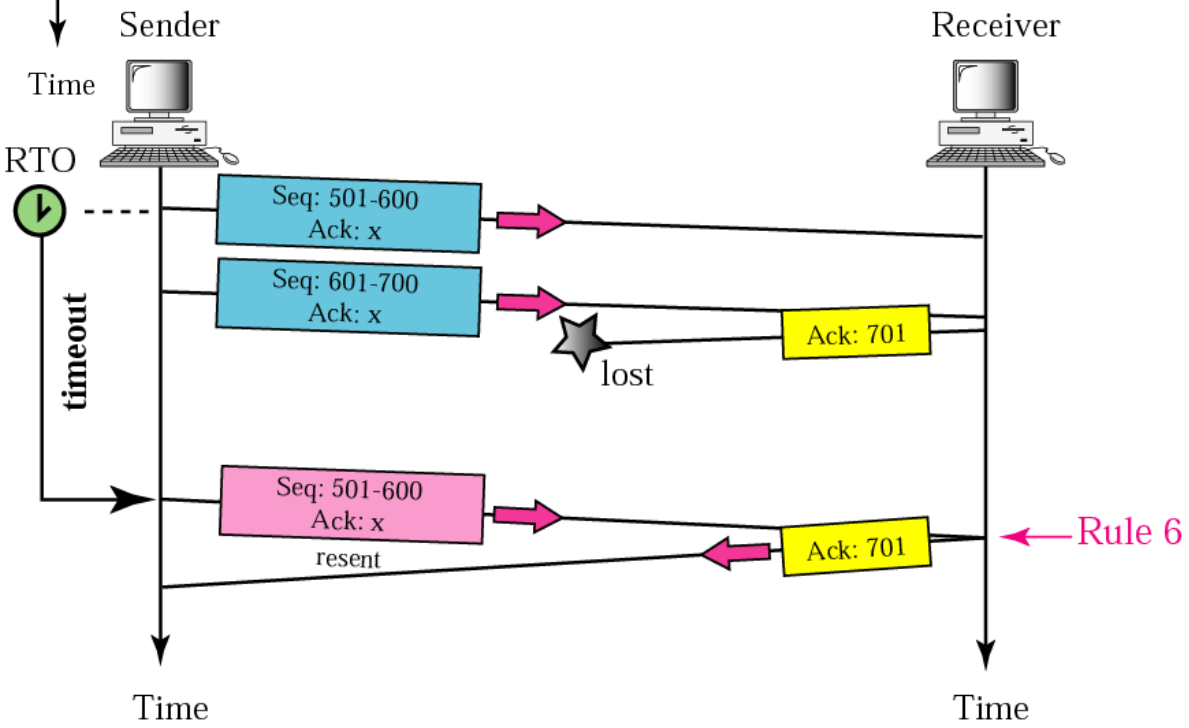


Control de Errores: Perdida de un ACK



Sin expirar el temporizador RTO

Temporizador RTO expirado



Temporizadores TCP

- TCP usa 4 temporizadores para controlar una conexión
  - **RTO**, temporizador de retransmisión
  - **Keepalive**, evita mantener conexiones indefinidamente
    - Segundos que una conexión puede estar en silencio, `tcp_keepalive_time`
    - Expirado el temporizador se envían un máximo de `tcp_keepalive_probes` sondas cada `tcp_keepalive_intvl` segundos.
    - Si no se recibe ningún ACK para las sondas se cierra la conexión.
    - Ej. 2horas, 10 sondas cada 75 segundos
  - **TIMEWAIT**, es útil en dos situaciones:
    - Volver a enviar el último ACK durante el cierre activo (se recibe FIN)
    - Previene la colisión de números de secuencia de dos conexiones distintas (puerto y nº de secuencia no se pueden reutilizar)
    - 2 * MSL (maximum segment life-time). Eg. 30, 60, 120 segundos
  - **Temporizador de persistencia**:
    - Asociado a la recepción de un tamaño de ventana 0
    - Recupera la pérdida de un ACK posterior con el nuevo tamaño
    - Se envía una sonda que fuerza el envío de un ACK
    - Ej. 60 segundos

Temporizador de Retransmisión

- 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 debe adaptarse a esta situación
- Las principales técnicas utilizadas para fijar los temporizadores de retransmisión son las siguientes:
  - Método de la media ponderada (algoritmo de Jacobson)
  - Método de la varianza (algoritmo de Jacobson/Karels)
  - Algoritmo de Karn

Temporizador de Retransmisión

Tiempo de ida y vuelta medido (RTT_M)

- Cuando se envía segmento, se mide el tiempo transcurrido desde que se envía el segmento hasta que se recibe el ACK, denominado RTT_M (Measured Round-Trip Time)
- Sólo hay un temporizador RTTM
- El valor del RTTM puede experimentar grandes fluctuaciones

Tiempo de ida y vuelta suavizado (RTT_S)

- Evitar las fluctuaciones del RTT
- RTT_S (Smoothed Round-Trip Time), es la media ponderada entre el RTT_M y el último RTTs calculado:

  Medida 1: $RTT_S[1] = RTT_M[1]$

  Medida k: $RTT_S[k] = (1 - \alpha) \times RTT_S[k-1] + \alpha \times RTT_M[k]$, $\alpha < 1$ (Ej. $\alpha = ⅛$)

Desviación del RTT (RTT_D)

- Considera la variación del tiempo de ida y vuelta

  Medida 1: $RTT_D[1] = RTT_M[1]/2$

  Medida k: $RTT_D[k] = (1-\beta) \times RTT_D[k-1] + \beta \times |RTT_S[k] - RTT_M[k]|$ $\beta < 1$ (Ej. $\beta = 1/4$)

Temporizador de Retransmisión

Temporizador de Retransmisión (Algoritmo de Jacobson)

- Considera únicamente el RTT_S
- El RTO se calcula después de cada medida del RTO como

  $RTO = \gamma \times RTT_S$ (Ej. $\gamma=2$, el doble del RTT estimado)

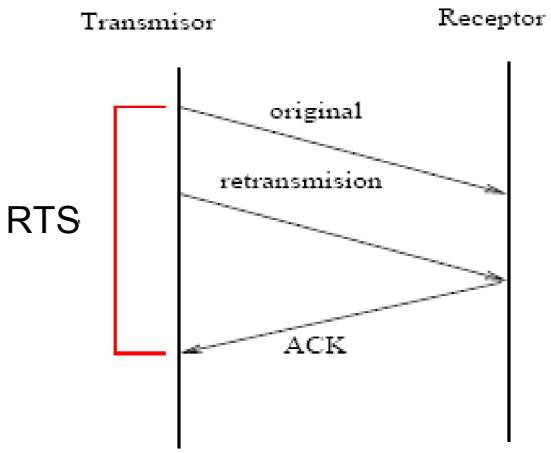
Temporizador de Retransmisión (Algoritmo de Jacobson/Karels)

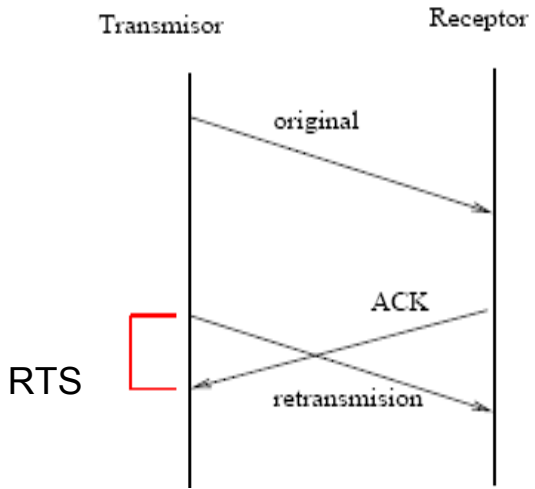
- Combina el RTT_S y el RTT_D

  $RTO = RTT_S + 4 \times RTT_D$

Algoritmo de Karn

- Previene la ambigüedad en el cálculo del RTT_M cuando hay retransmisiones
- No usa el RTT_M en estos casos
- En caso de retransmisión (**RTO exponential backoff**) RTO = 2×RTO

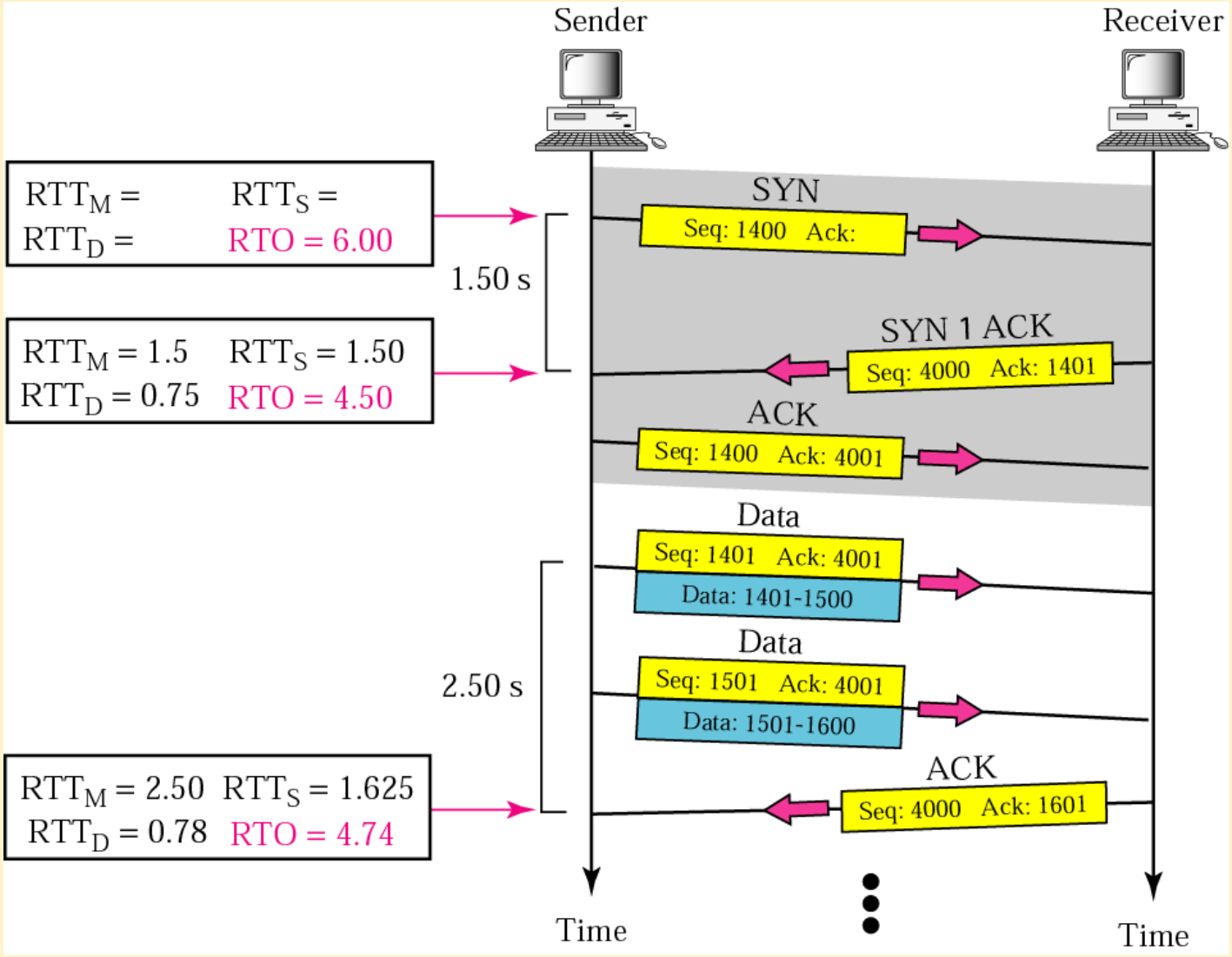




Temporizador de Retransmisión

Ejemplo: Calcular los valores de los temporizadores:

- $\alpha=1/8$, $\beta=1/4$



Temporizador de Retransmisión

Ejemplo: Calcular los valores de los temporizadores:

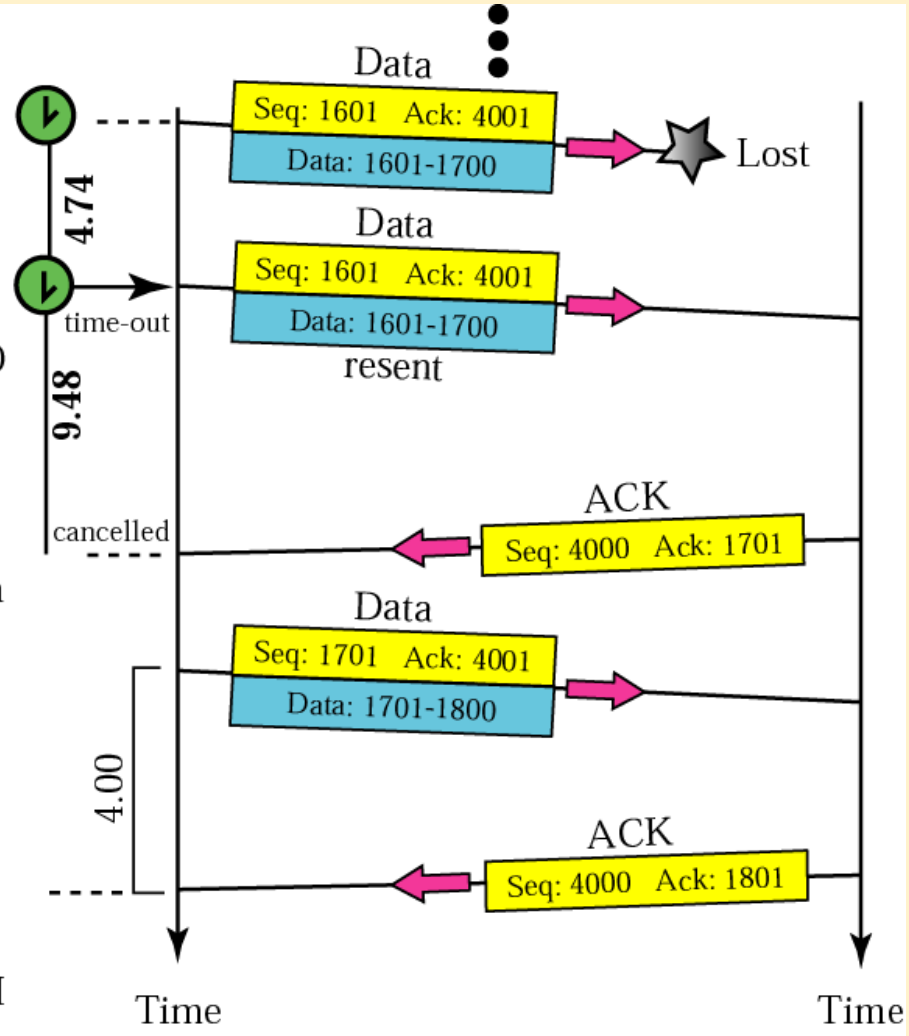
- $\alpha=1/8$, $\beta=1/4$

$RTT_M = 2.50$ $RTT_S = 1.625$
$RTT_D = 0.78$ RTO = 4.74
Values from previous example

RTO = 2 x 4.74 = 9.48
Exponential Backoff of RTO

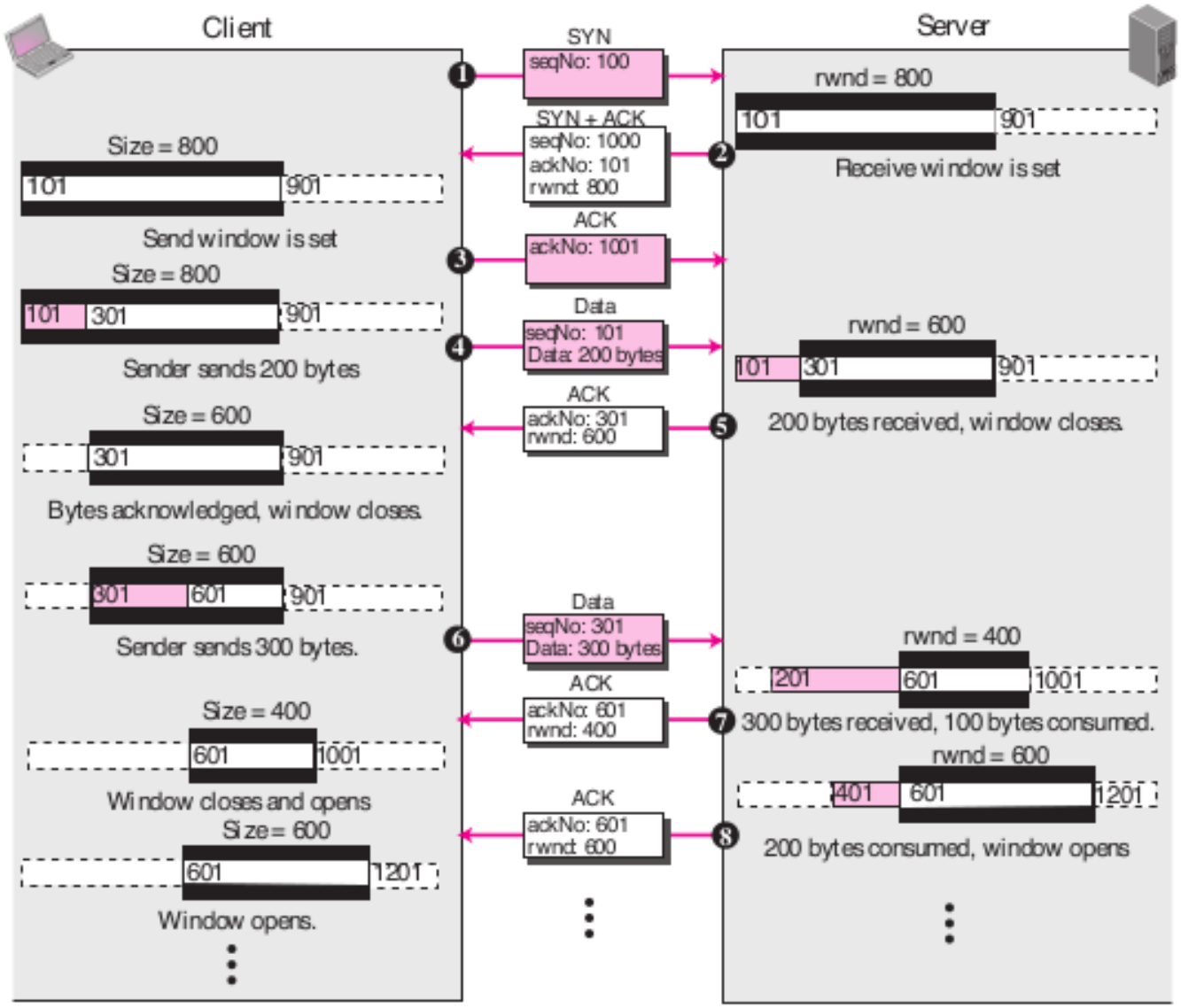
RTO = 2 x 4.74 = 9.48
No change, Karn's algorithm

$RTT_M = 4.00$ $RTT_S = 1.92$
$RTT_D = 1.105$ RTO = 6.34
New values based on new RTT_M



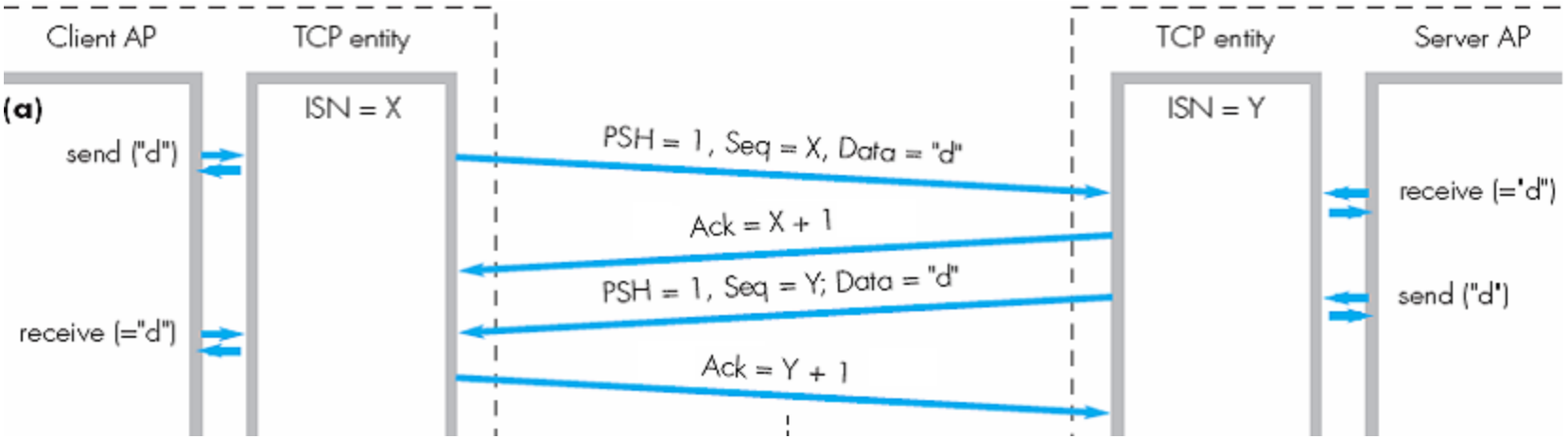
Control de Flujo

- Controla la tasa de envío de datos para evitar la sobrecarga del receptor
- El control de flujo en se realiza mediante la ventana de recepción, anunciada en cada ACK.



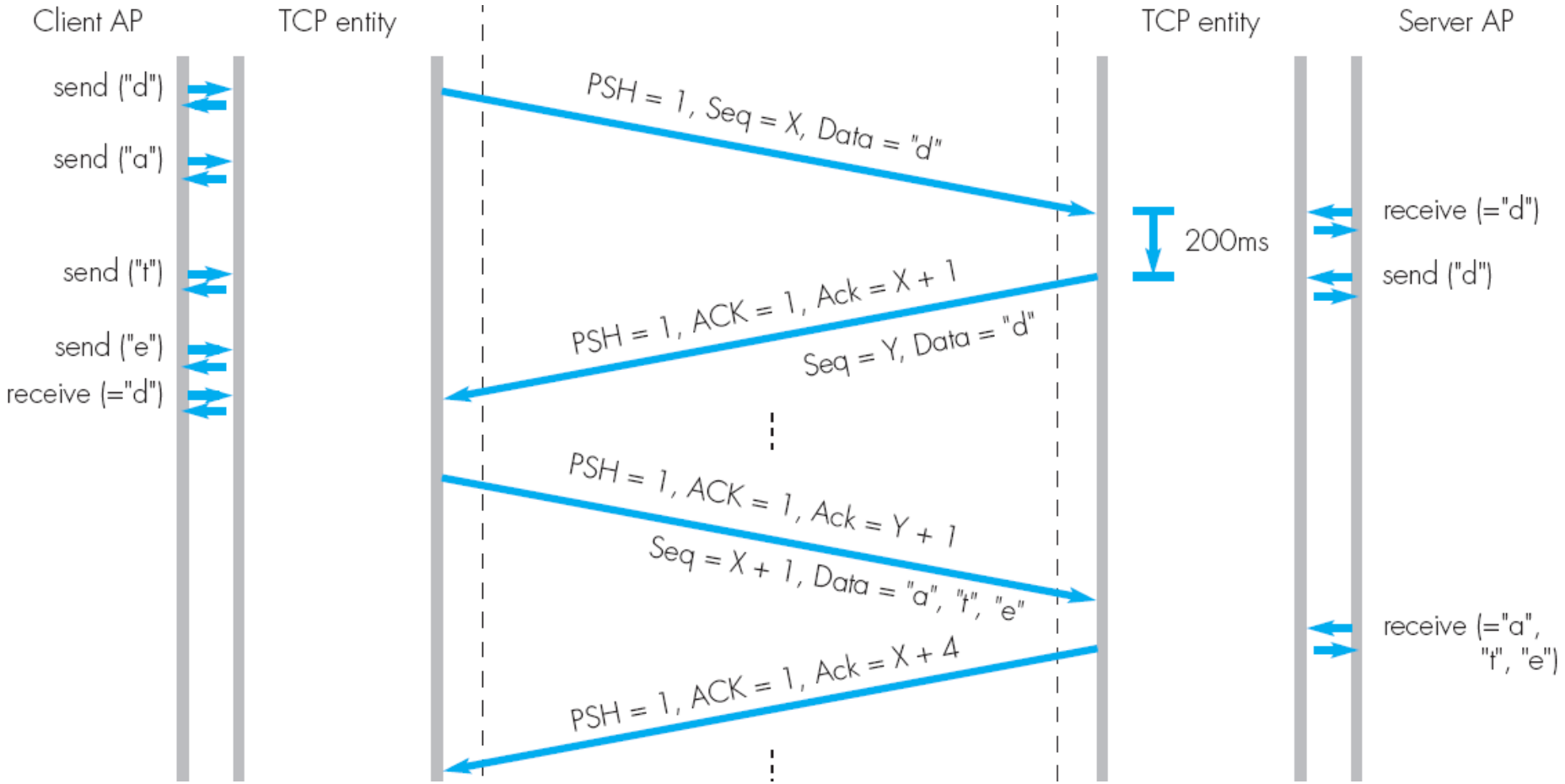
Control de Flujo: Síndrome de la ventana trivial

- El síndrome de la ventana trivial (*silly window*) se produce cuando:
  - La aplicación emisora genera datos a un ritmo muy lento (ej. byte a byte)
  - La aplicación receptora consume datos a un ritmo muy lento
- **Ventana trivial en el emisor** (ej. Aplicaciones interactivas)
  - Cada carácter necesita 4 mensajes TCP/IP (40bytes de cabeceras)
  - Un carácter (1 bytes) usa más de 160 bytes
- **Algoritmo de Nagle**
  - El emisor envía el primer mensaje (aunque sea un sólo byte)
  - Los siguientes mensajes se retrasan hasta que:
    - se recibe un ACK del receptor
    - se acumulan MSS bytes de la aplicación
    - expira un temporizador



Control de Flujo: Síndrome de la ventana trivial

Ejemplo: Transmisión con el algoritmo de Nagle

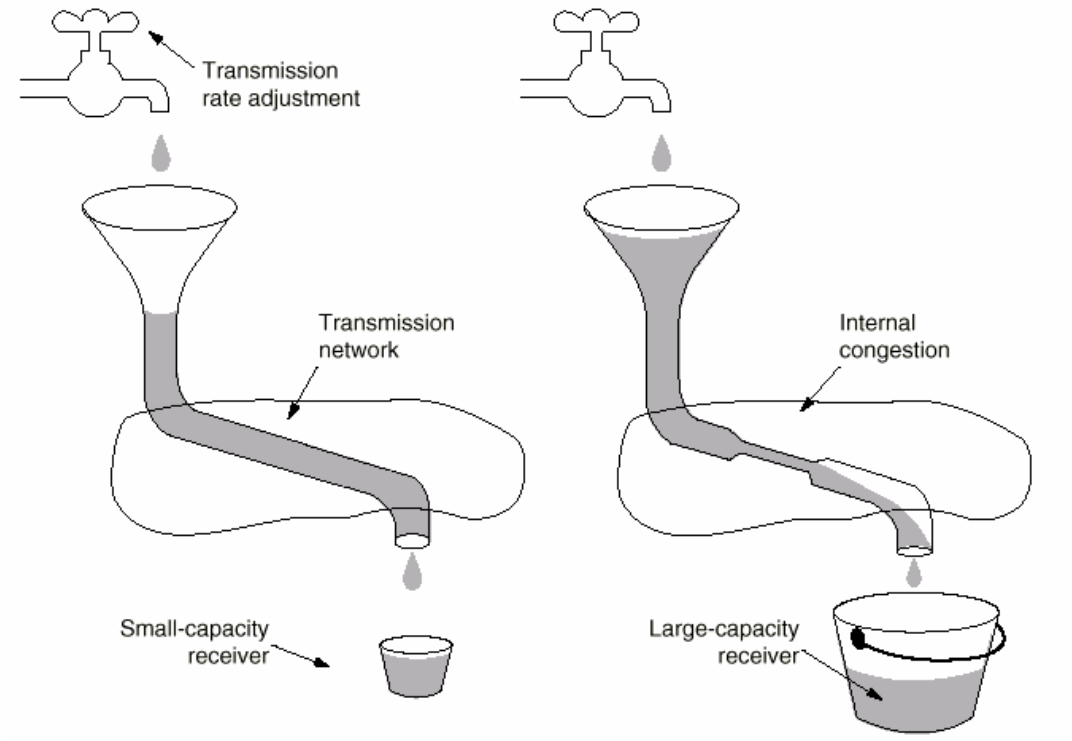


Control de Flujo: Síndrome de la ventana trivial

- **Ventana trivial en el receptor**
  - La aplicación consume los datos a un ritmo lento
  - Se anuncian ventanas de tamaño reducido, produciendo el efecto anterior
- **Algoritmo de Clark**
  - Anunciar un tamaño de ventana 0 hasta que:
    - Se puede recibir un segmento completo (MSS)
    - Se ha liberado la mitad del buffer de recepción
- Retrasar los ACKs
    - Para el desplazamiento de la ventana del emisor
    - Reduce el tráfico (número de ACKs) pero puede provocar retransmisiones innecesarias
    - TCP establece que no deben retrasarse más de 500ms

Control de la Congestión

- Cuando se pierde paquetes en Internet, la mayoría de las veces se debe a un problema de congestión en algún punto de la red:
  - El router no puede procesar y reexpedir paquetes al ritmo al que los recibe
  - Cuando el router se satura, empieza a descartar paquetes (incluidas las confirmaciones)
- El control de la congestión y el flujo son dos mecanismos diferentes



Control de la Congestión

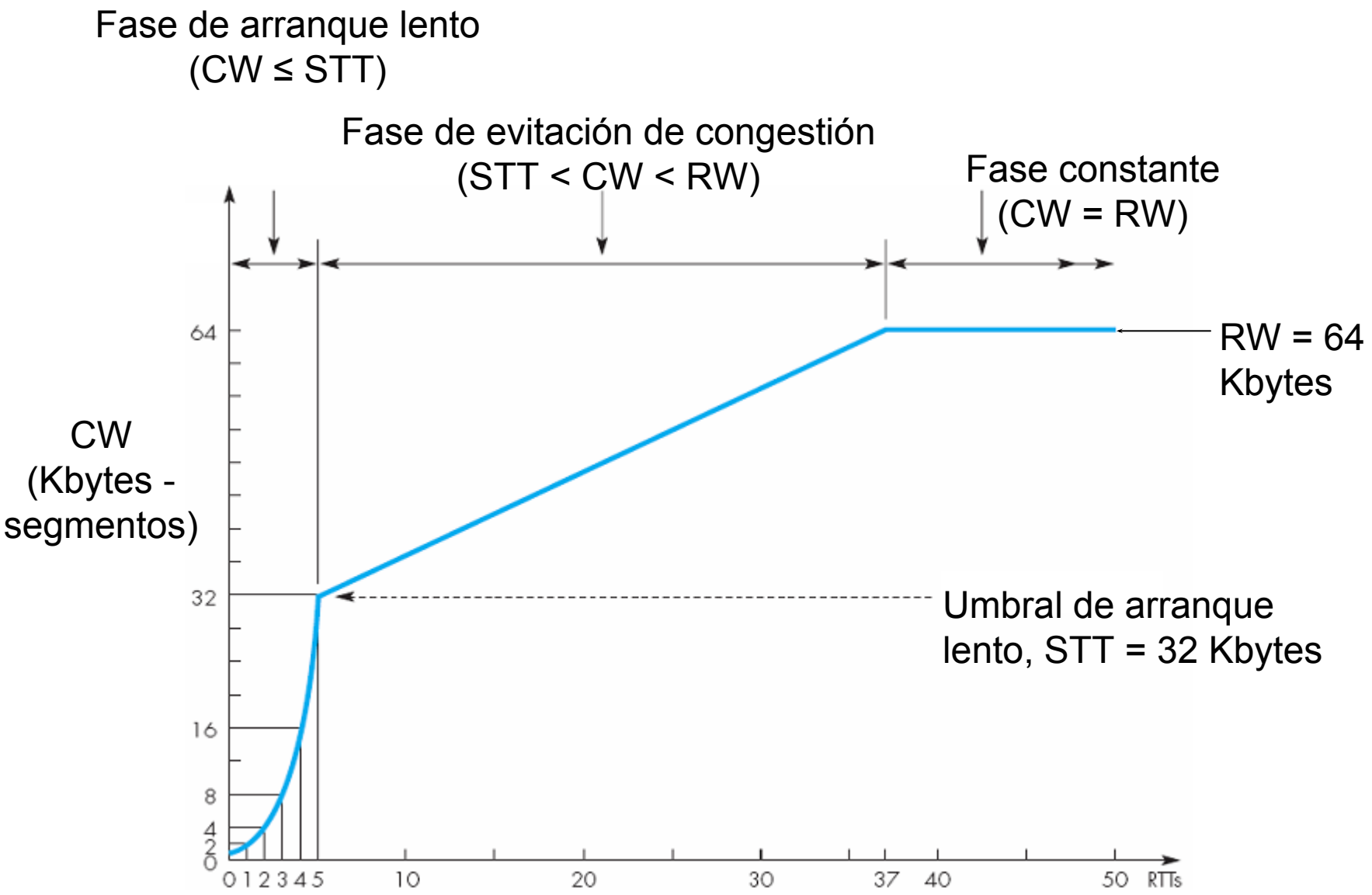
- El emisor utiliza el ritmo de llegada de confirmaciones para regular el ritmo de envío de segmentos de datos
- Esto se implementa mediante la **ventana de congestión (CW)**
  - La ventana de congestión es complementaria a la ventana de recepción (RW) usada para el control de flujo
  - En una situación de no congestión (sin pérdida o retraso de segmentos) la ventana de congestión alcanza el mismo tamaño que la ventana de recepción (CW=RW)
  - Cuando se produce una situación de congestión el tamaño de CW se va reduciendo progresivamente
  - Cuando la situación de congestión desaparece, el tamaño de CW se va aumentando progresivamente
  - El número máximo de bytes que puede enviar el emisor (AW, Allowed Window) es el mínimo de ambos tamaños de ventana:

$$AW = \min \{ RW, CW \}$$

Control de la Congestión

- La red está sin congestión cuando no se pierden o retrasan segmentos
- La transmisión comienza con un tamaño de ventana de congestión CW = 1
  - El emisor envía un único segmento de tamaño máximo igual a MSS
- A continuación, la CW va aumentando, pasando por tres fases distintas:
  - **Fase de arranque lento** (*slow start*)
    - La CW se incrementa en uno por cada segmento enviado y confirmado
    - Esto provoca un crecimiento exponencial (CW = 1, 2, 4, 8, 16, 32, …)
    - Esta fase termina cuando el tamaño de CW alcanza un cierto umbral, denominado umbral de arranque lento (STT, *Slow Start Threshold*)
    - Inicialmente, el valor del STT suele ser de 64 Kbytes
  - **Fase de evitación de congestión** (*congestion avoidance*)
    - A partir del STT, la CW se incrementa en 1 cada vez que se envía y se confirma una ventana completa (es decir, CW segmentos)
    - Esto provoca un crecimiento lineal
    - Esta fase termina cuando la CW alcanza el tamaño de la ventana de recepción (RW)
  - **Fase constante**
    - En esta fase, la CW se mantiene a un valor constante (CW = RW)

Control de la Congestión

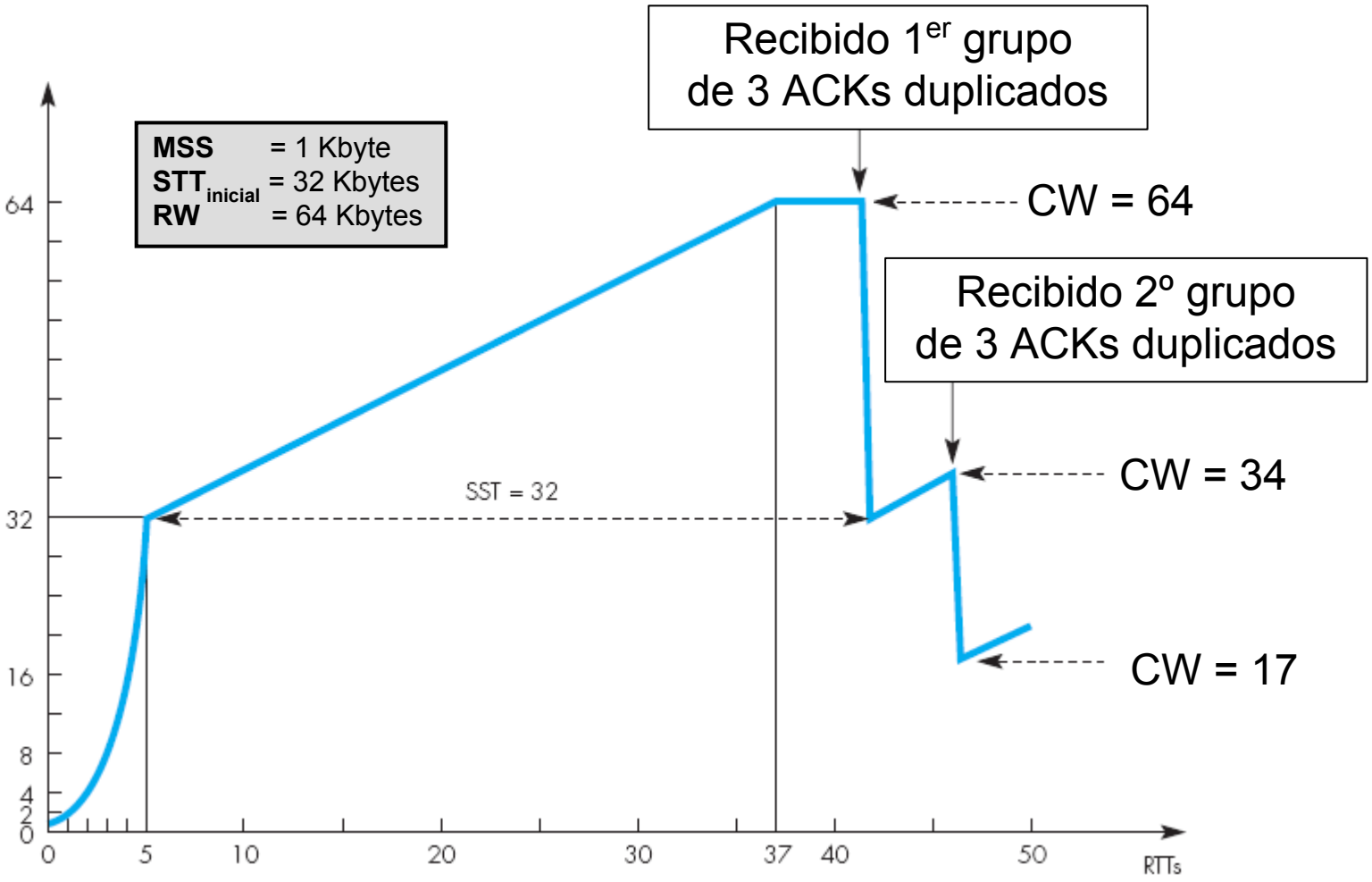


Control de la Congestión

- La situación de congestión en la red se detecta indirectamente
- **Recepción de 3 ACKs duplicados**
  - Nivel de congestión leve, sigue habiendo tráfico en la red (llegan las confirmaciones)
  - Se activa el método de recuperación rápida (*fast recovery*):
    - Dividir el valor de CW a la mitad
    - Ejecutar el método de evitación de colisiones a partir de ese valor de CW
- **Expiración del temporizador de retransmisión** (timeout)
  - Nivel de congestión elevado, se interpreta que el tráfico en la red está interrumpido (no llegan confirmaciones)
  - En este caso se realizan las siguientes acciones:
    - Inicializar el tamaño de la ventana de congestión a CW = 1
    - Reducir el umbral de arranque lento (STT), fijándolo a la mitad del valor que tenía la CW antes de producirse el timeout
    - Ejecutar el método de arranque lento a partir de CW = 1

Control de la Congestión

- **Recepción de 3 ACKs duplicados**



Control de la Congestión

- **Expiración de los temporizadores de retransmisión**