Xarxes de Computadors

Tema 3 – Protocolos UDP y TCP

Temario

- ~~1) Introducción~~
- ~~2) Redes IP~~
- **3) Protocolos UDP y TCP**
- 4) Redes de área local (LAN)
- 5) Protocolos del nivel aplicación

Tema 3 – Protocolos UDP y TCP

- a) Introducción
- b) El protocolo UDP
- c) El protocolo TCP
  - Arquitectura
  - El MSS
  - Números de secuencia
  - Establecimiento y terminación de una conexión TCP
  - Funcionamiento durante la transmisión
    - Control de flujo
    - Control de congestión
  - Cabecera TCP

Tema 3 – Protocolos UDP y TCP

- **a) Introducción**
- b) El protocolo UDP
- c) El protocolo TCP
  - Arquitectura
  - El MSS
  - Números de secuencia
  - Establecimiento y terminación de una conexión TCP
  - Funcionamiento durante la transmisión
    - Control de flujo
    - Control de congestión
  - Cabecera TCP

Tema 3 – Introducción

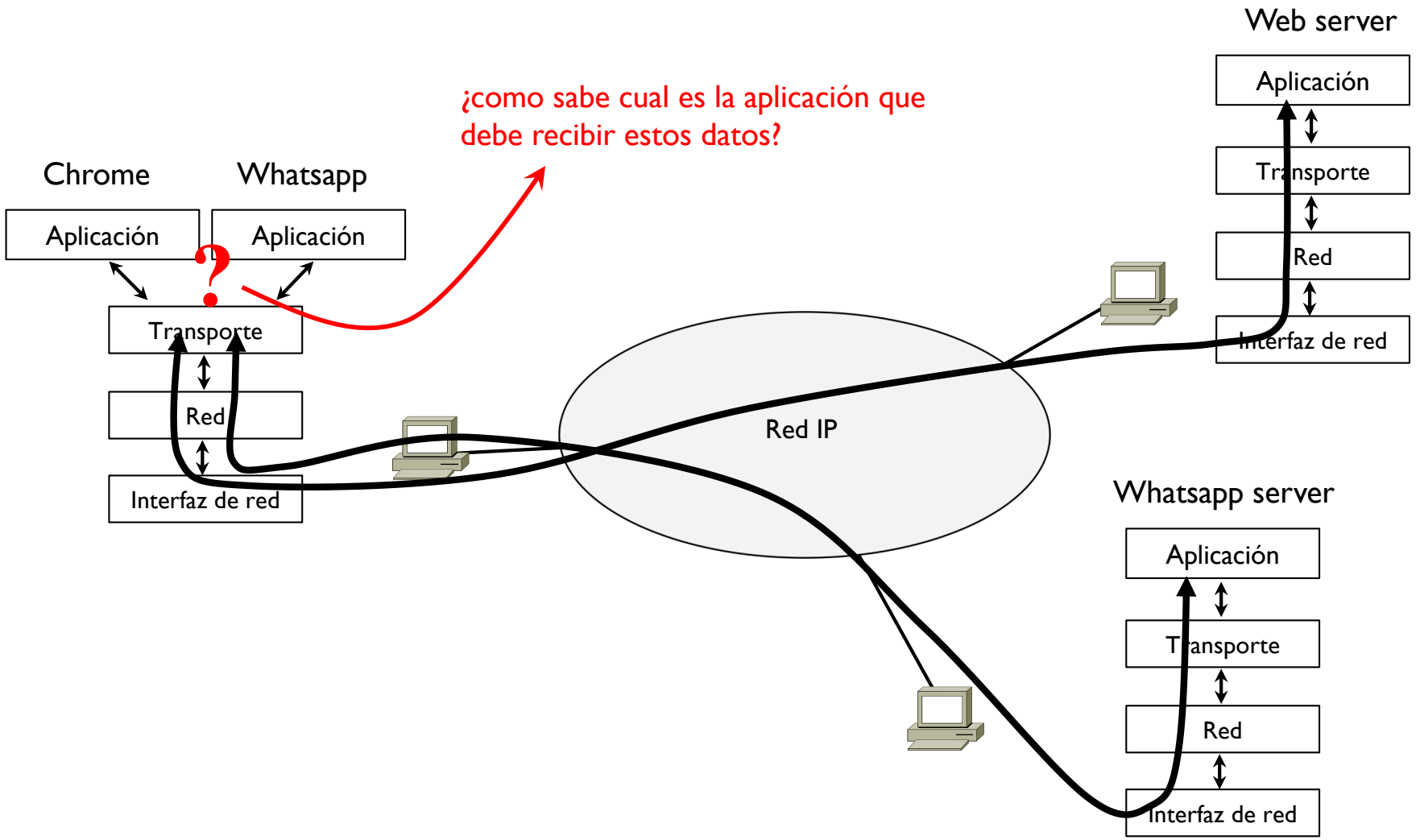
Protocolo no fiable UDP
Protocolo fiable TCP

IP proporciona un método simple de entrega de datagramas al máximo de sus posibilidades

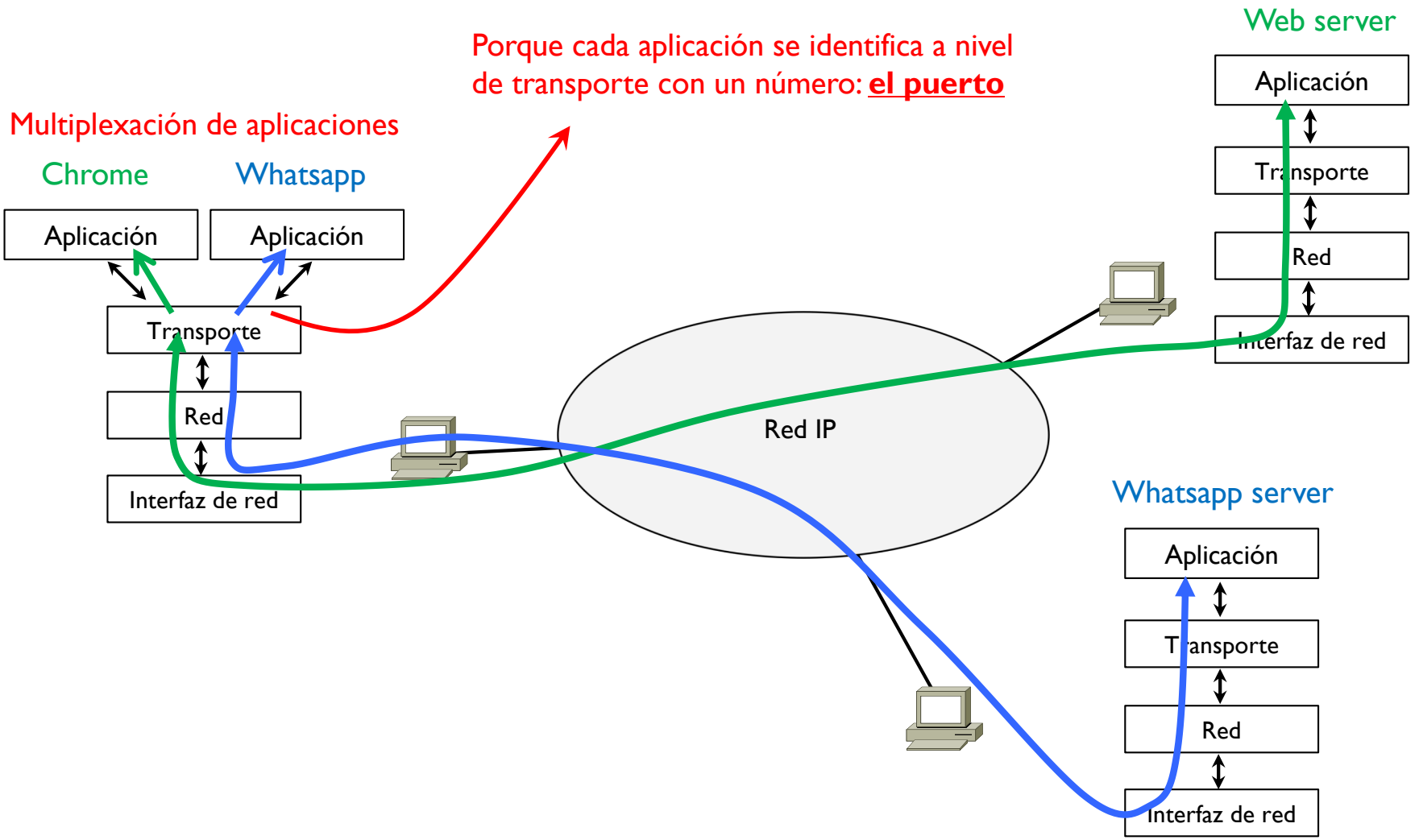
- Best-effort
- No fiable, no hay recuperación de datagramas perdidos
- No orientado a la conexión, se envían los datagramas a los destinos sin previamente haber "avisado"

- Objetivo de la capa de transporte
  - Se ocupa de la interacción entre las aplicaciones y las redes IP
  - Proporciona un método de multiplexación de aplicaciones entre host origen y destino (extremo a extremo) para redes IP
  - Opcionalmente proporciona fiabilidad a la transmisión: recuperación en caso de pérdida

Tema 3 – Introducción



Tema 3 – Introducción



Tema 3 – Introducción

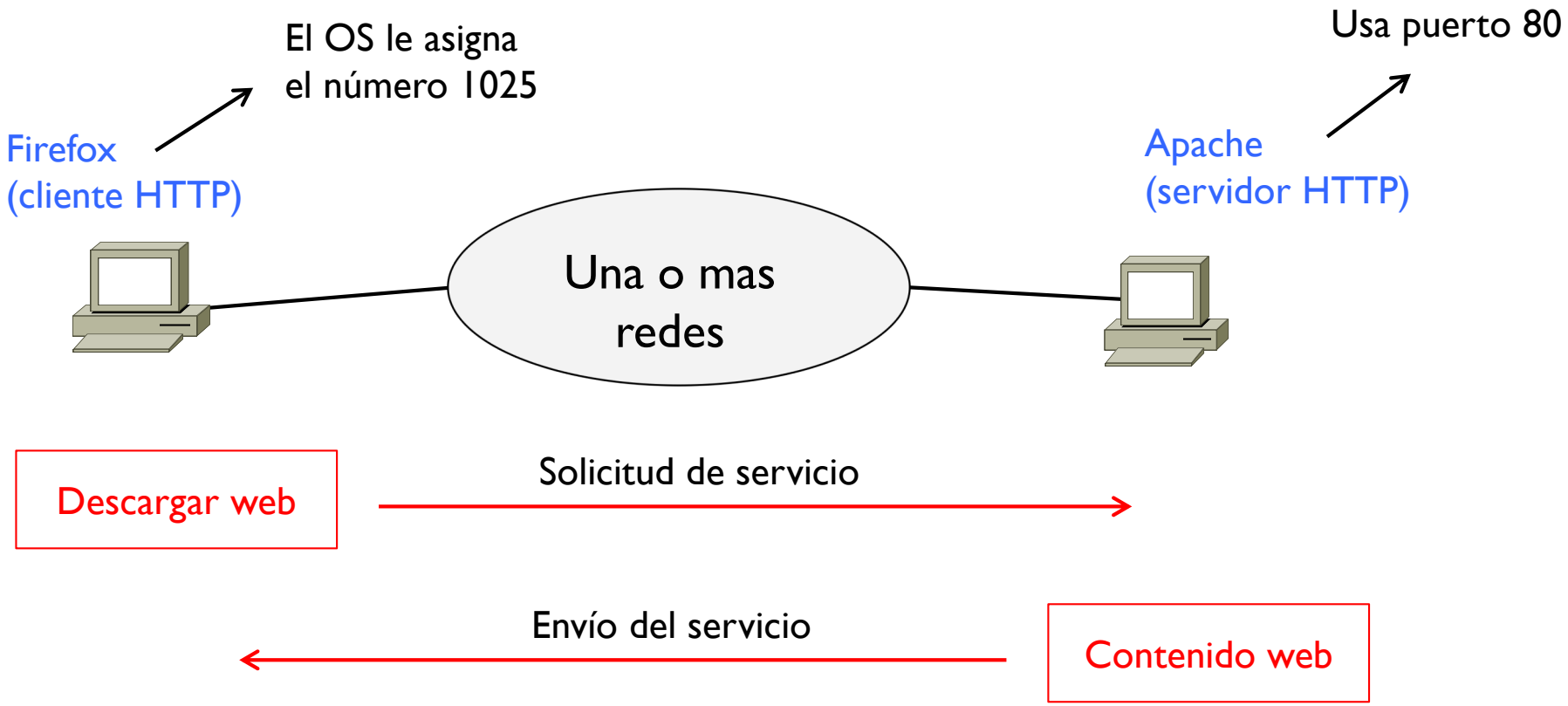
- Es un número de 16 bits
- Se representa como un único número decimal

0 - 65535

- Identifica la aplicación de red
- Los primeros 1024 números (de 0 a 1023) están asignados a aplicaciones conocidas del TCP/IP
  - Suelen ser servidores que proporcionan un servicio conocido a la red, hosts y/o routers
  - HTTP 80 SMTP 25 DHCP 67 y 68 SSH 22
  - FTP 20 y 21 DNS 53 RIP 520 Telnet 23
- Los otros números (de 1024 a 65535) los asigna generalmente en automático el Sistema Operativo y se conocen como números efímeros
  - Suelen ser números asignados a clientes de servidores conocidos
- En la cabecera TCP/UDP hay 2 puertos
  - Uno identifica la aplicación origen
  - Uno identifica la aplicación destino

Tema 3 – Introducción

- Suelen basarse en la arquitectura cliente - servidor



Tema 3 – Introducción

- TCP/IP tiene dos protocolos para la capa de transporte

- UDP (RFC 768)
  - Solo proporciona multiplexación de aplicaciones

- TCP (RFC 793)
  - Proporciona multiplexación de aplicaciones
  - Fiable
    - si se pierde algo, proporciona un mecanismo que recupera y retransmite
  - Orientado a la conexión
    - los dos extremos deben ponerse de acuerdo estableciendo una conexión entre ellos antes de poder transmitirse datos

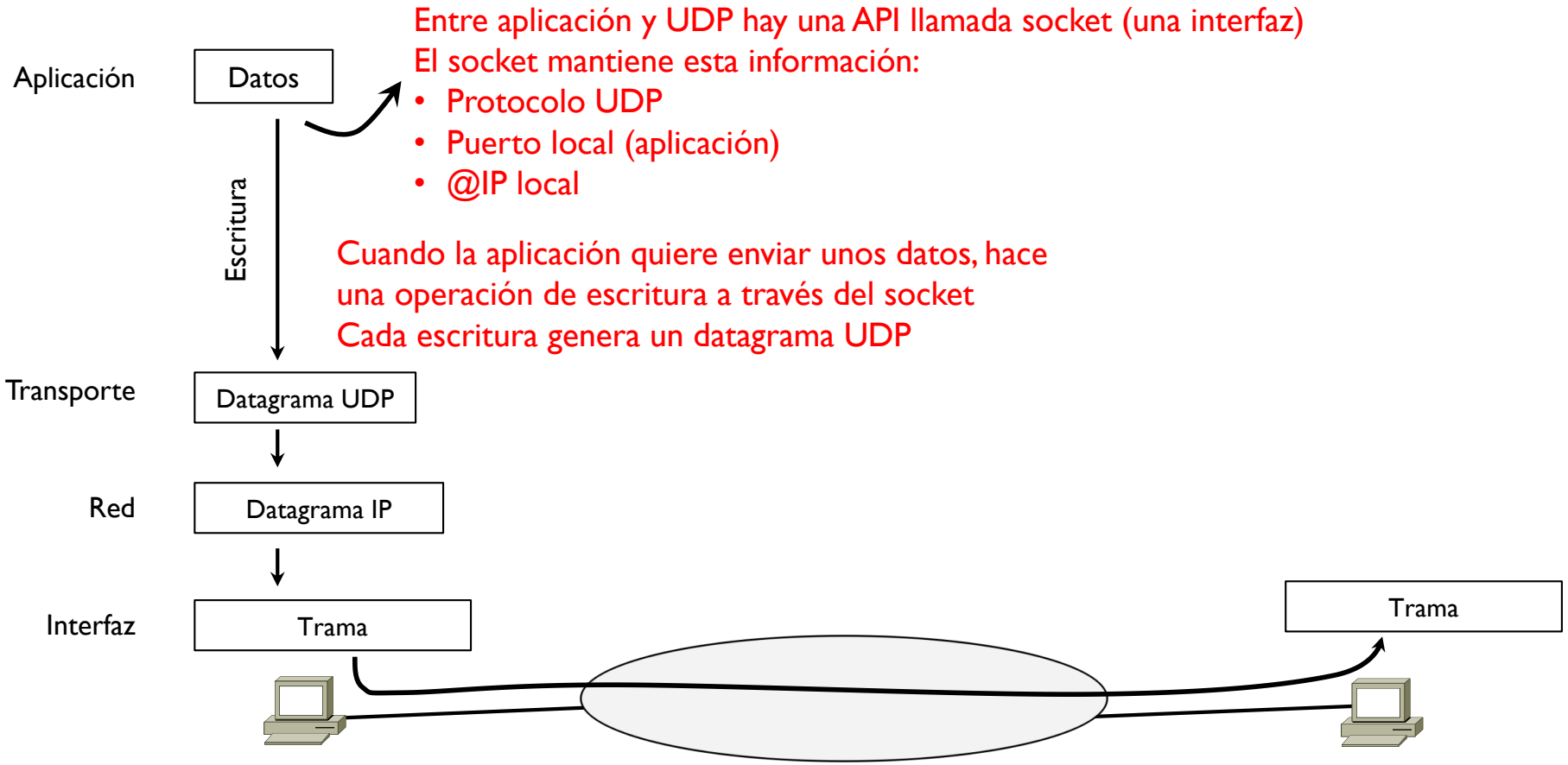
Tema 3 – Protocolos UDP y TCP

- a) Introducción
- **b) El protocolo UDP**
- c) El protocolo TCP
  - Arquitectura
  - El MSS
  - Números de secuencia
  - Establecimiento y terminación de una conexión TCP
  - Funcionamiento durante la transmisión
    - Control de flujo
    - Control de congestión
  - Cabecera TCP

Tema 3 – User Datagram Protocol (UDP)

RFC 768

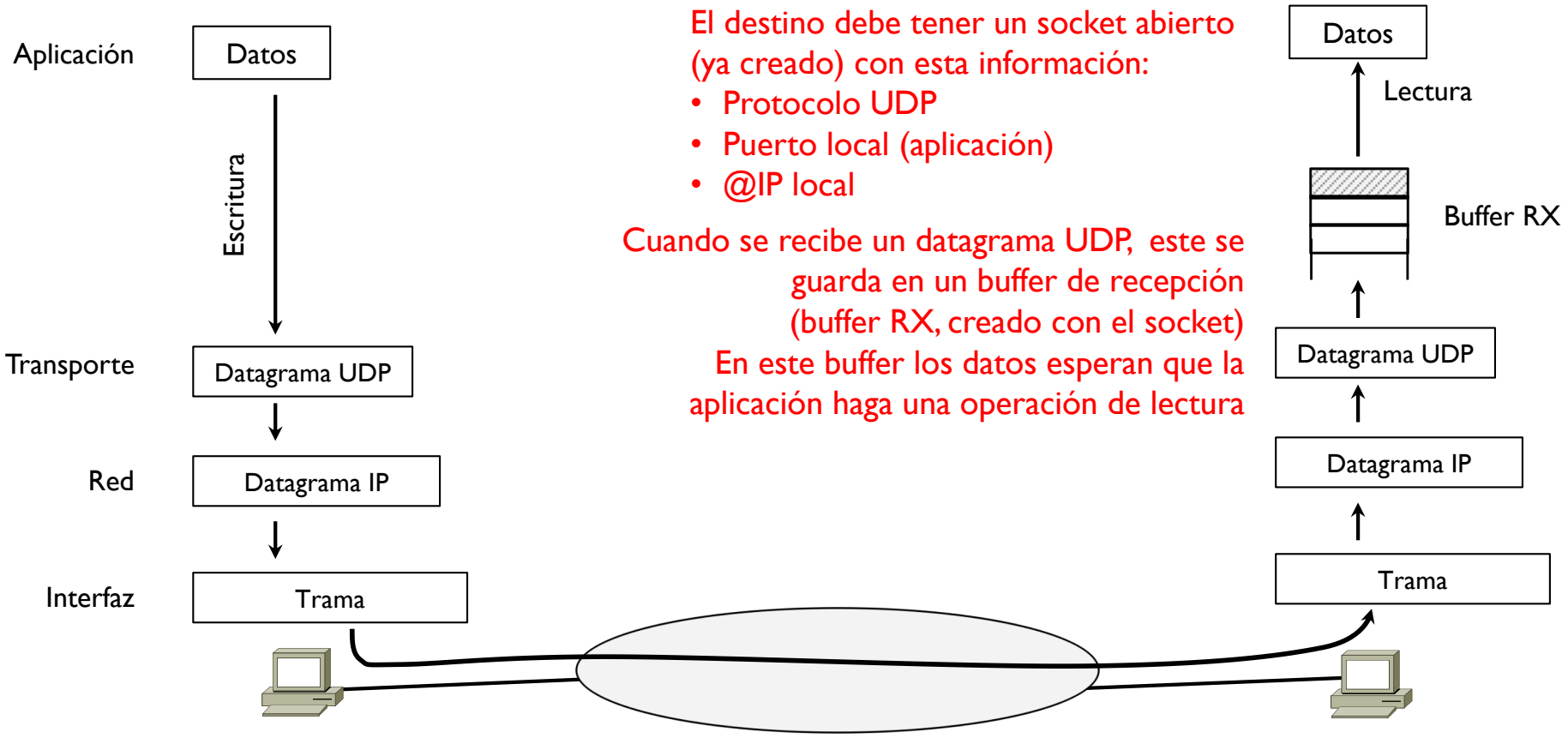
- Protocolo extremo a extremo que proporciona solamente multiplexación de aplicaciones mediante puertos



Tema 3 – User Datagram Protocol (UDP)

RFC 768

- Protocolo extremo a extremo que proporciona solamente multiplexación de aplicaciones mediante puertos

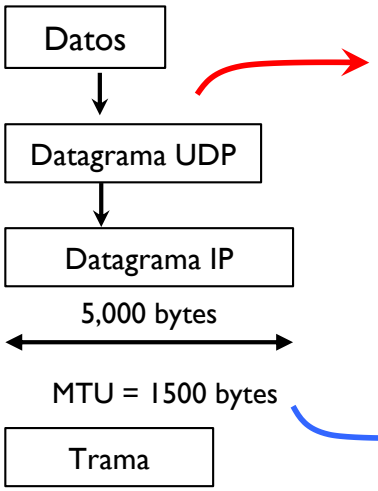


Tema 3 – User Datagram Protocol (UDP)

- La aplicación por lo tanto cuando debe enviar unos datos, hace una operación de escritura a través del socket
- Esta escritura genera finalmente un datagrama UDP

- ¿qué pasa si el datagrama UDP se encapsula en un datagrama IP que luego es demasiado grande para la interfaz de red?
  - Es decir la longitud del datagrama IP es superior a la MTU de la interfaz de red

- Dos soluciones:



Se configura la aplicación para que genere datos de tamaño limitado (por ejemplo nunca más de 512 bytes)

Se fragmenta el datagrama IP en el host origen

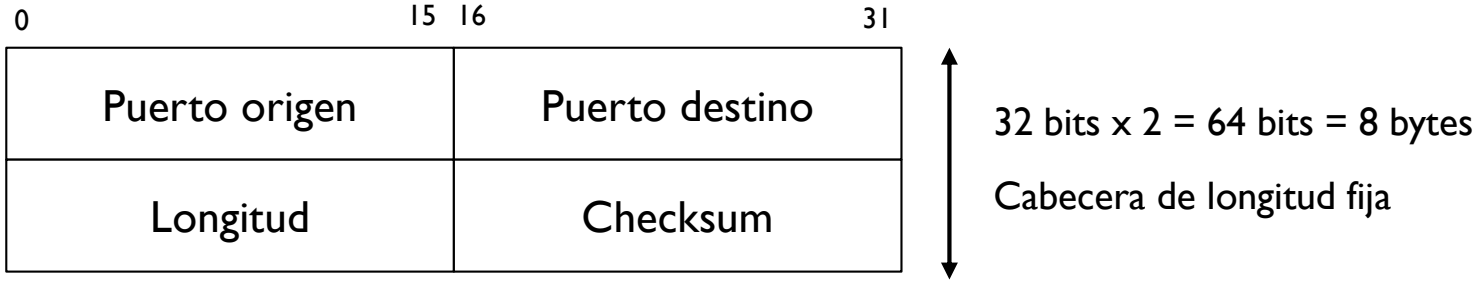
Tema 3 – User Datagram Protocol (UDP)

UDP se usa para aplicaciones

- Que no necesitan fiabilidad porque envían mensajes periódicos
  - Por ejemplo se ha visto el caso de RIP
  - También aplicaciones como DHCP y DNS (se verá más adelante)

- Que son real-time, donde el objetivo es enviar rápidamente los datos y sin que haya parones debido a la recuperación de información perdida
  - Streaming audio/video
  - Voice over IP
  - Videoconferencias
  - Es decir se prefiere perder un poco de calidad que esperar o bloquear el streaming de datos

Tema 3 – Cabecera UDP



- Puerto origen: identifica la aplicación origen de los datos
- Puerto destino: identifica la aplicación destino de los datos
- Longitud: longitud total del datagrama UDP, es decir datos + cabecera UDP
- Checksum: control de error de lectura de la información

Tema 3 – Protocolos UDP y TCP

- a) Introducción
- b) El protocolo UDP
- **c) El protocolo TCP**
  - Arquitectura
  - El MSS
  - Números de secuencia
  - Establecimiento y terminación de una conexión TCP
  - Funcionamiento durante la transmisión
    - Control de flujo
    - Control de congestión
  - Cabecera TCP

Tema 3 – Transmission Control Protocol (TCP)

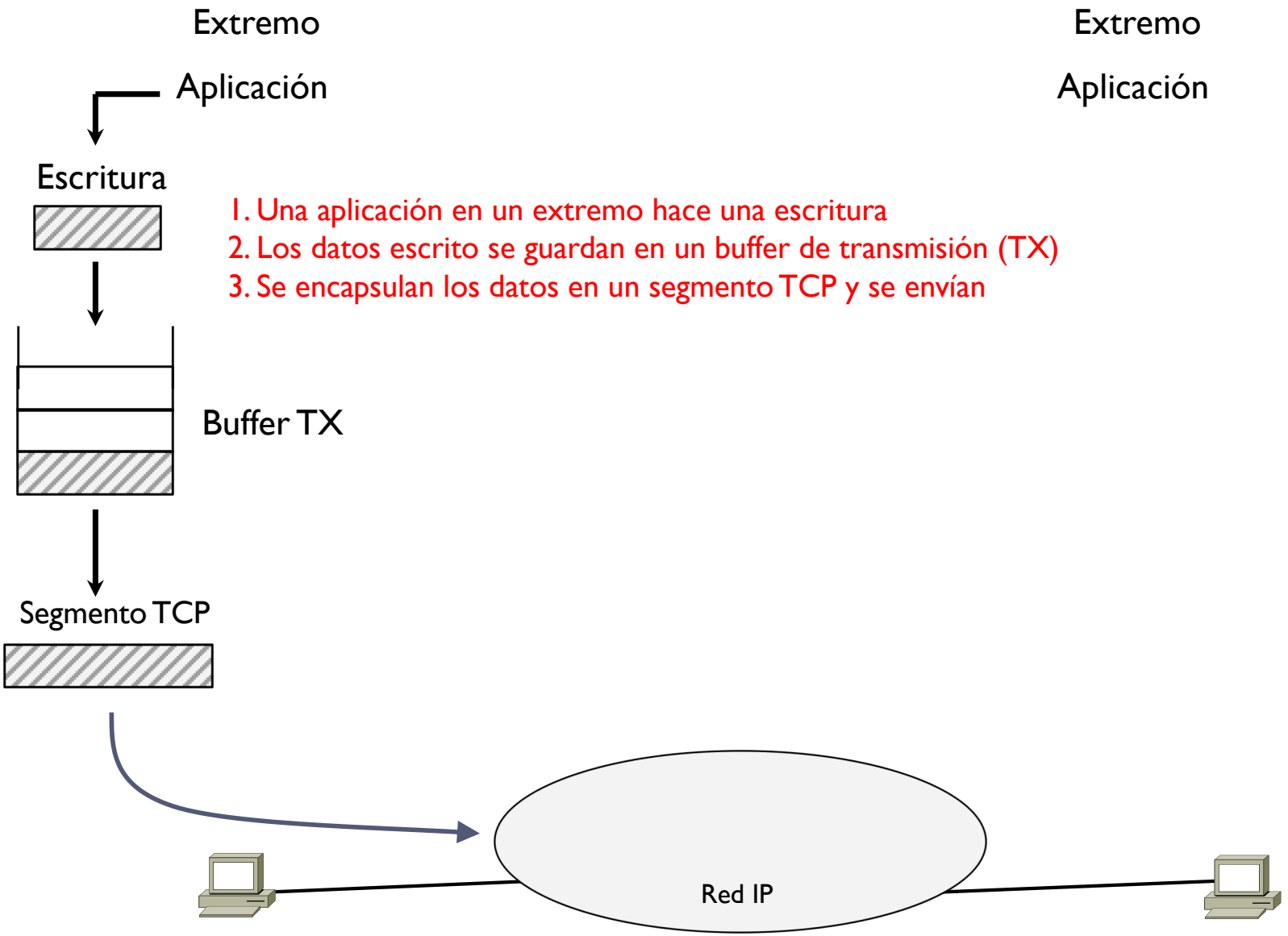
RFC 793

- Basado en el paradigma cliente – servidor
- Multiplexación de aplicaciones
  - Identificación de aplicaciones a través de puertos
- Orientado a la conexión
  - Un extremo debe previamente avisar el otro extremo (fase de establecimiento de la conexión)
  - También debe avisar para terminar (fase de terminación)
  - Extremo origen y extremo destino deben mantener el mismo orden en los datos
- Fiable
  - Si un dato se pierde, proporciona un mecanismo para recuperarlo y retransmitirlo

Tema 3 – Transmission Control Protocol (TCP)

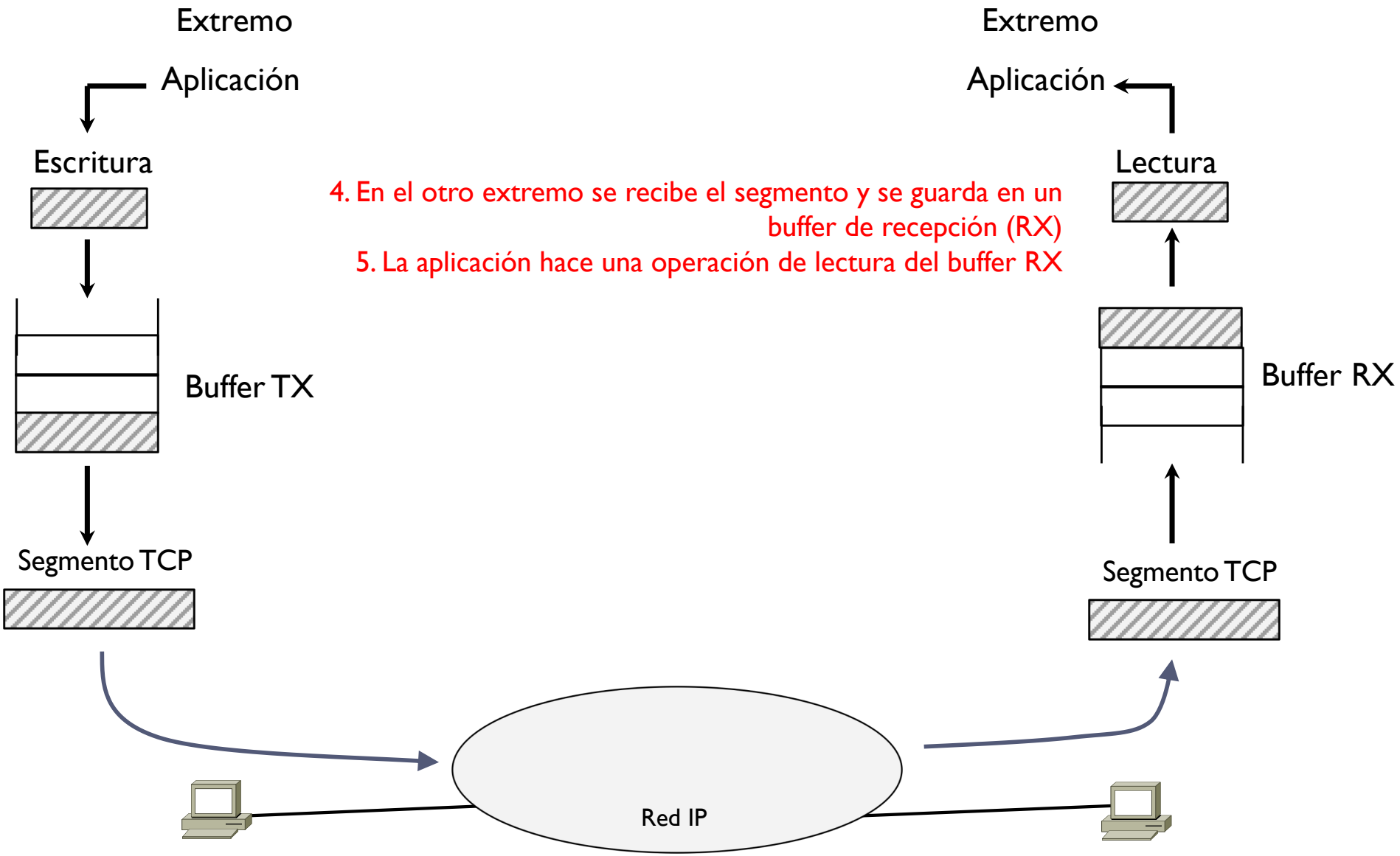
- Arquitectura del TCP con buffer de TX y RX
- Unidad de información es el segmento
  - Concepto de MSS y de número de secuencia
- Confirmaciones (ack)
  - Se necesita usar ack para saber si el dato ha sido recibido correctamente
- Temporizador (RTO)
  - Al transmitir el primer segmento, se inicializa el RTO
  - Cada vez que se recibe un ack nuevo, se reinicializa el RTO
  - Si pasado un RTO no se ha recibido el ack, se da por perdido el segmento, se vuelve a sacar del buffer de TX y se vuelve a enviar
- Control de flujo y control de congestión
  - TCP aplica un mecanismo de ventana deslizante para adaptar la tasa de envío de datos a la capacidad del extremo receptor y de la red

Tema 3 – Arquitectura del TCP

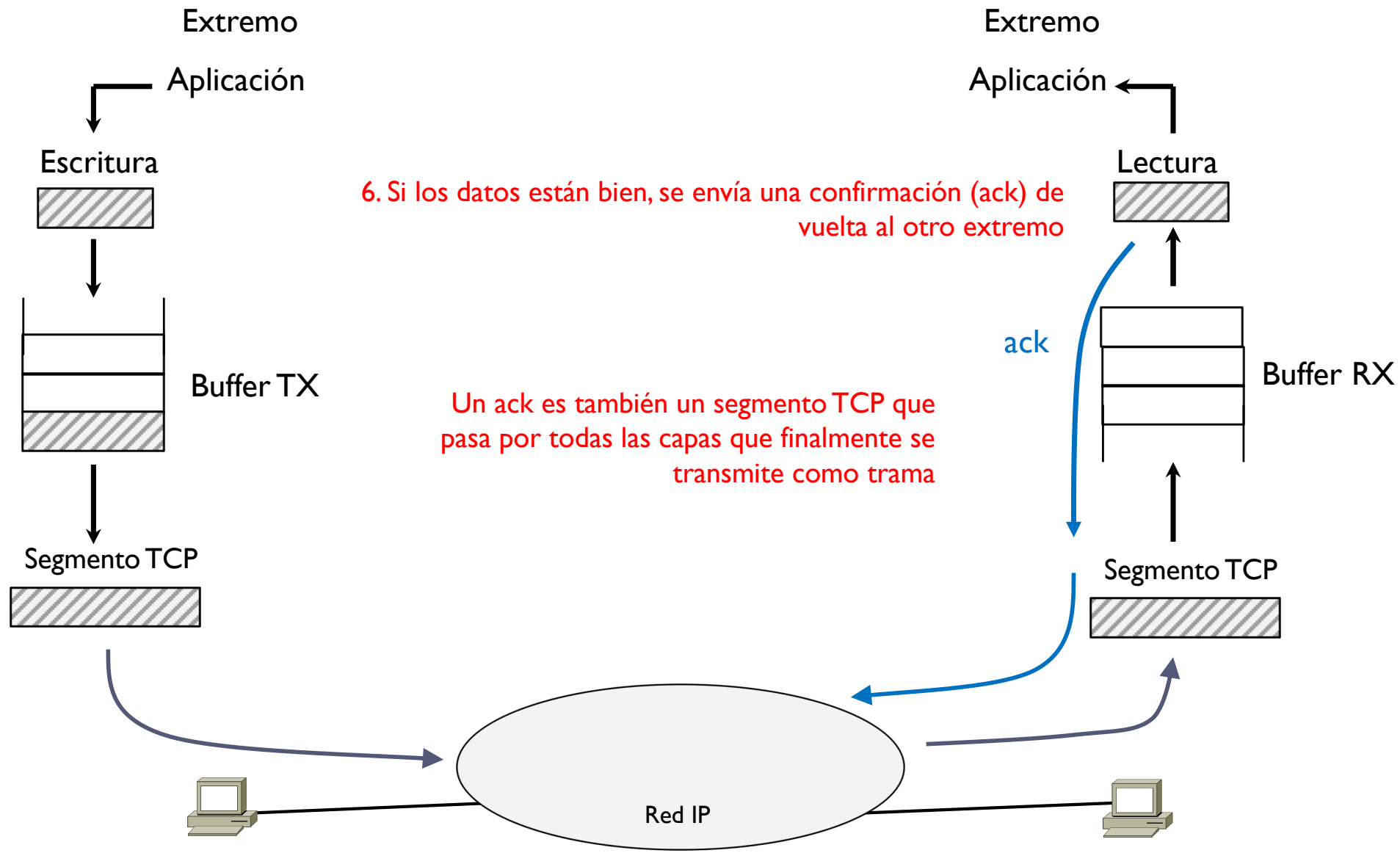


1. Una aplicación en un extremo hace una escritura
2. Los datos escrito se guardan en un buffer de transmisión (TX)
3. Se encapsulan los datos en un segmento TCP y se envían

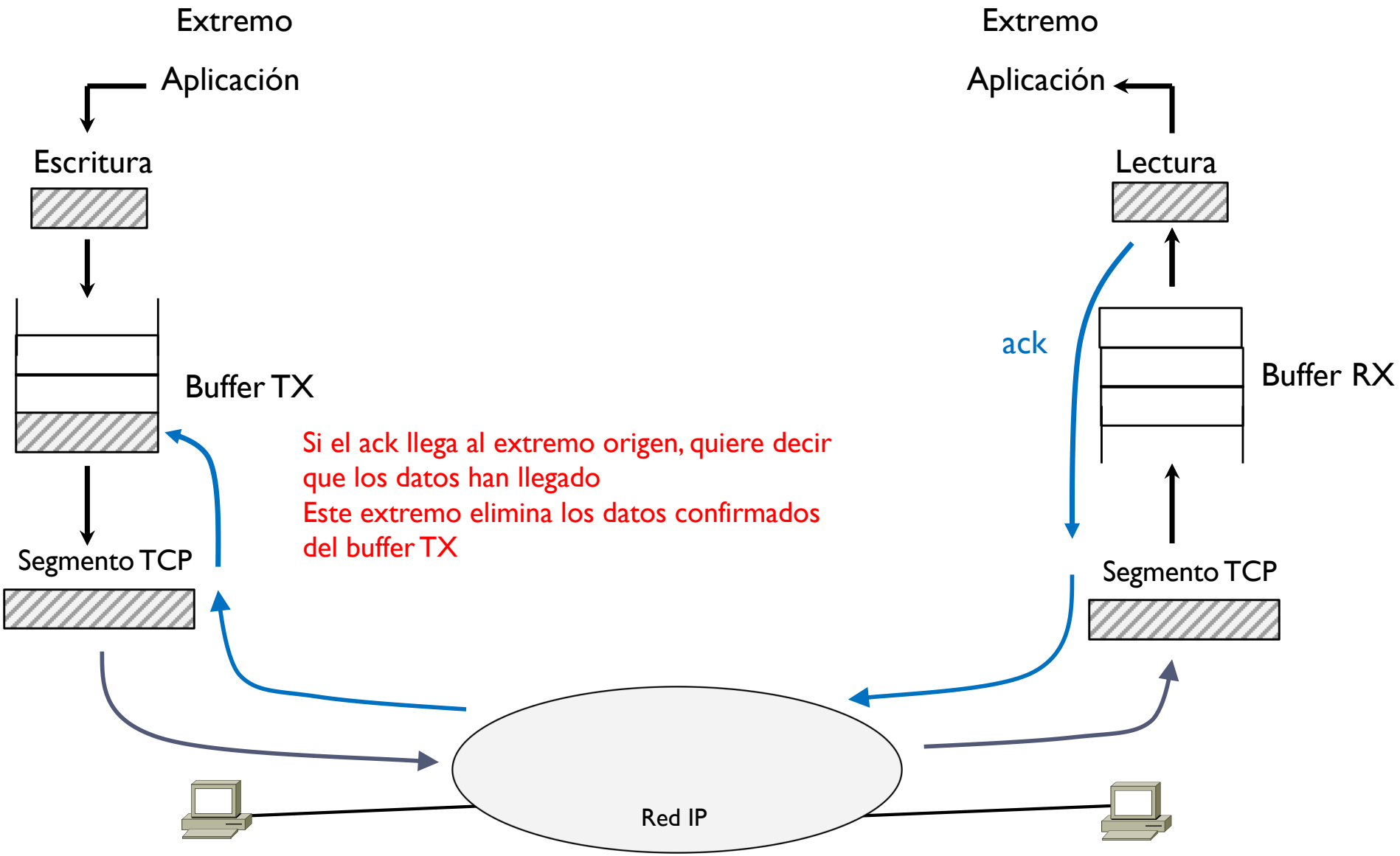
Tema 3 – Arquitectura del TCP



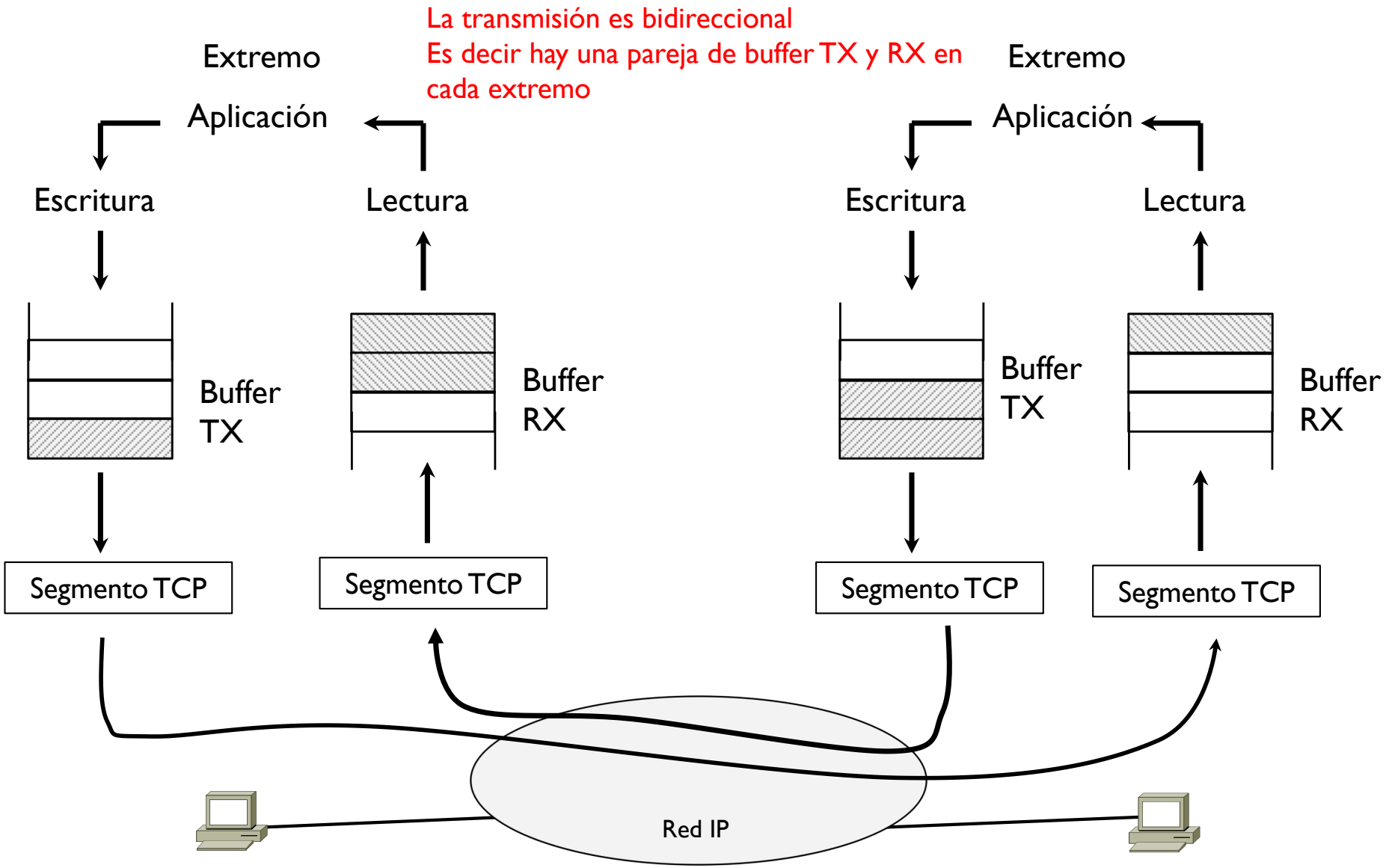
Tema 3 – Arquitectura del TCP



Tema 3 – Arquitectura del TCP



Tema 3 – Arquitectura del TCP



Tema 3 – Transmission Control Protocol (TCP)

- Arquitectura del TCP con buffer de TX y RX
- Unidad de información es el segmento
  - Concepto de MSS y de número de secuencia
- Confirmaciones (ack)
  - Se necesita usar ack para saber si el dato ha sido recibido correctamente
- Temporizador (RTO)
  - Al transmitir el primer segmento, se inicializa el RTO
  - Cada vez que se recibe un ack nuevo, se reinicializa el RTO
  - Si pasado un RTO no se ha recibido el ack, se da por perdido el segmento, se vuelve a sacar del buffer de TX y se vuelve a enviar
- Control de flujo y control de congestión
  - TCP aplica un mecanismo de ventana deslizante para adaptar la tasa de envío de datos a la capacidad del extremo receptor y de la red

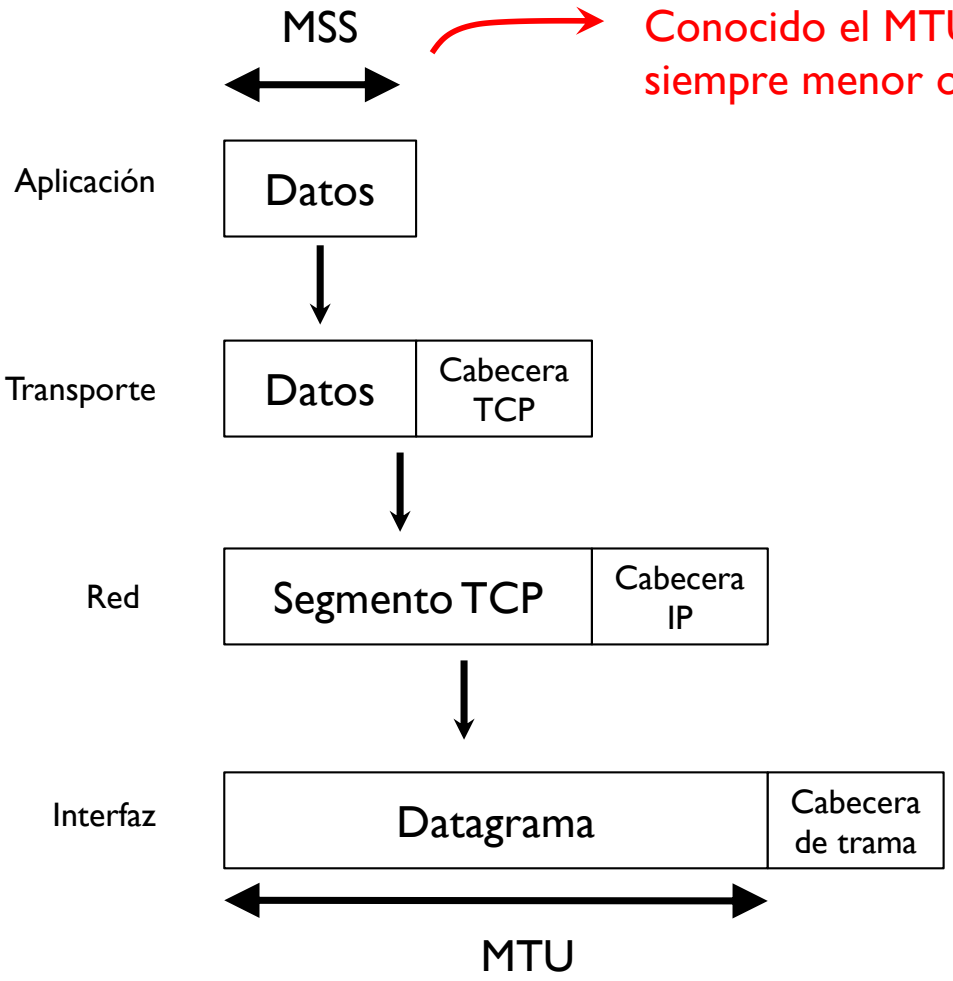
Tema 3 – Concepto de MSS

- MSS es un parámetro del TCP
  - Maximum Segment Size
  - Tamaño máximo de un segmento

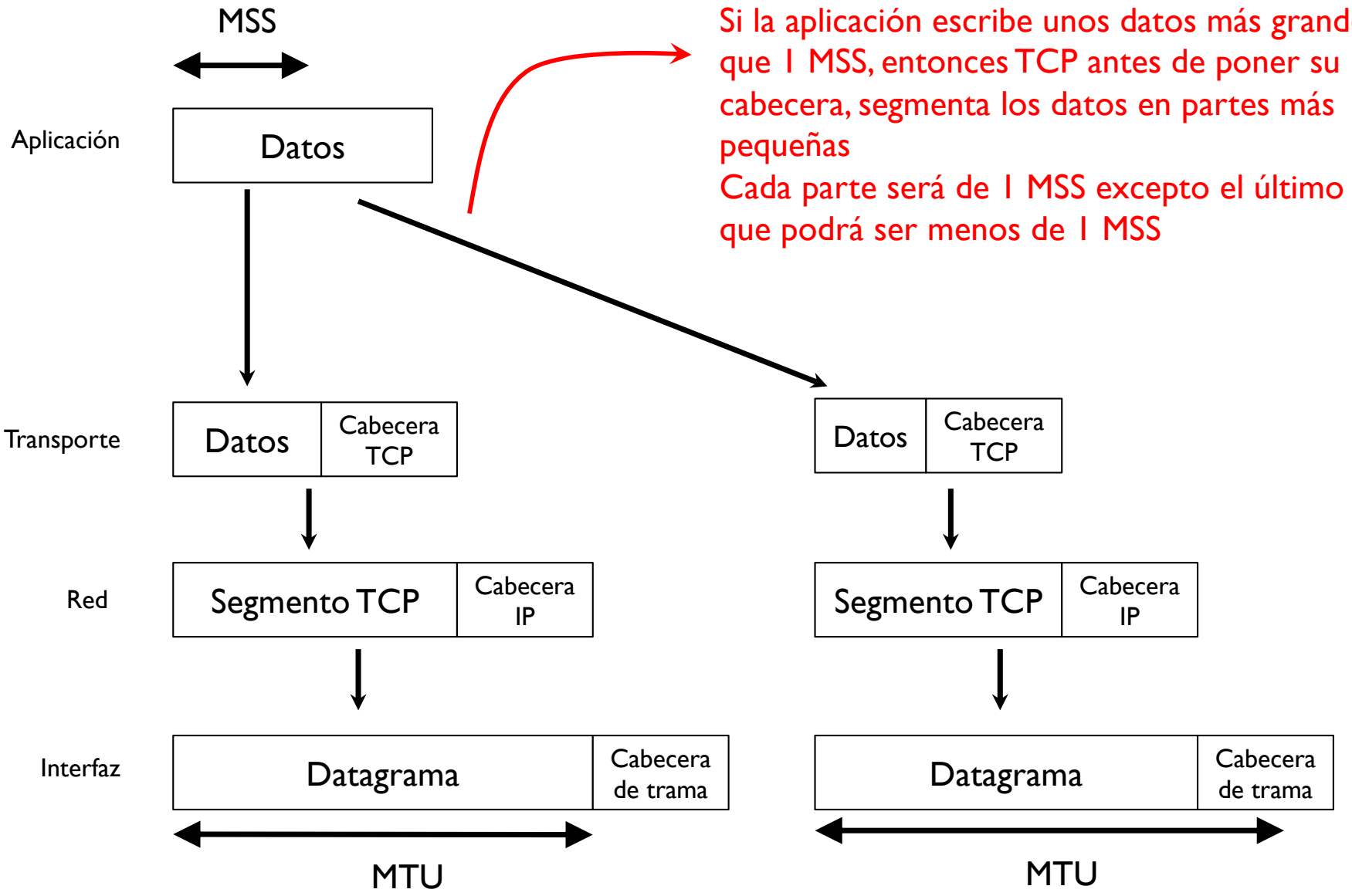
- Un extremo determina el MSS usando la MTU de su interfaz de red
  - MSS = MTU – LongitudCabeceraIP – LongitudCabeceraTCP

  - Puede usar MTU path discovery para saber la máxima MTU posible que evite la fragmentación en el camino hacia un determinado destino

Tema 3 – Concepto de MSS

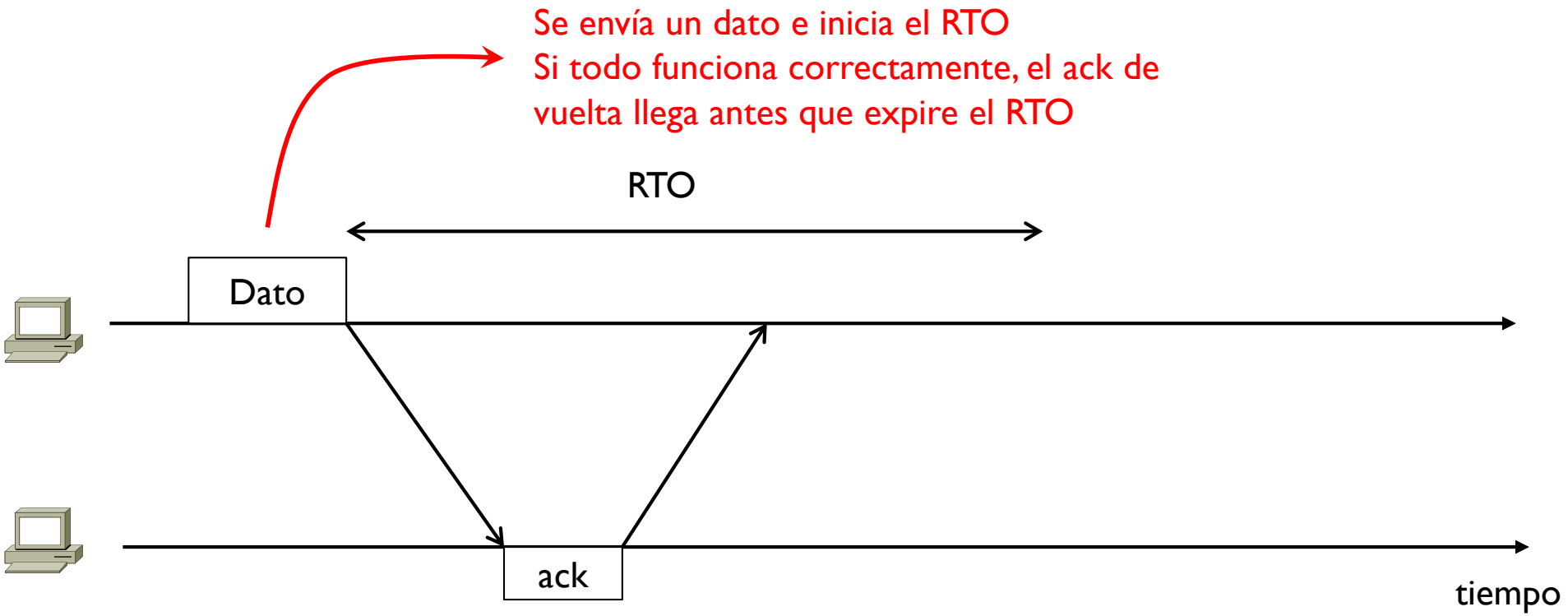


Tema 3 – Concepto de MSS

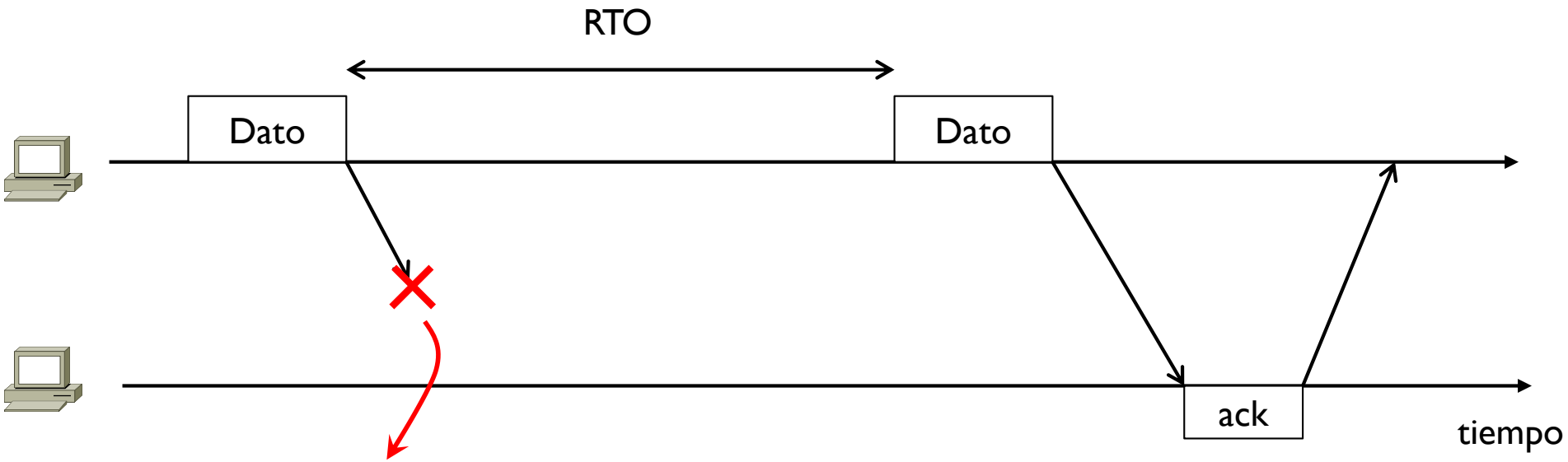


Tema 3 – Números de secuencia

- Para relacionar unos datos transmitido con el correspondiente ack, se usan números de secuencia
- ¿por qué?

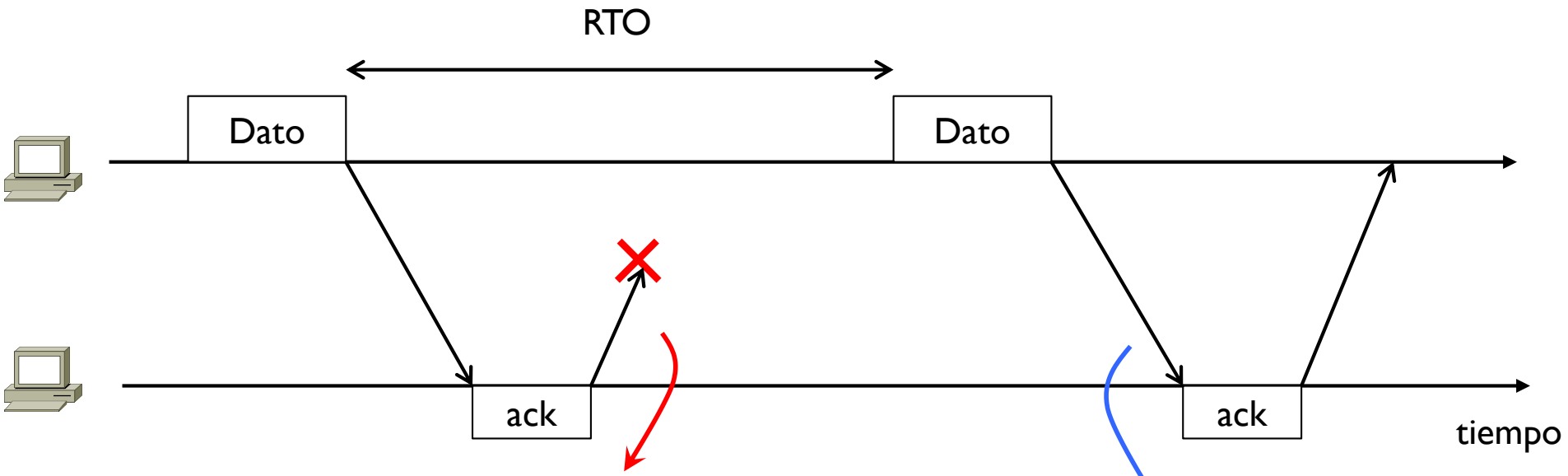


Tema 3 – Números de secuencia



Suponiendo que se pierde un dato en el camino hacia el otro extremo, entonces el primer extremo no recibe el ack
Pasado el RTO, el extremo coge el dato del buffer de TX y vuelve a transmitirlo

Tema 3 – Números de secuencia



Suponer que lo que se pierde es el ack (ya se ha dicho que un ack es también un segmento TCP) que se transmite como cualquier otra información

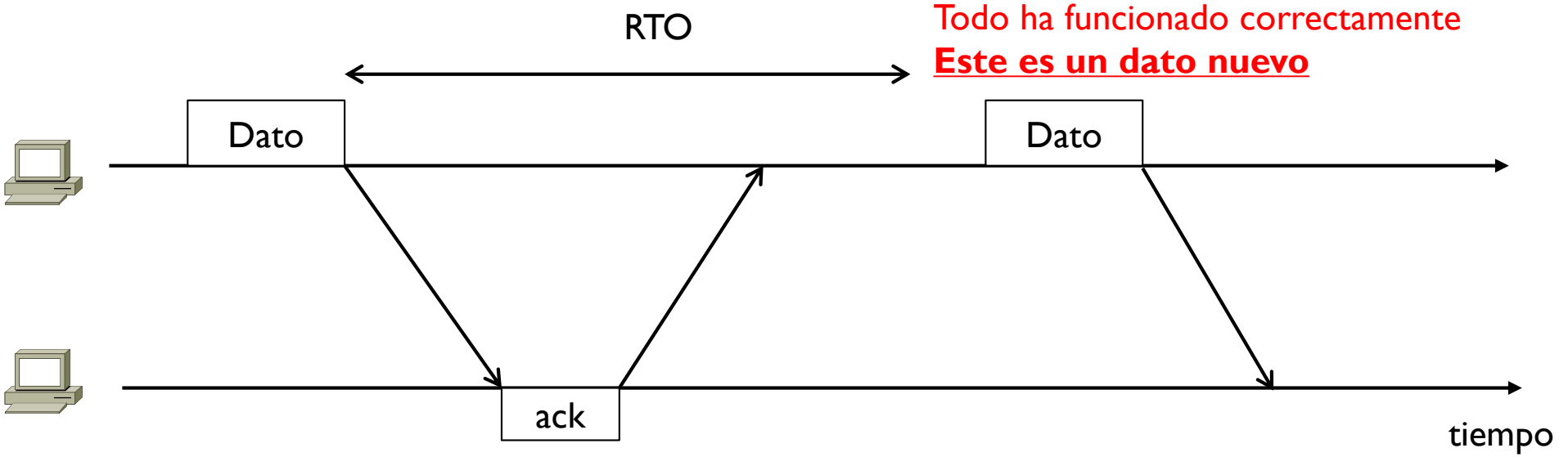
En este caso el extremo ha recibido correctamente el dato

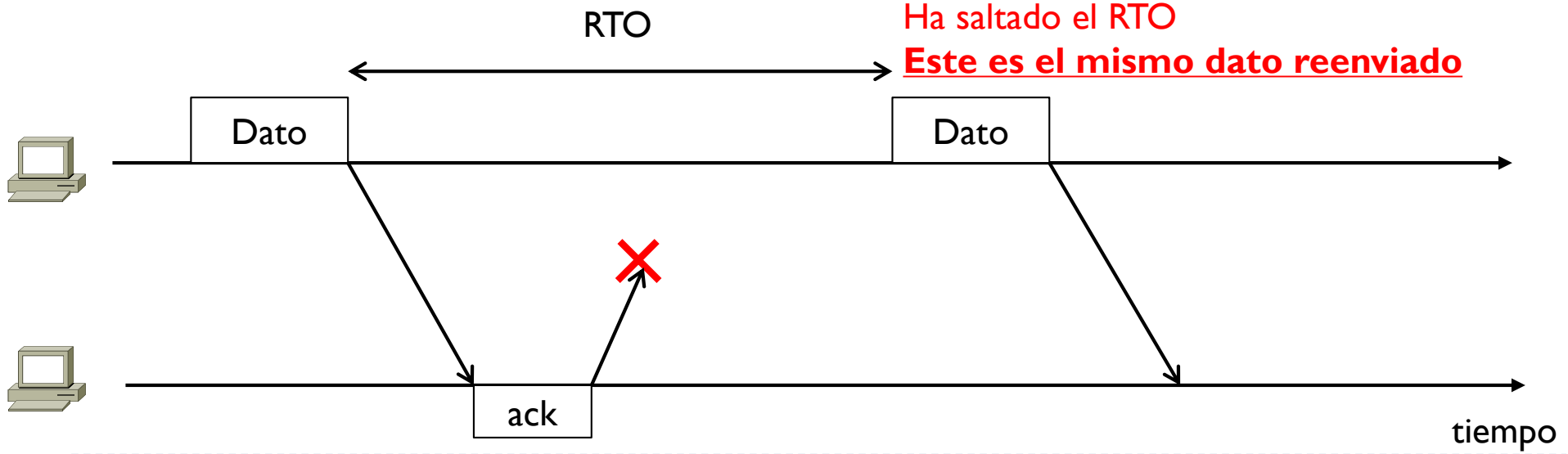
El extremo que transmite pero no se puede enterar; pasado el RTO, este coge el dato del buffer de TX y vuelve a transmitirlo

Vuelve a recibir el mismo dato de antes.

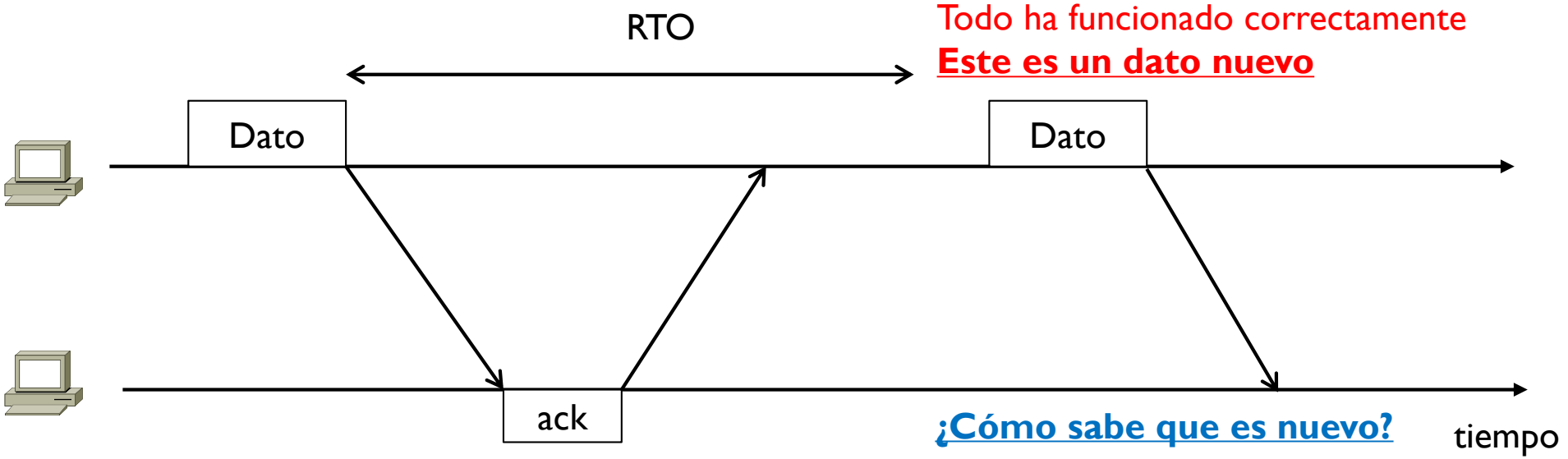
Pero, ¿cómo sabe que es el mismo de antes? Podría también ser uno nuevo…

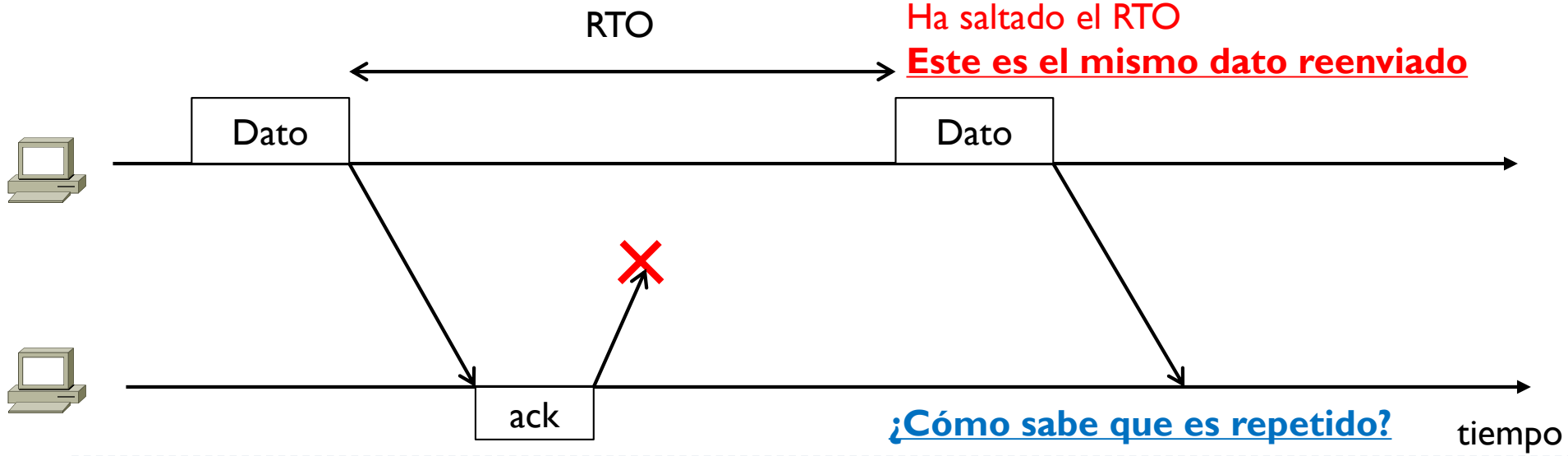
Tema 3 – Números de secuencia



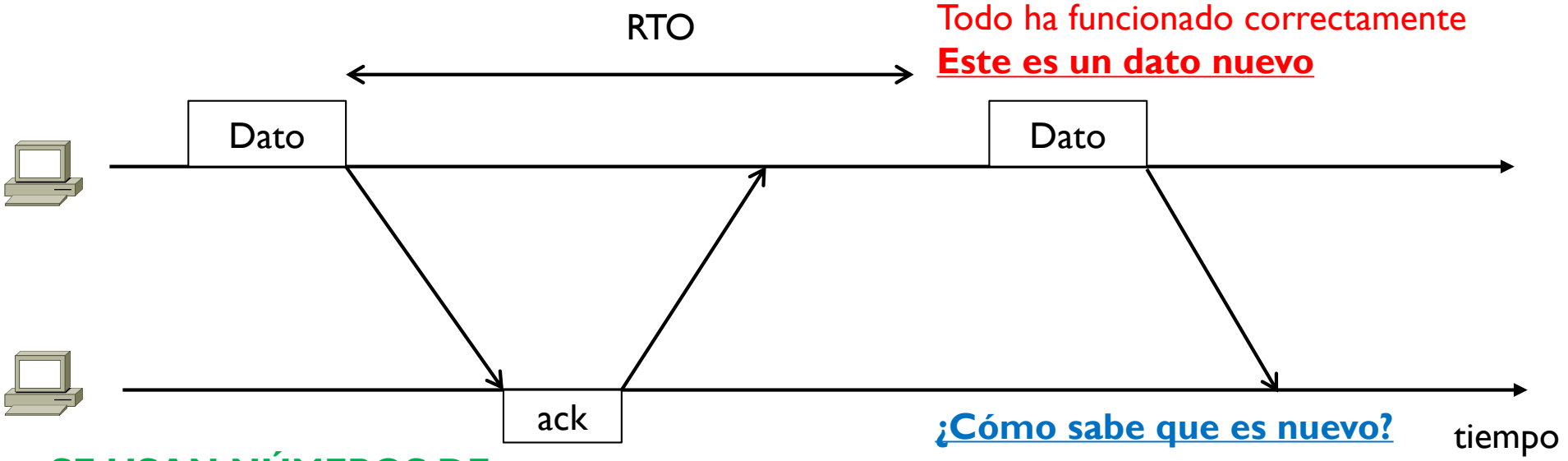


Tema 3 – Números de secuencia

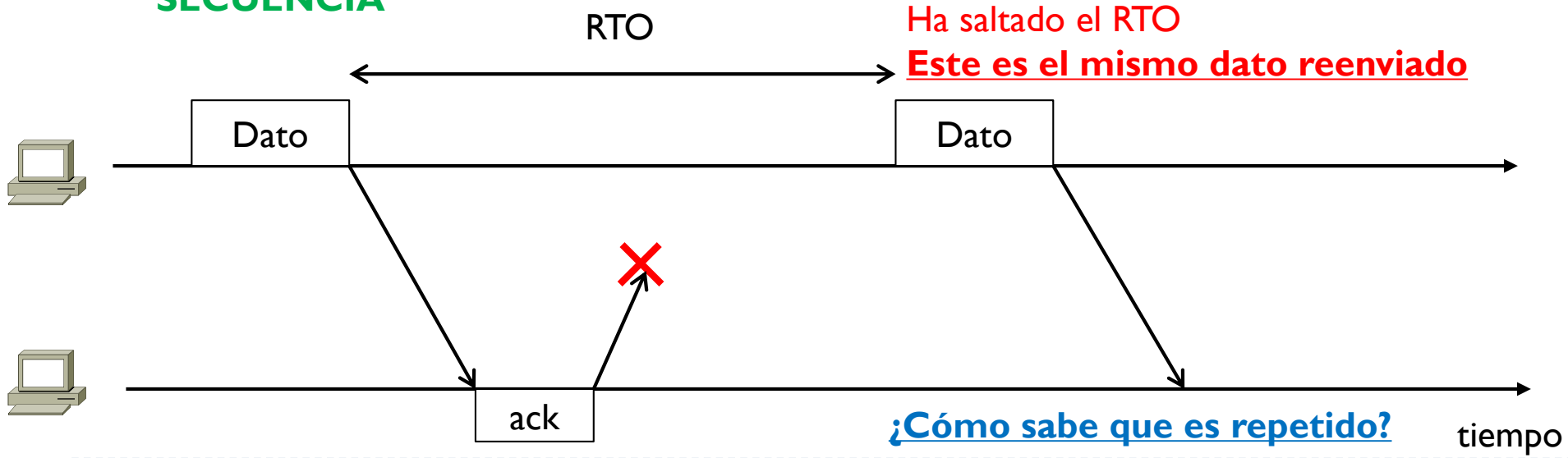




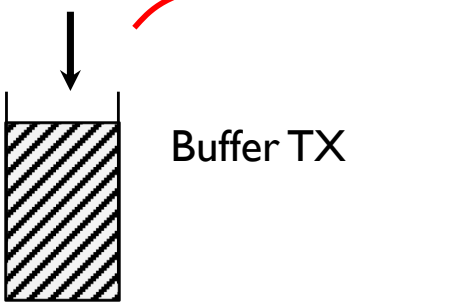
Tema 3 – Números de secuencia



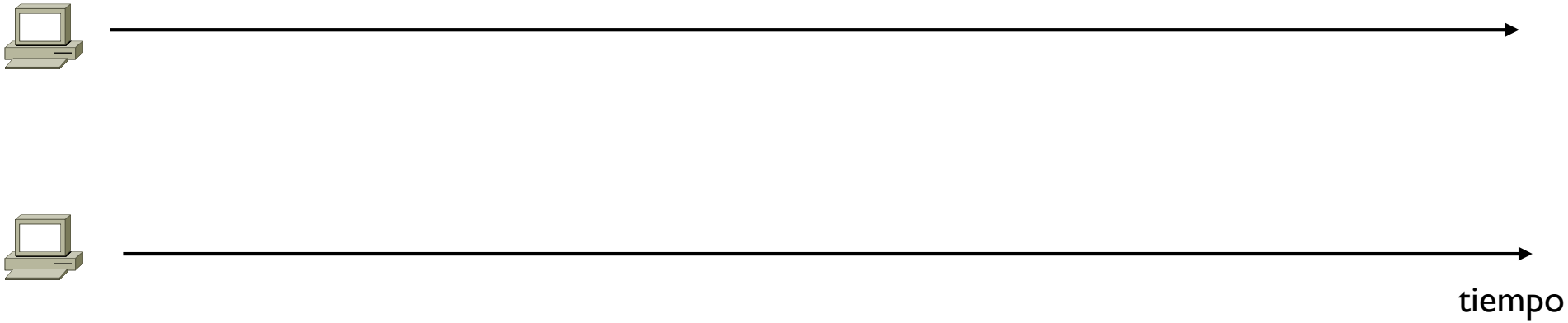
SE USAN NÚMEROS DE SECUENCIA



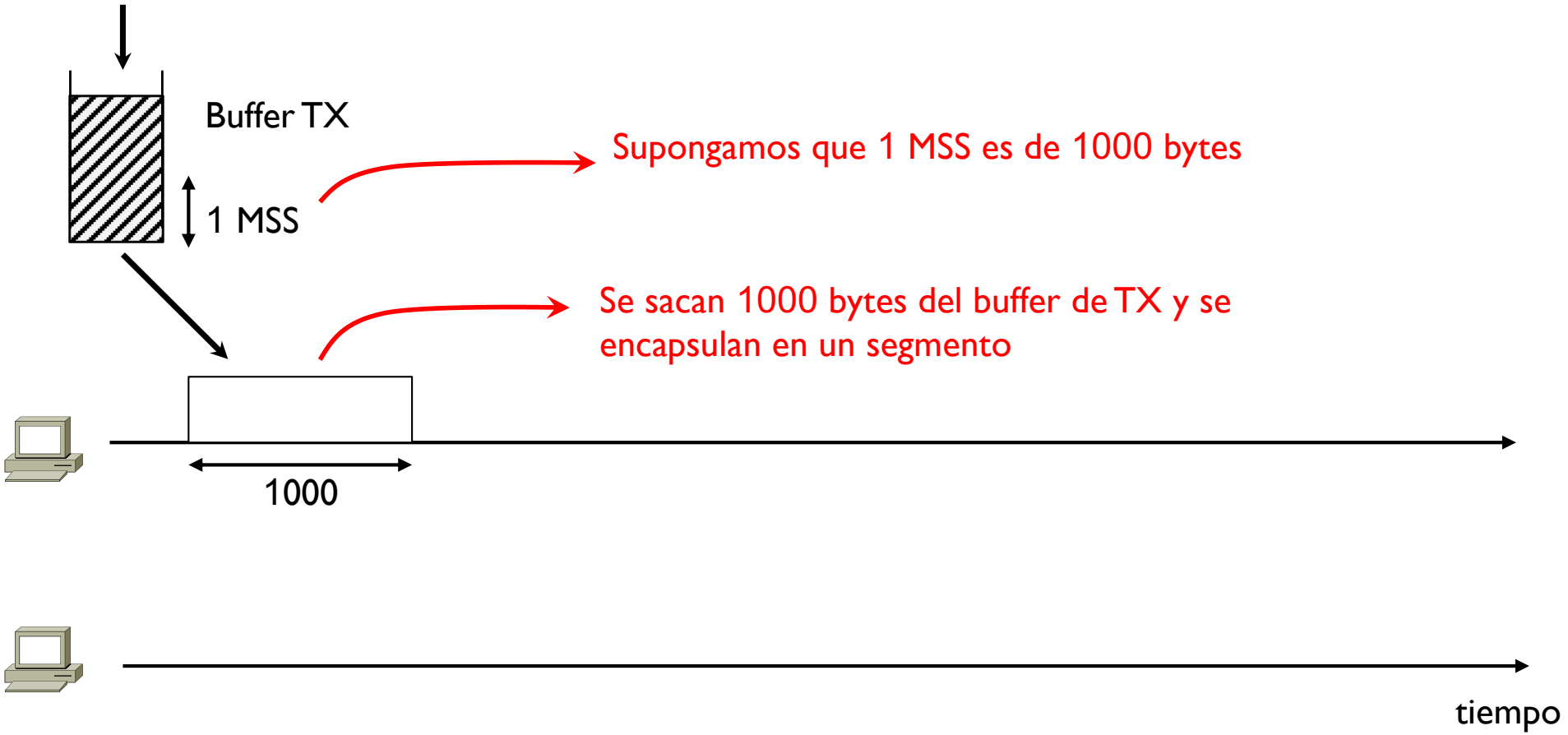
Tema 3 – Números de secuencia



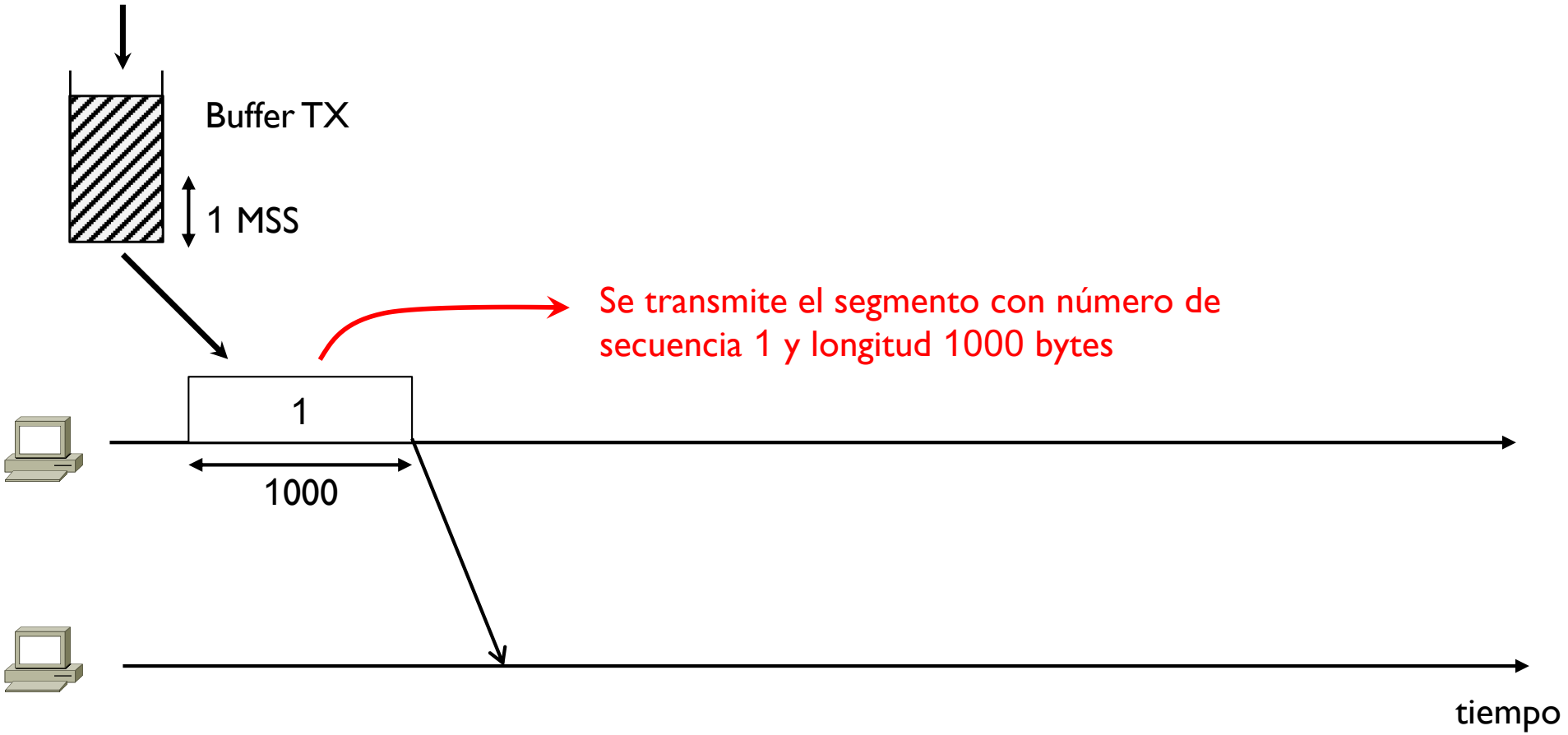
Supongamos se quieren transmitir un total de 2500 bytes de datos
La operación de escritura de la aplicación llena el buffer de TX con 2500 bytes de datos



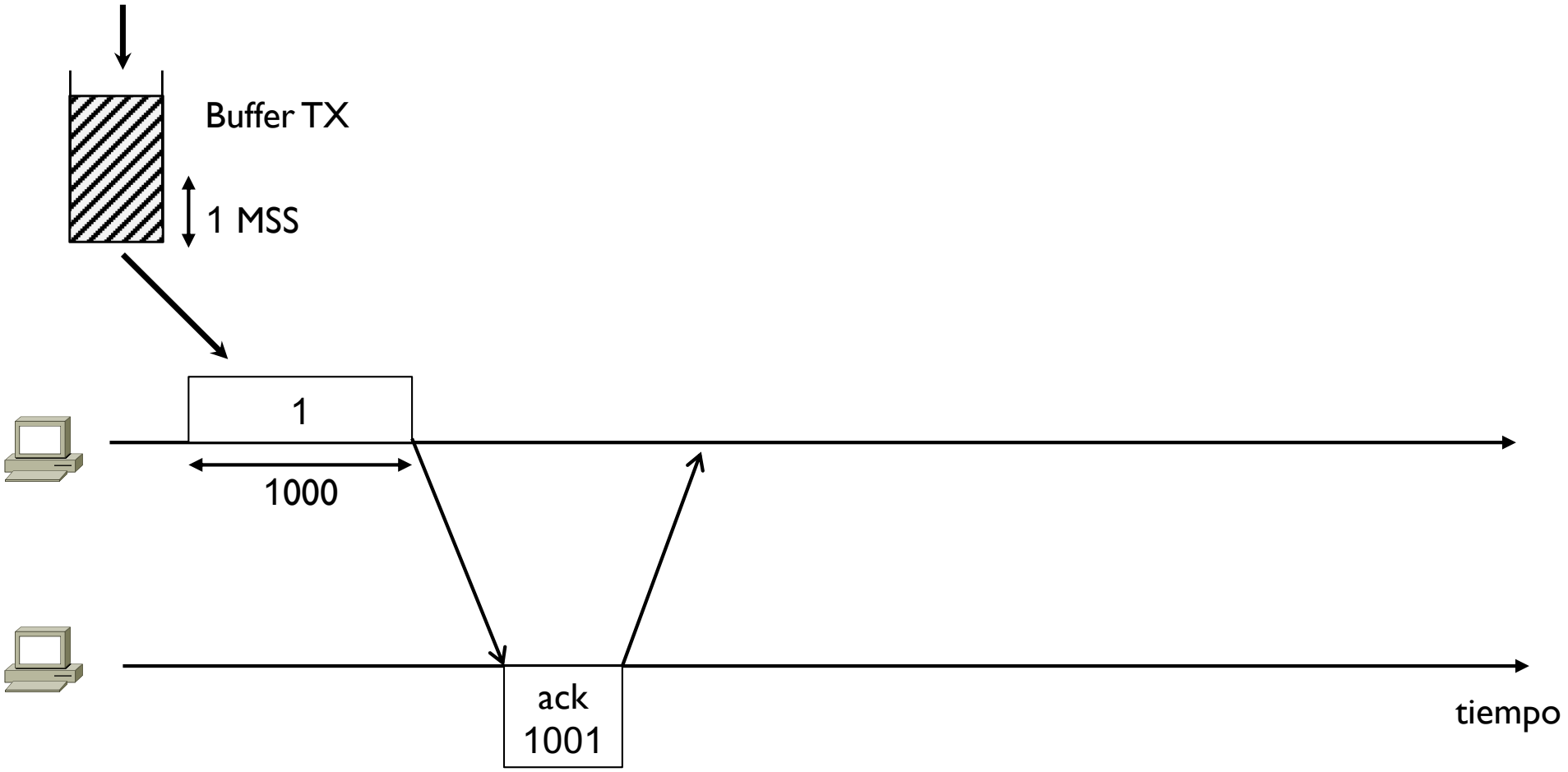
Tema 3 – Números de secuencia



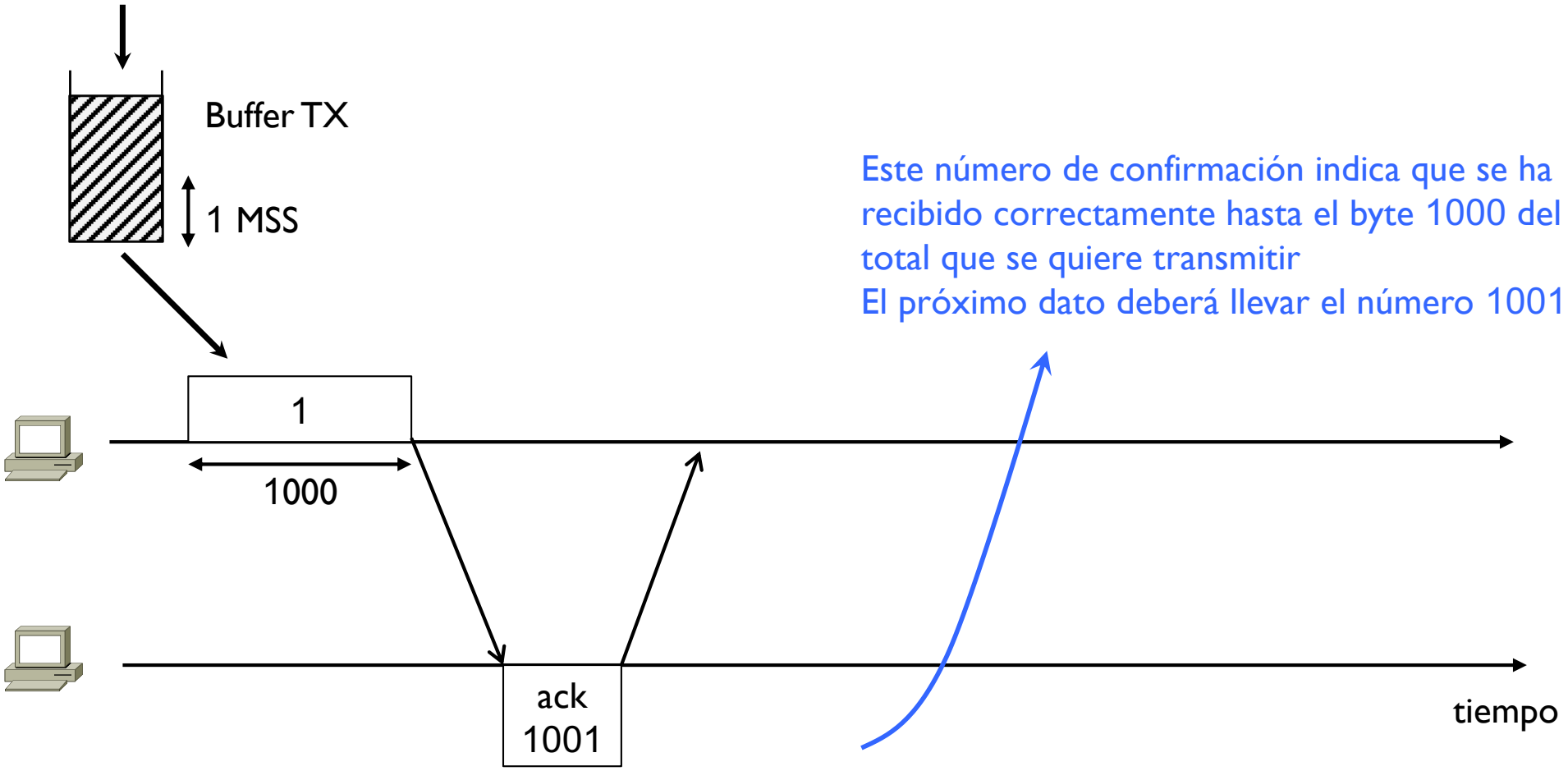
Tema 3 – Números de secuencia



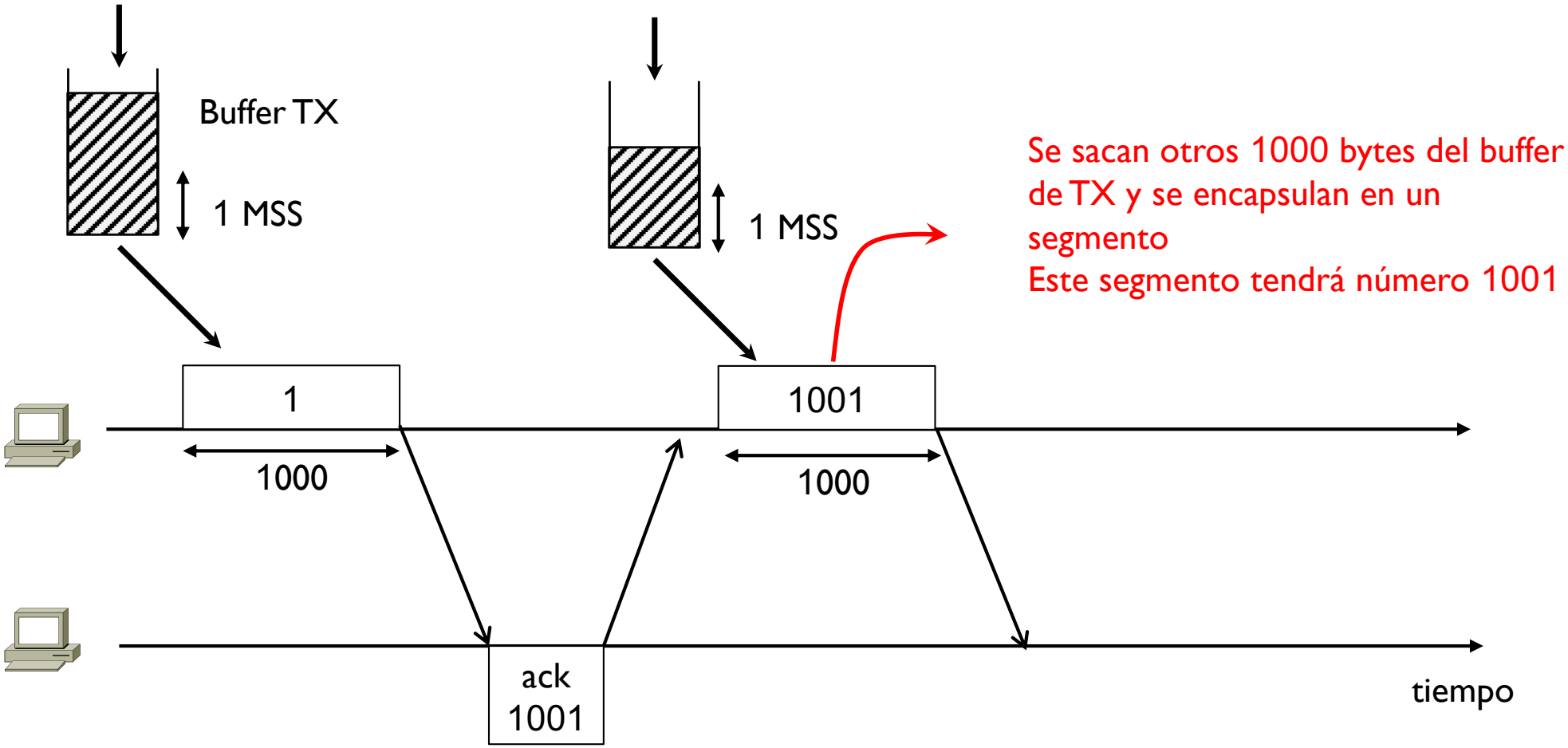
Tema 3 – Números de secuencia



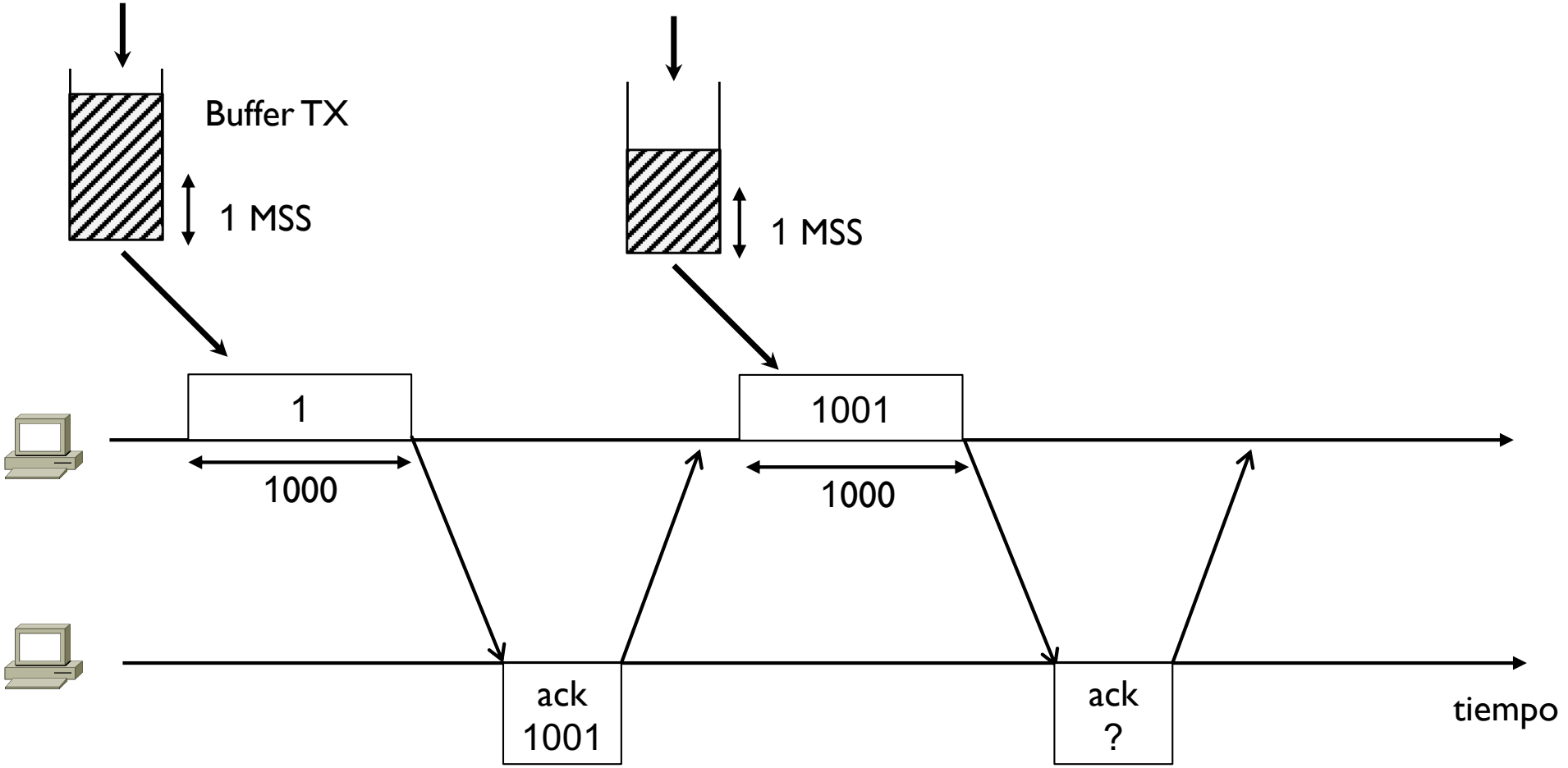
Tema 3 – Números de secuencia



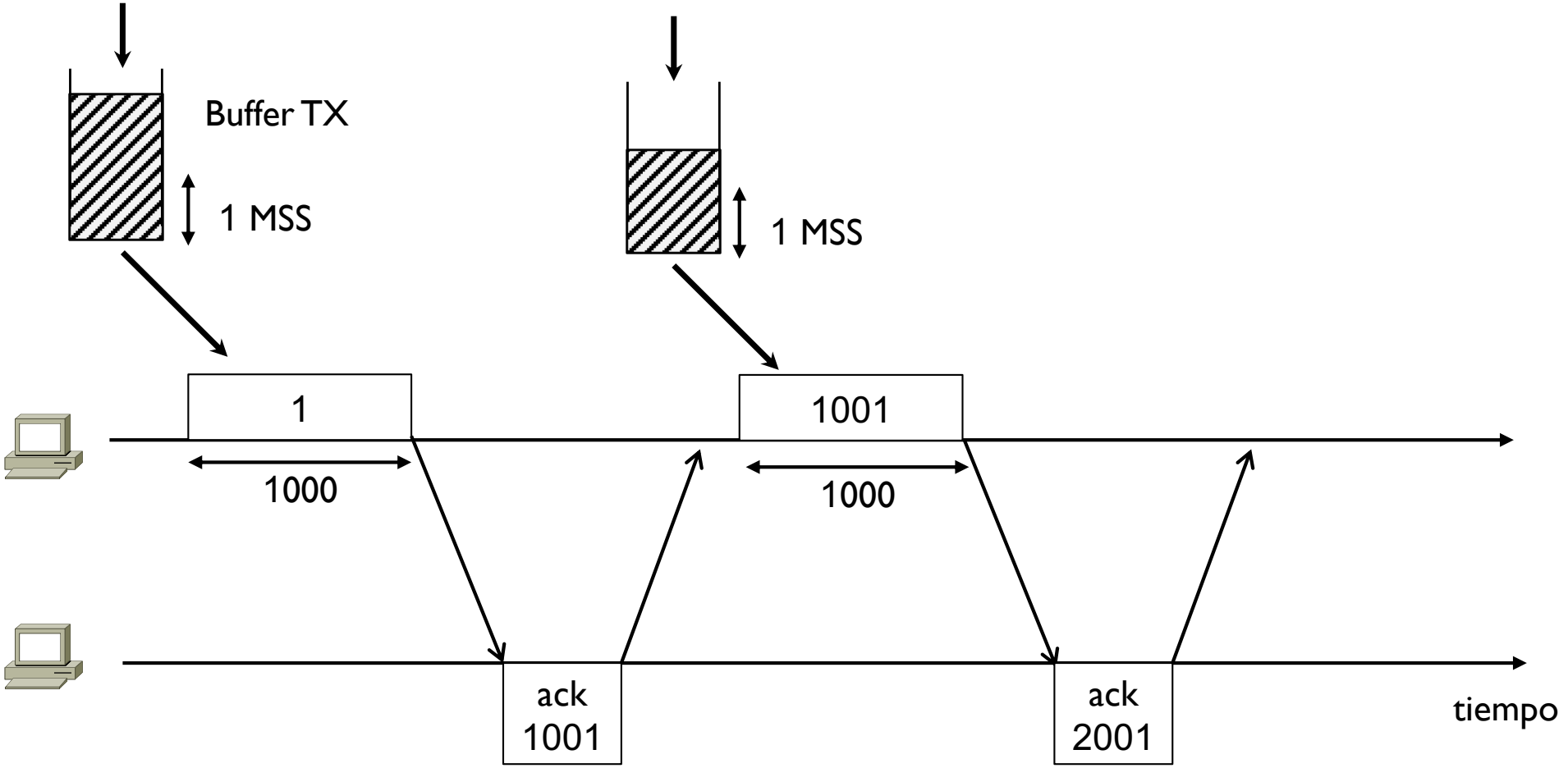
Tema 3 – Números de secuencia



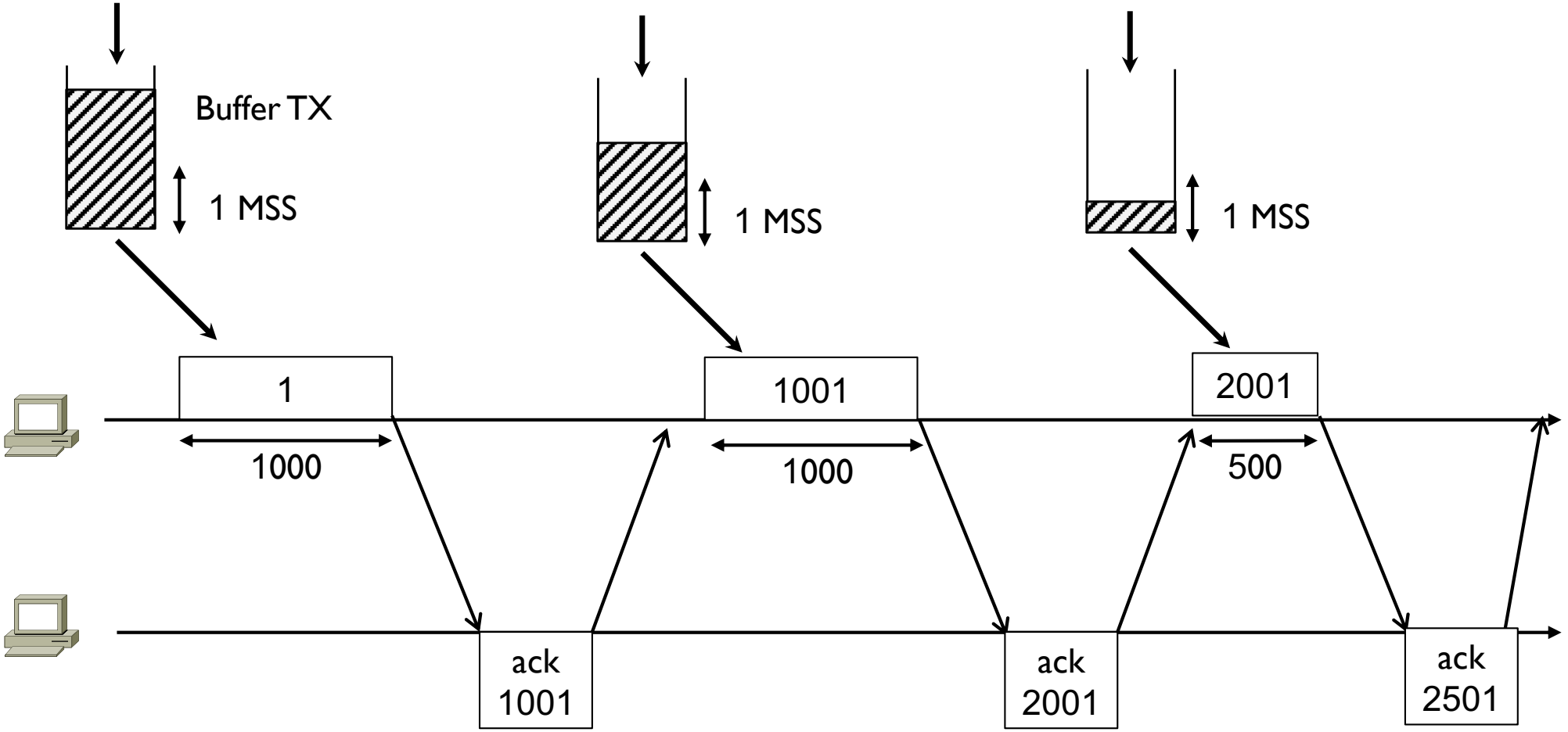
Tema 3 – Números de secuencia



Tema 3 – Números de secuencia

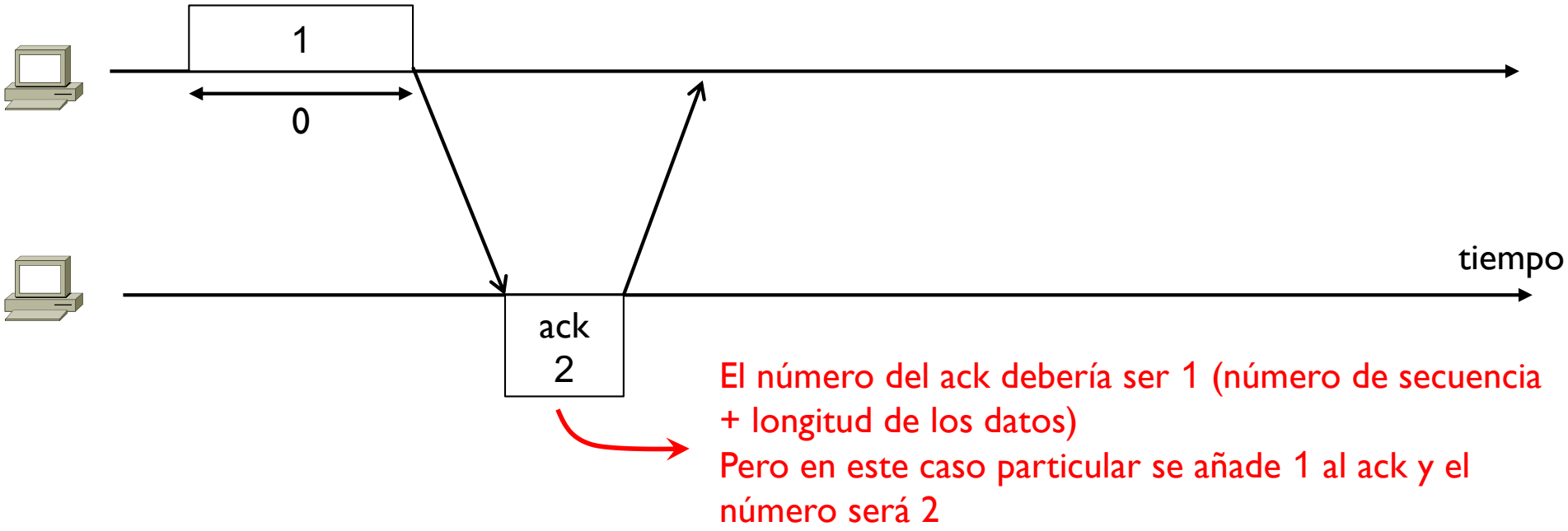


Tema 3 – Números de secuencia



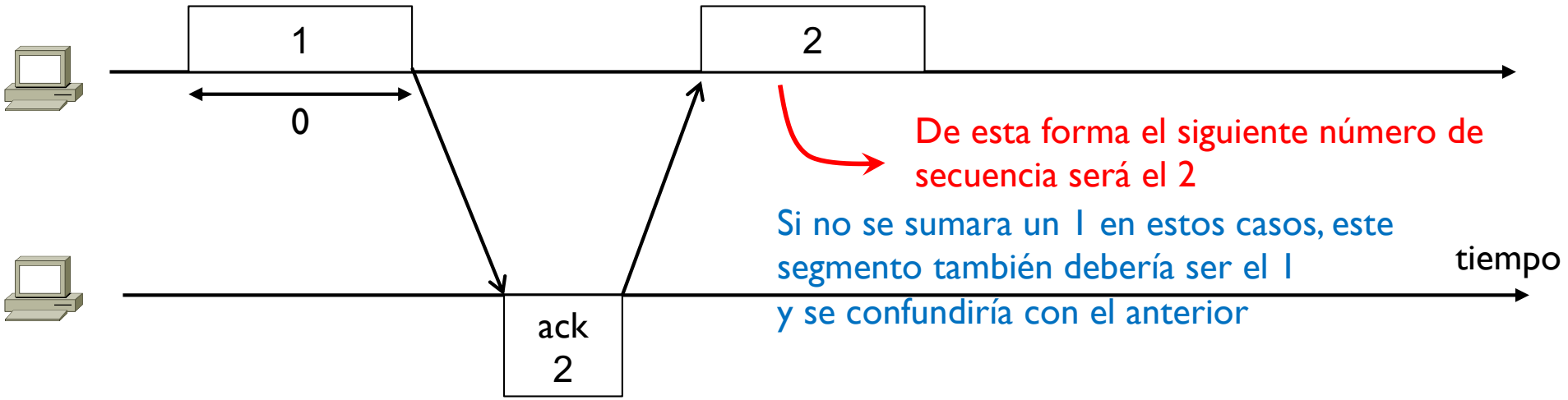
Tema 3 – Números de secuencia

- Caso particular
  - Hay segmentos especiales que sirven al TCP para establecer la conexión y para cerrarla
  - En estos casos, se suma 1 al ack



Tema 3 – Números de secuencia

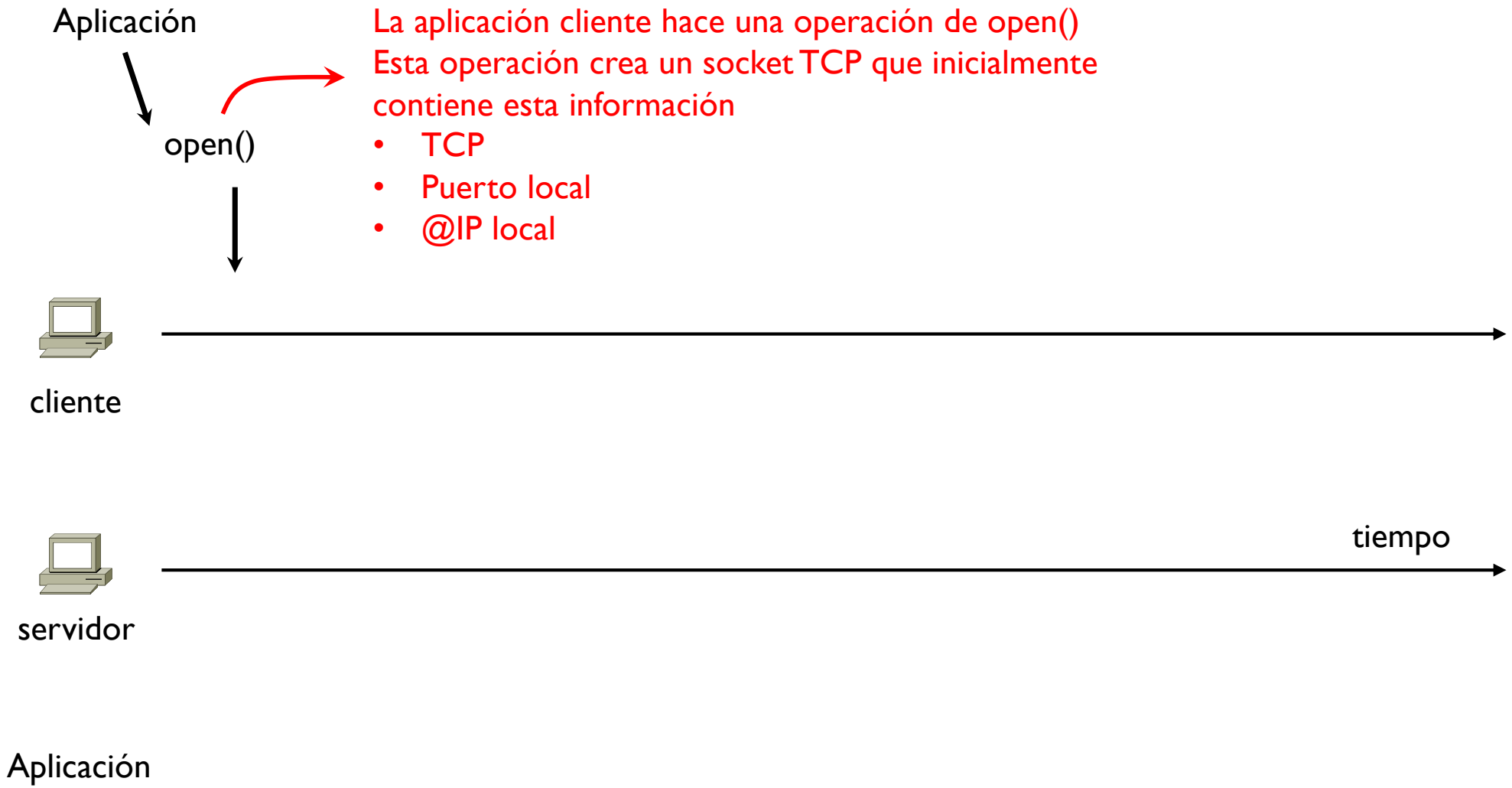
- Caso particular
  - Hay segmentos especiales que sirven al TCP para establecer la conexión y para cerrarla
  - En estos casos, se suma 1 al ack



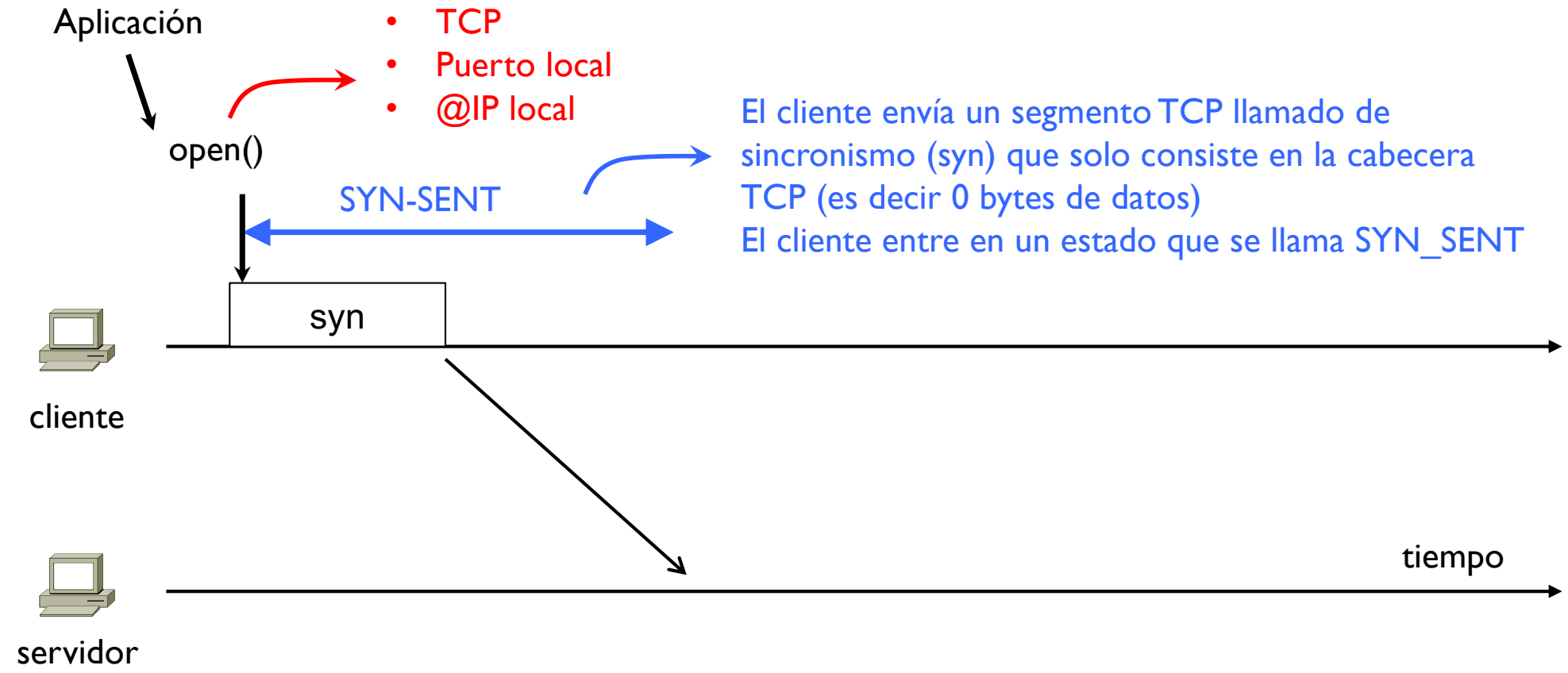
Tema 3 – Establecimiento de una conexión

- TCP está orientado a la conexión
  - Se necesita una fase previa a la transmisión de datos que ponga de acuerdo los dos extremos de la comunicación
  - Se usa un proceso que se llama Three Way Handshaking (3WH o TWH)
    - Apretón de manos en 3 pasos
  - Esta fase la inicia el extremo cliente (el que quiere un servicio) que empieza el 3WH con el extremo servidor (el que proporciona el servicio)

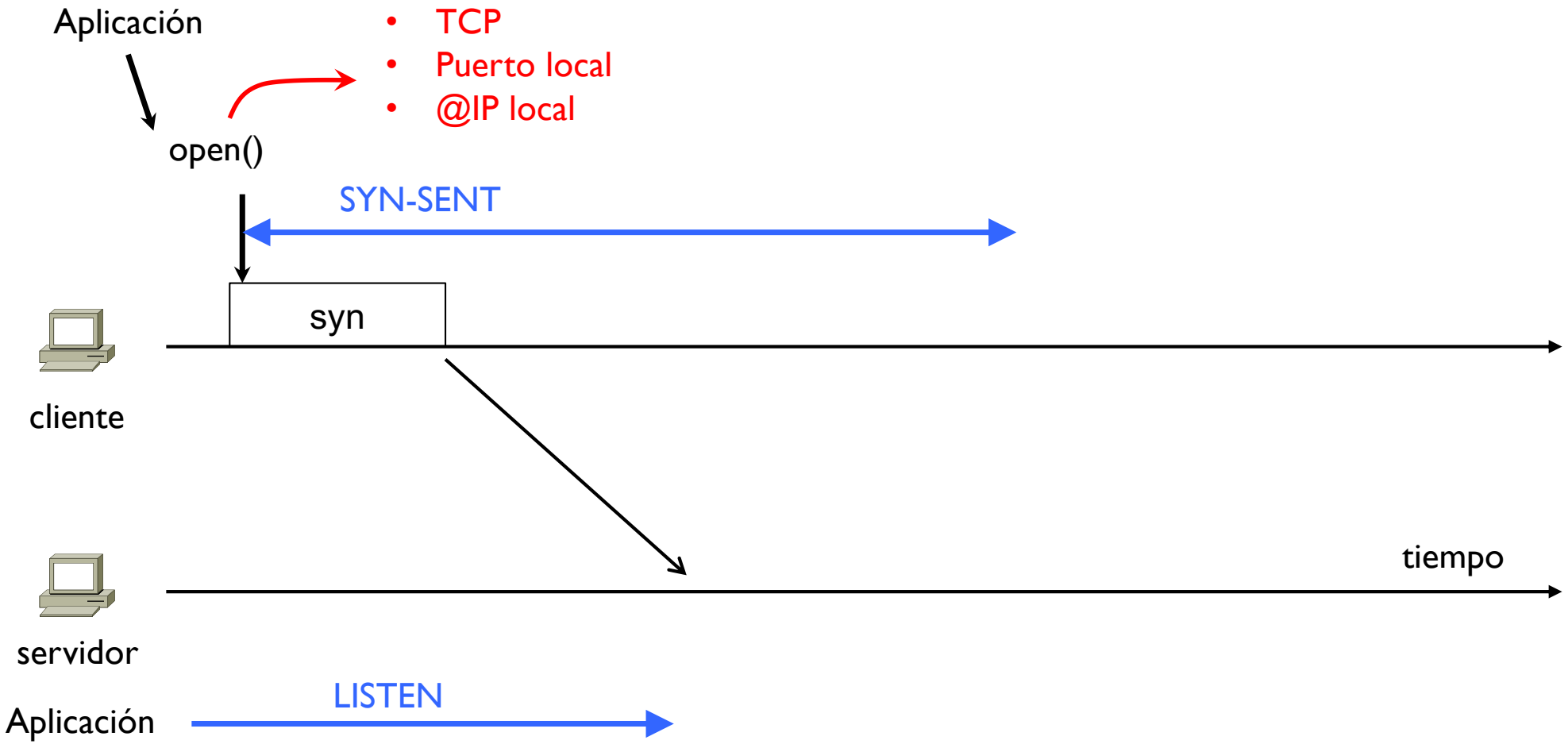
Tema 3 – Establecimiento de una conexión



Tema 3 – Establecimiento de una conexión



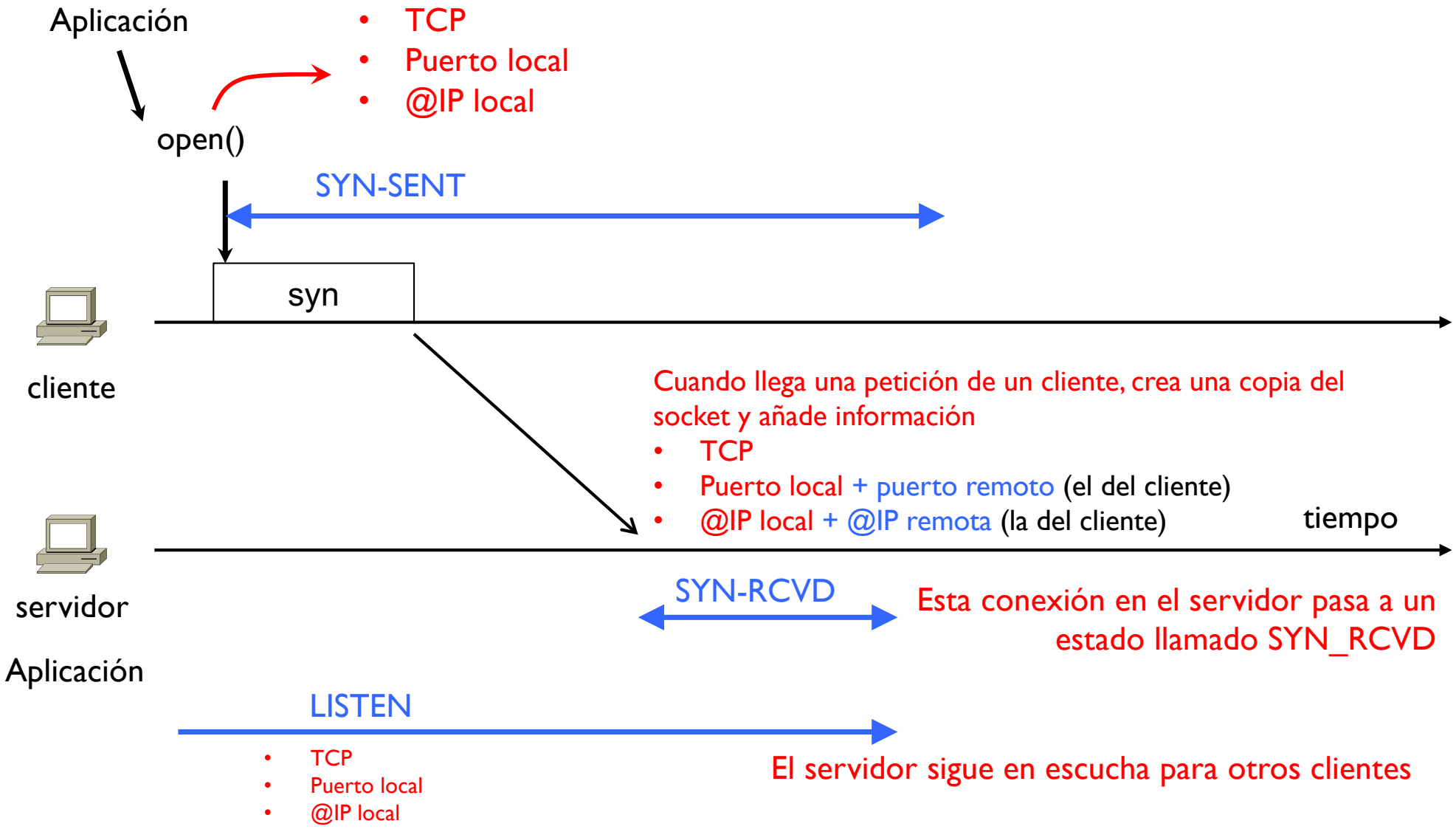
Tema 3 – Establecimiento de una conexión



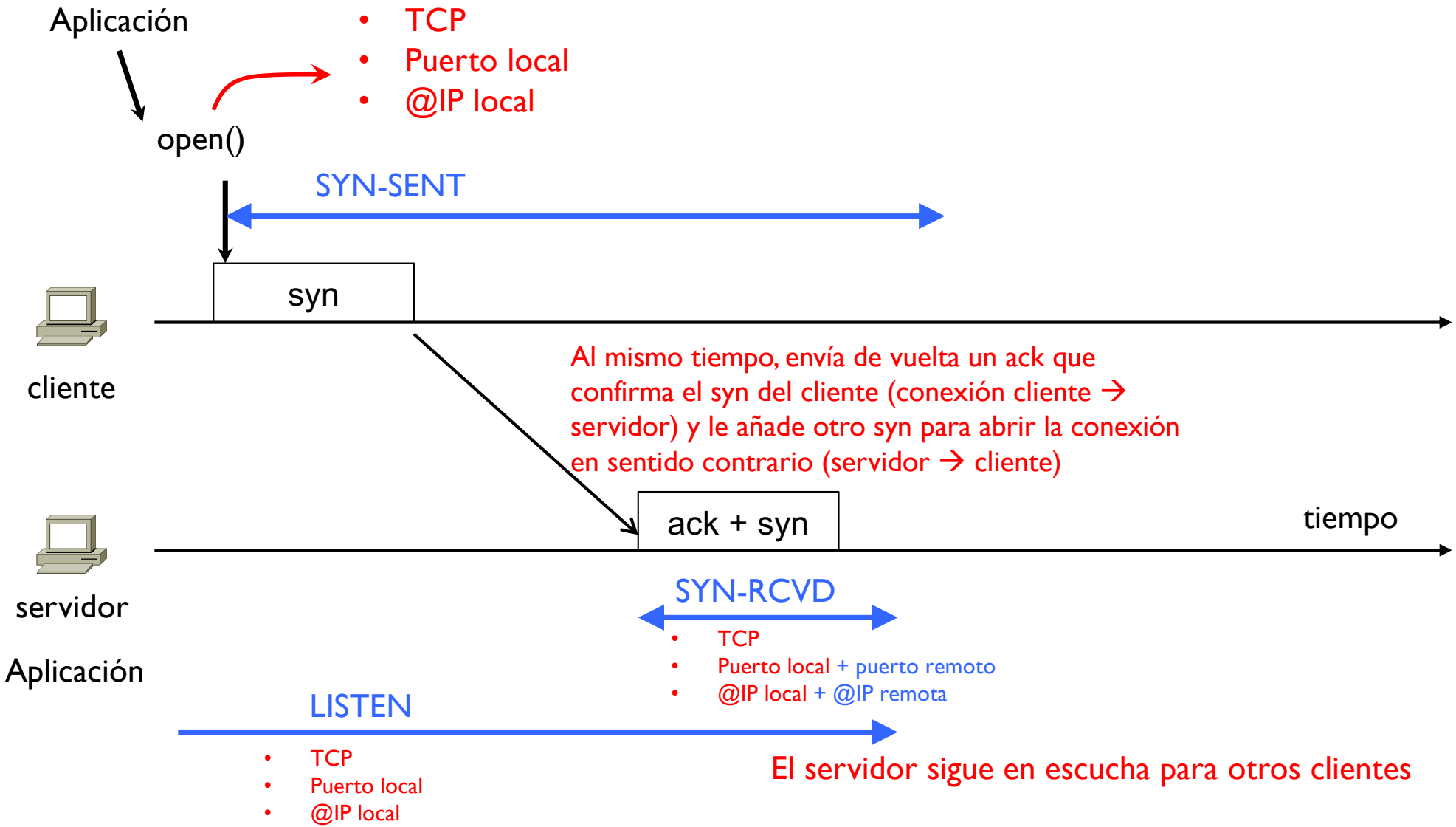
El servidor como tal, tiene ya creado un socket que se dice abierto en escucha
Este socket abierto contiene esta información

- TCP
- Puerto local
- @IP local

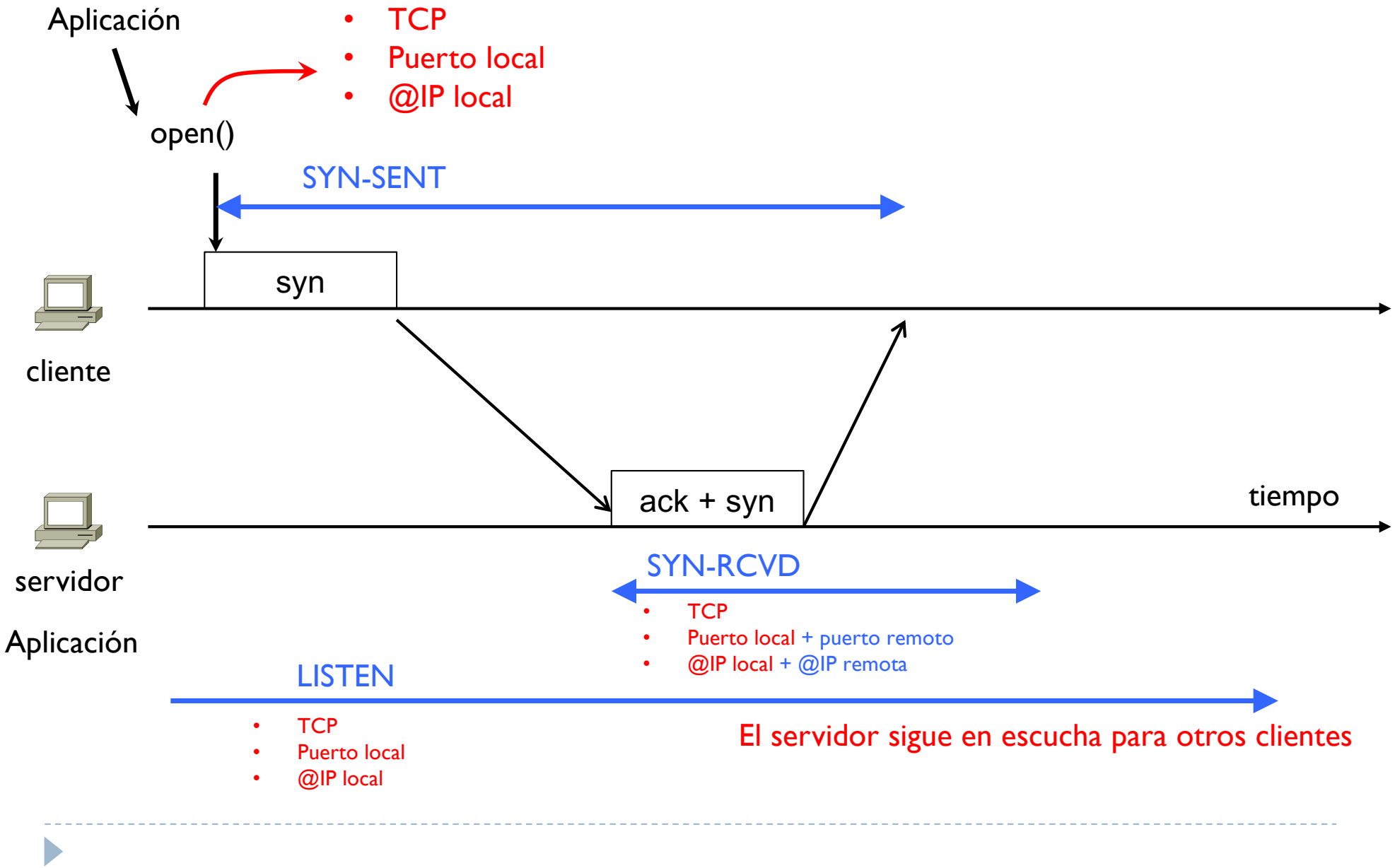
Tema 3 – Establecimiento de una conexión



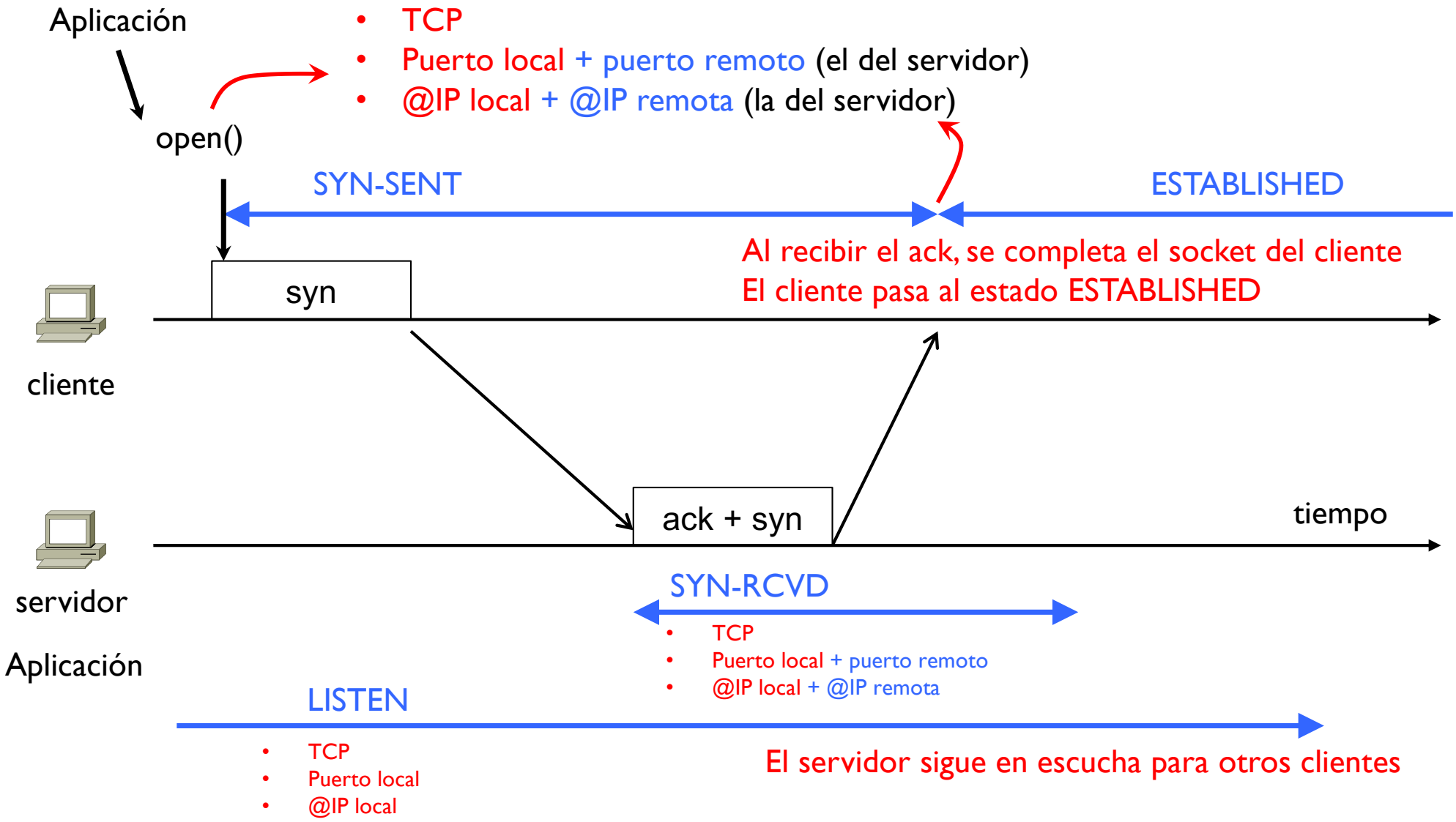
Tema 3 – Establecimiento de una conexión



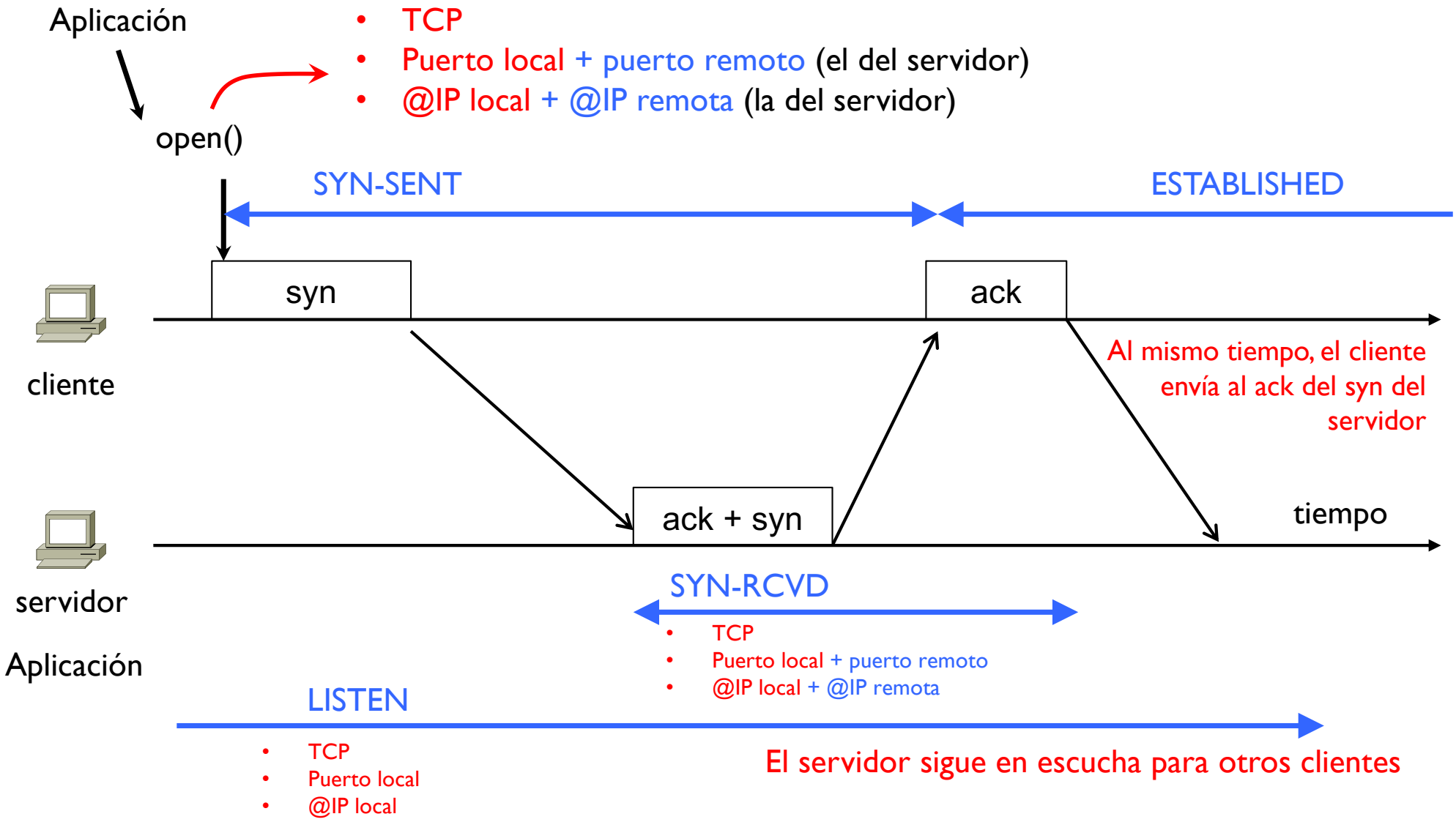
Tema 3 – Establecimiento de una conexión



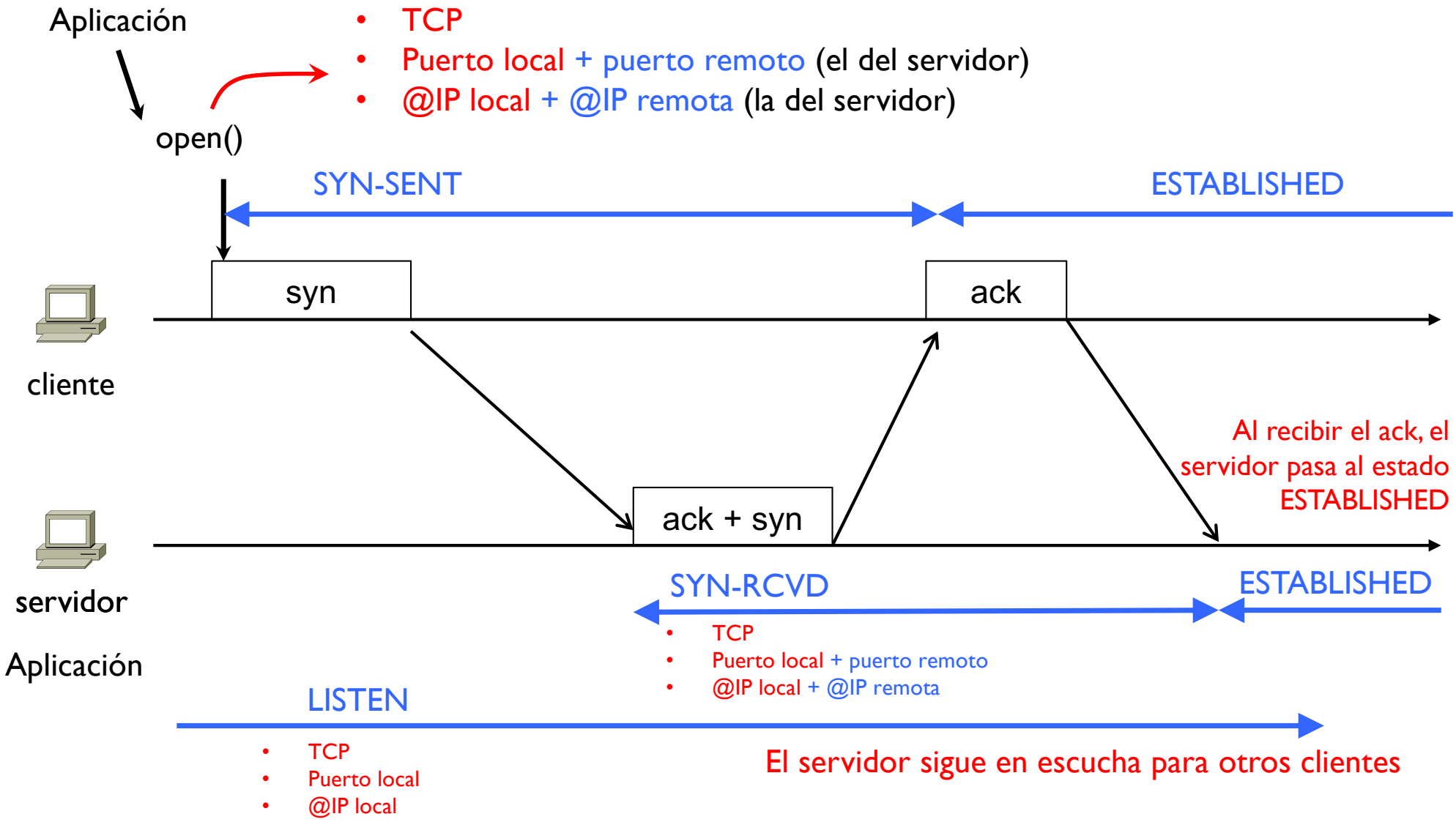
Tema 3 – Establecimiento de una conexión



Tema 3 – Establecimiento de una conexión



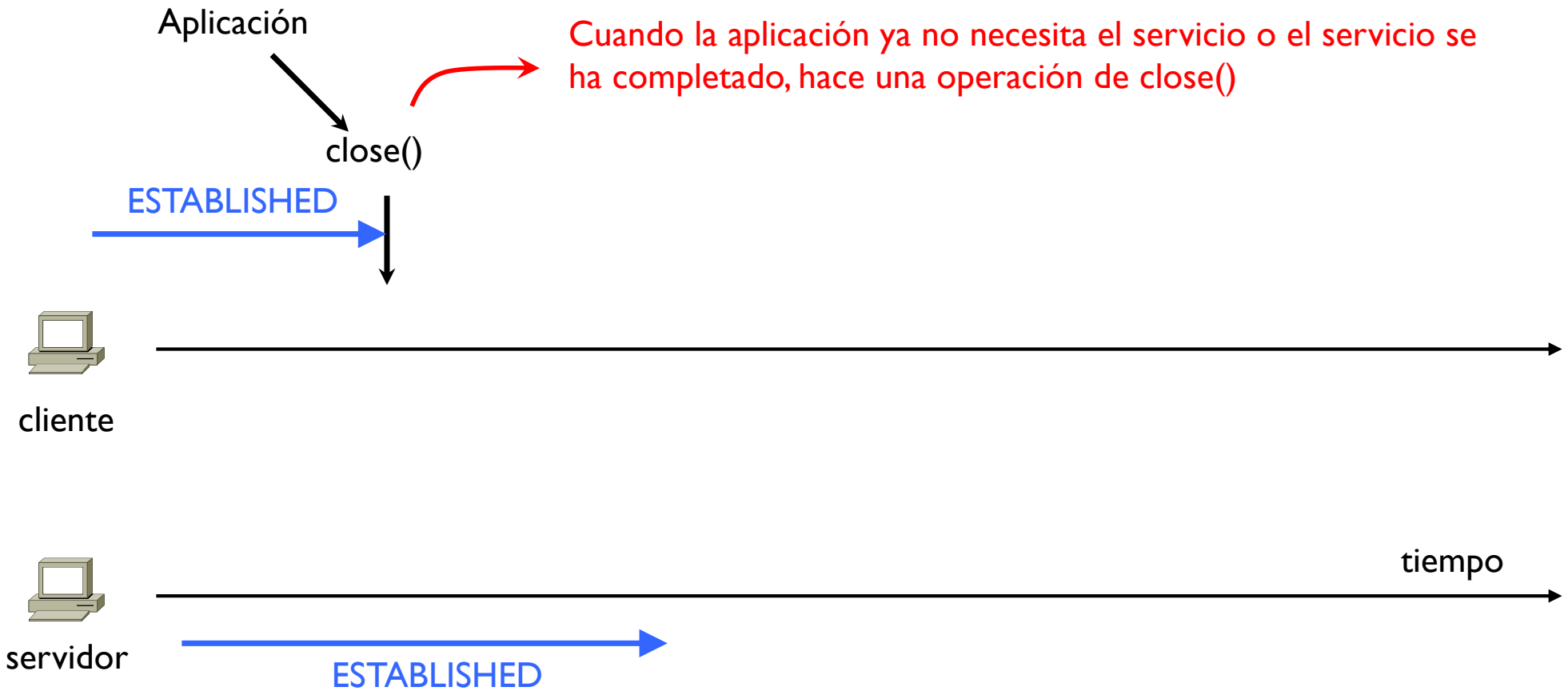
Tema 3 – Establecimiento de una conexión



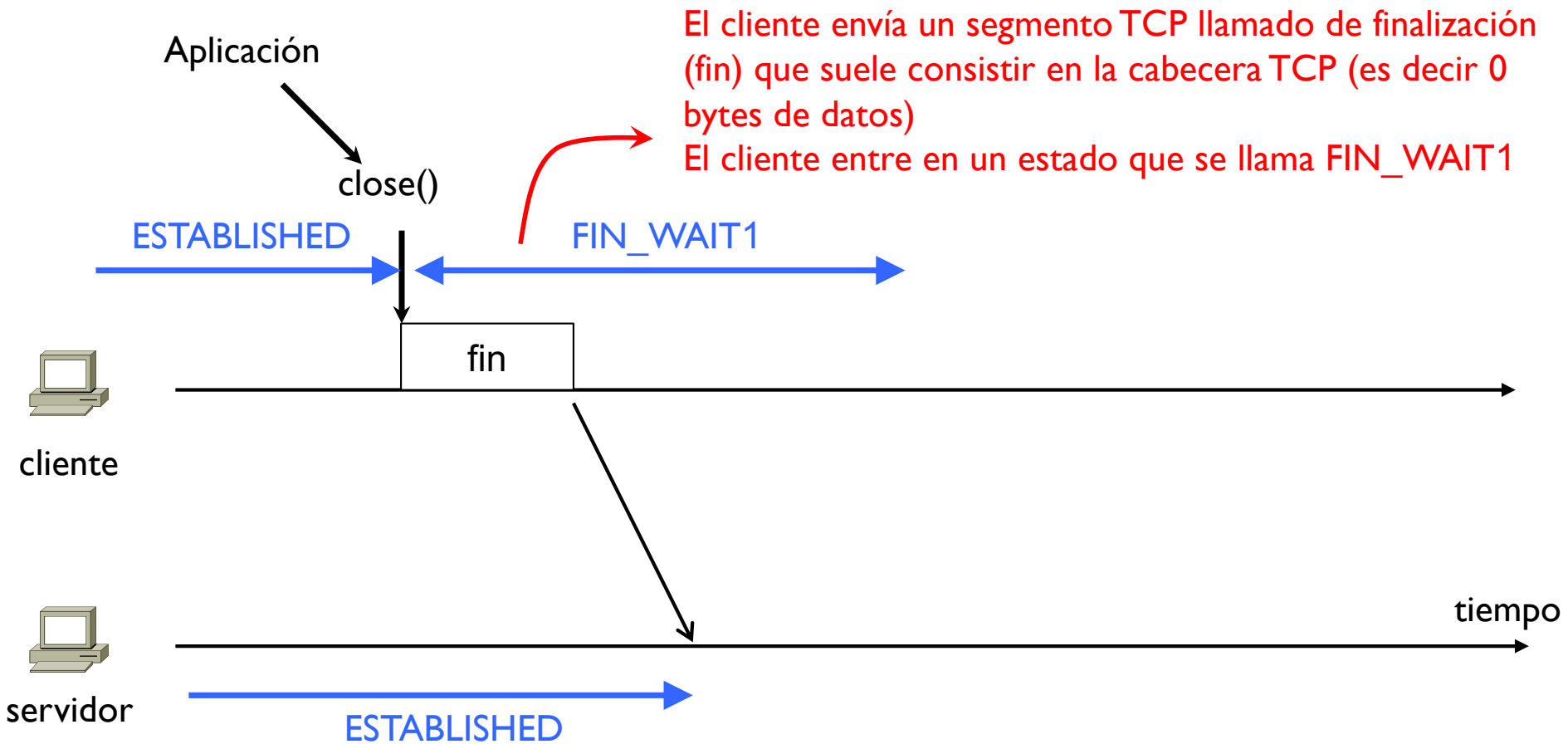
Tema 3 – Terminación de una conexión

- TCP está orientado a la conexión
  - Se necesita una fase previa a la transmisión de datos que ponga de acuerdo los dos extremos de la comunicación
  - Se usa un proceso que se llama Three Way Handshaking (3WH o TWH)
    - Apretón de manos en 3 pasos
  - Esta fase la inicia el extremo cliente (el que quiere un servicio) que empieza el 3WH con el extremo servidor (el que proporciona el servicio)
  - También se necesita una fase de terminación una vez que el servicio ha sido entregado
    - Esta fase la puede empezar cualquiera de los dos extremos

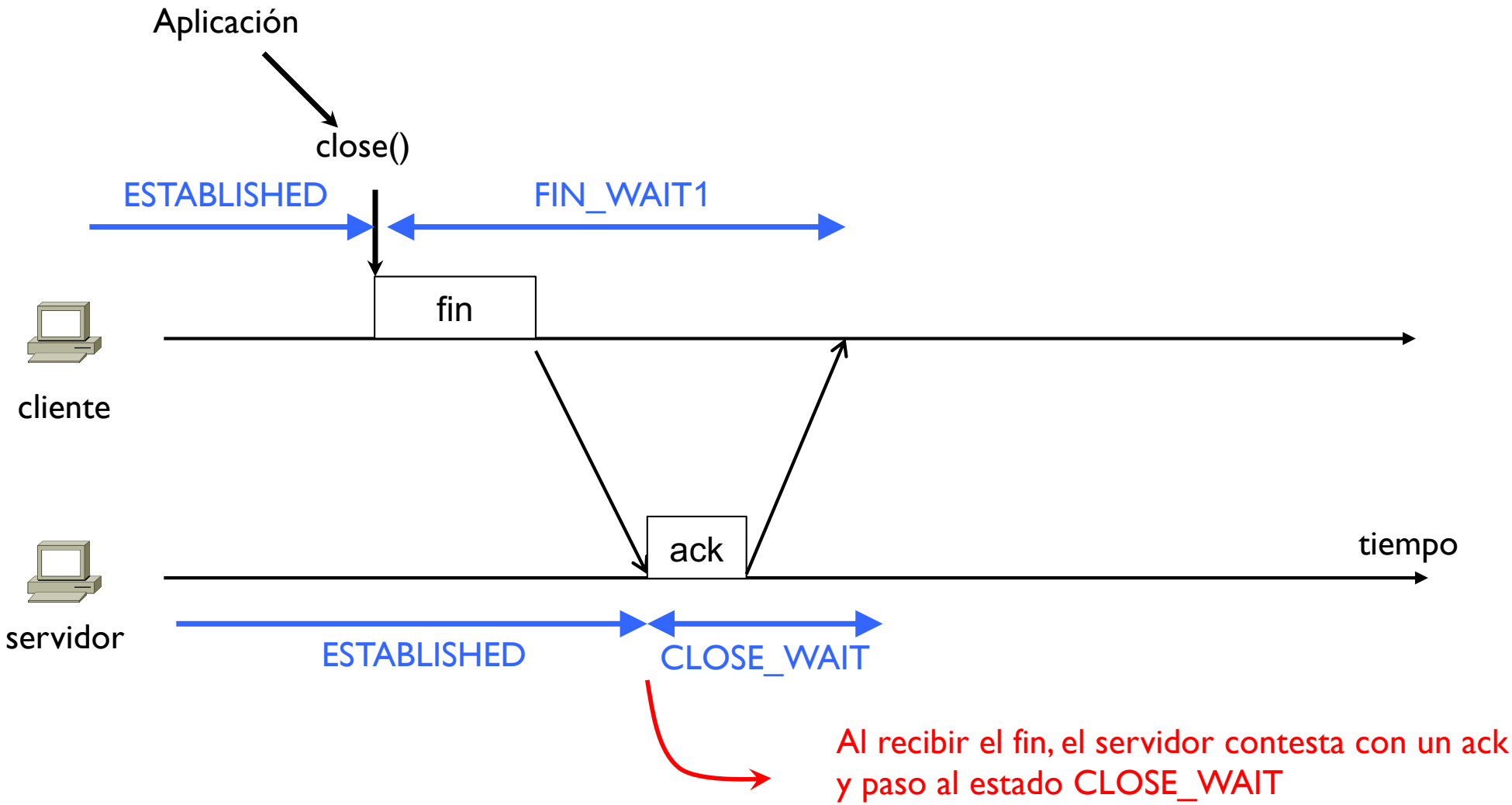
Tema 3 – Terminación de una conexión



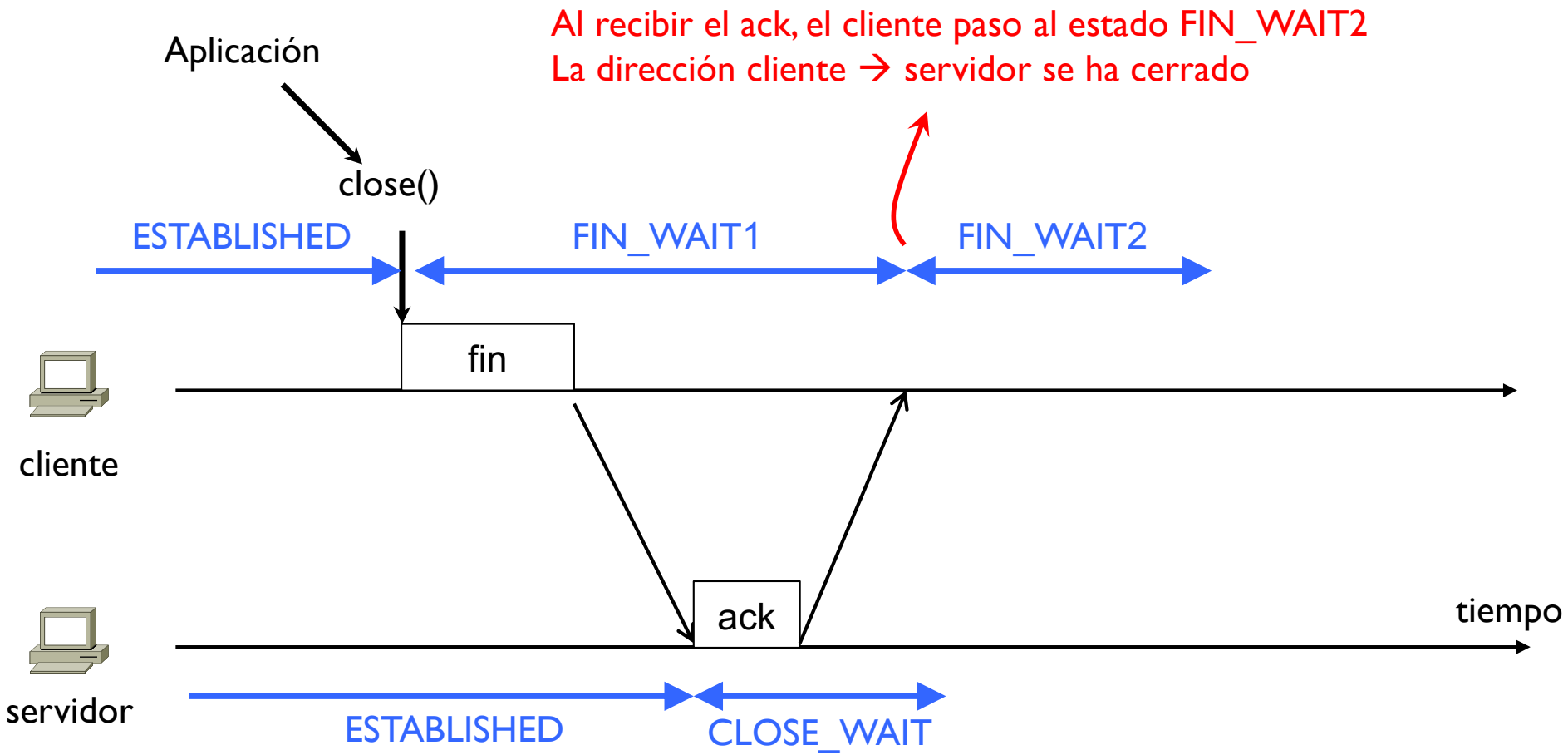
Tema 3 – Terminación de una conexión



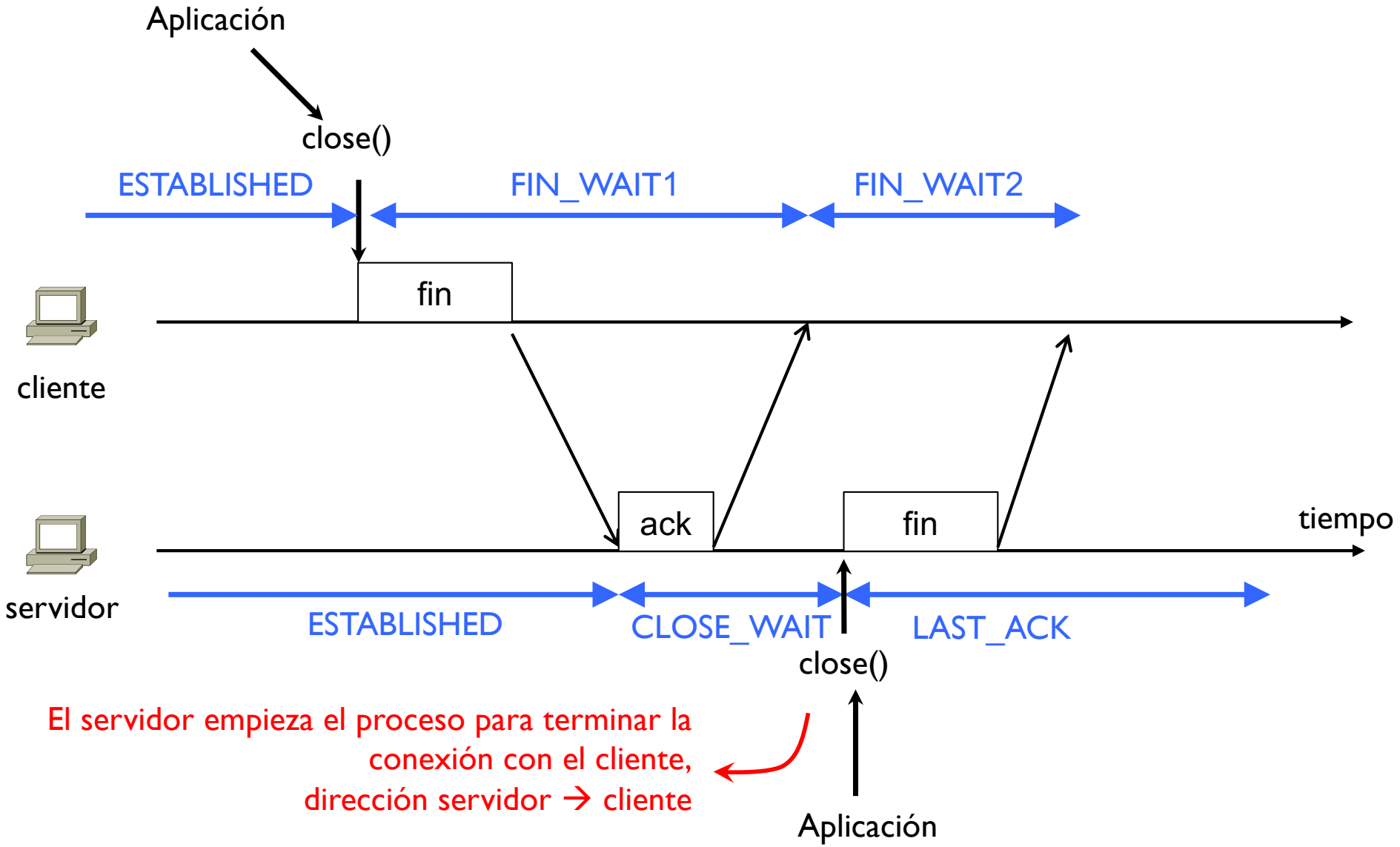
Tema 3 – Terminación de una conexión



Tema 3 – Terminación de una conexión

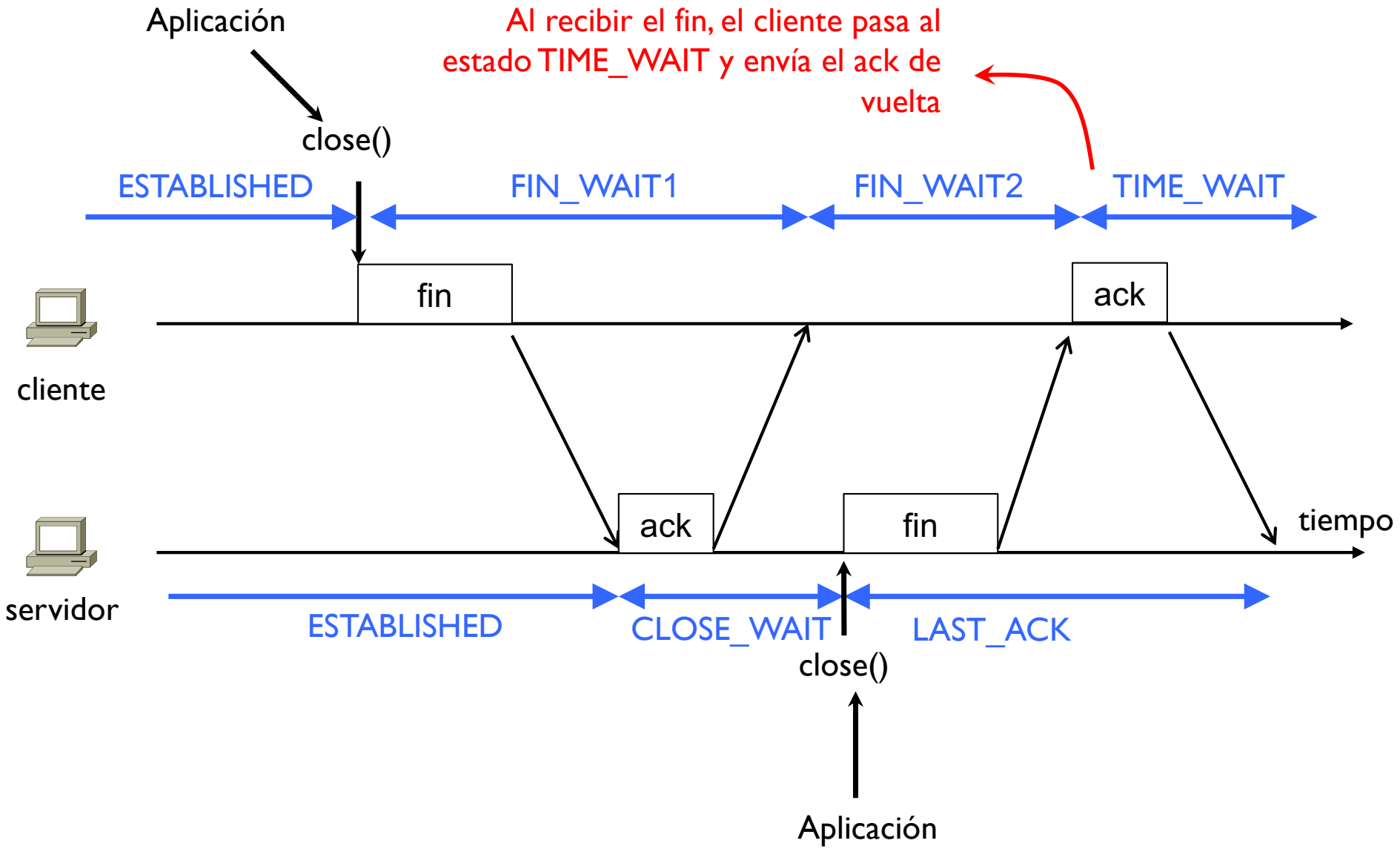


Tema 3 – Terminación de una conexión

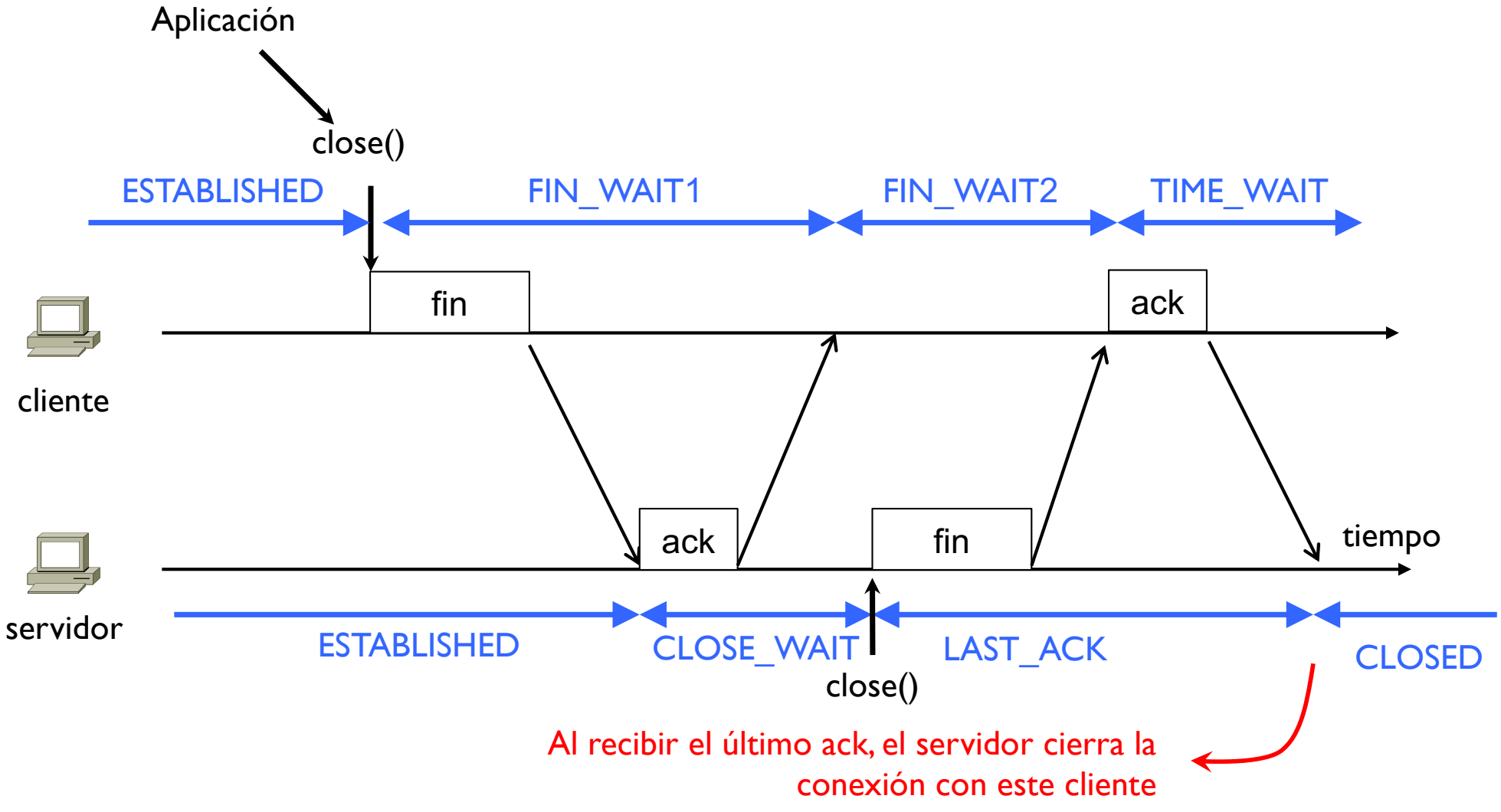


Envía un fin y pasa al estado LAST_ACK

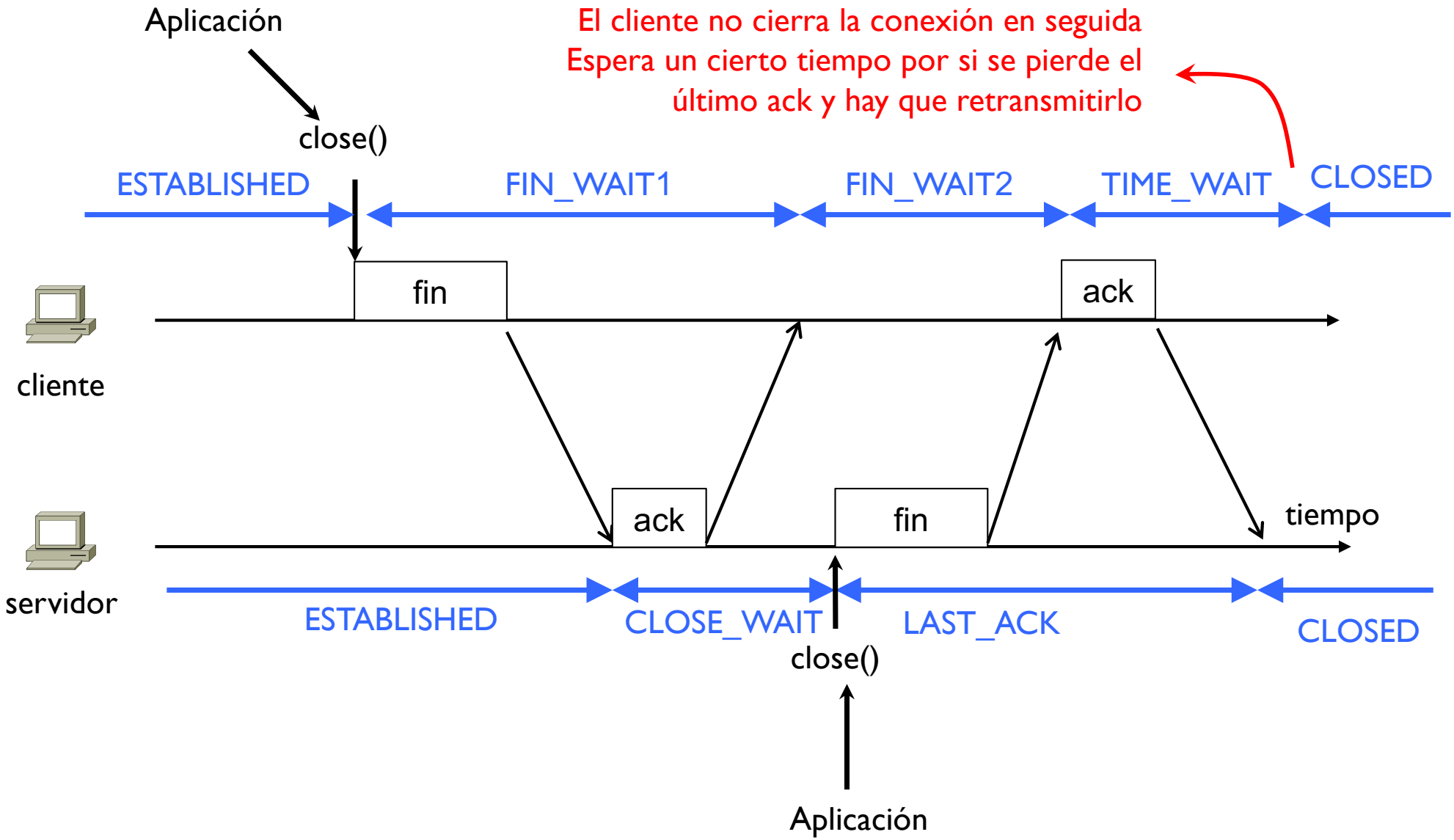
Tema 3 – Terminación de una conexión



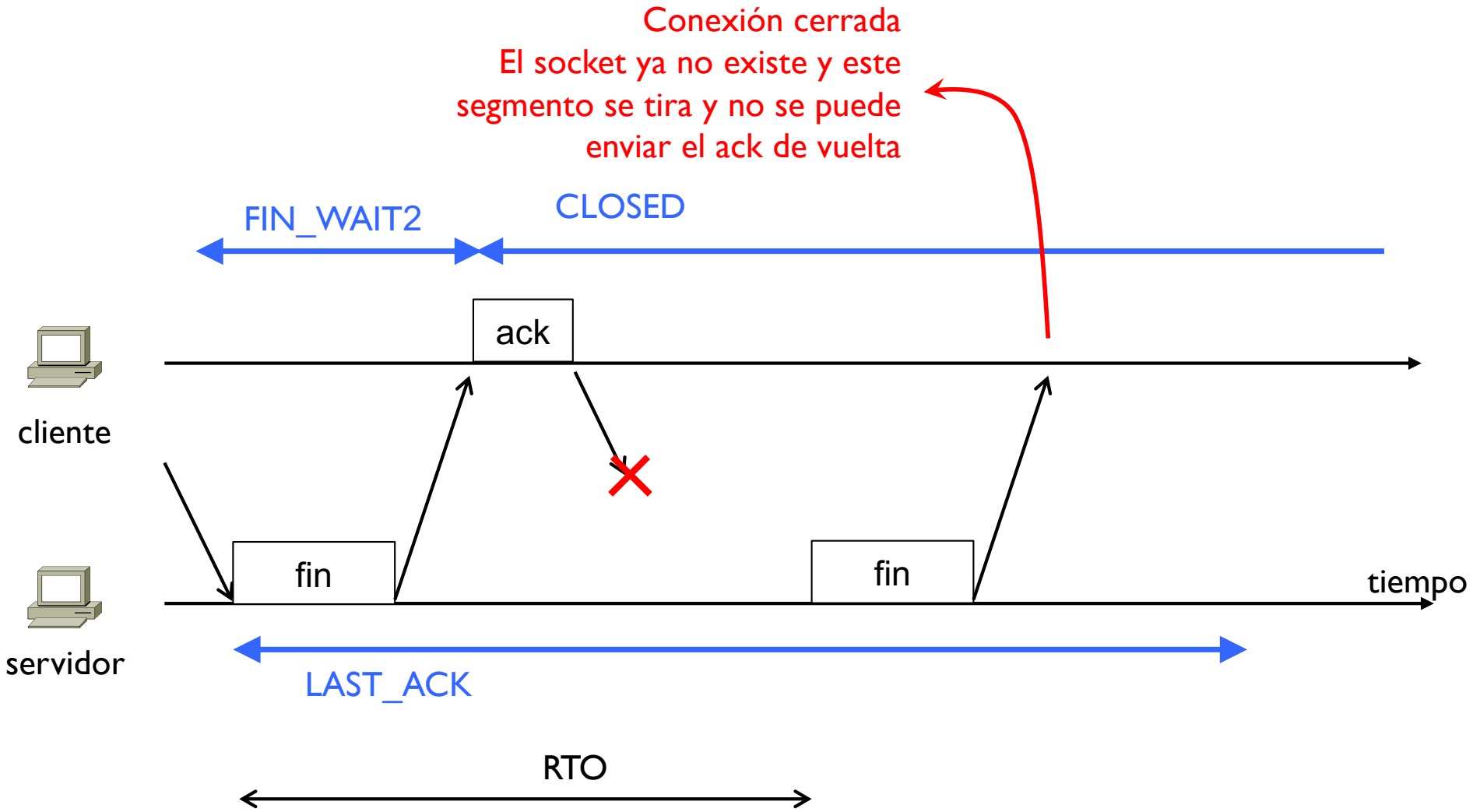
Tema 3 – Terminación de una conexión



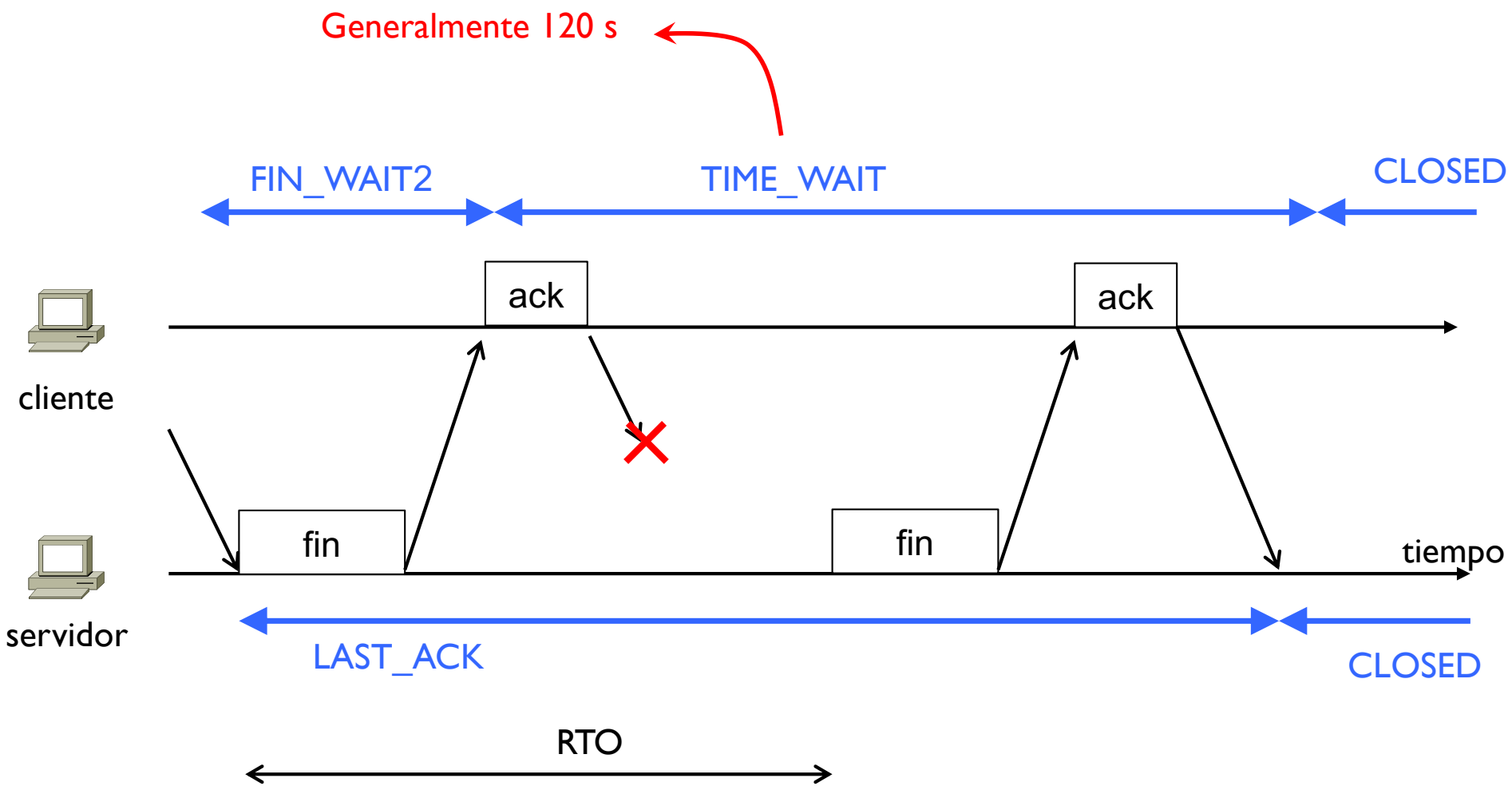
Tema 3 – Terminación de una conexión



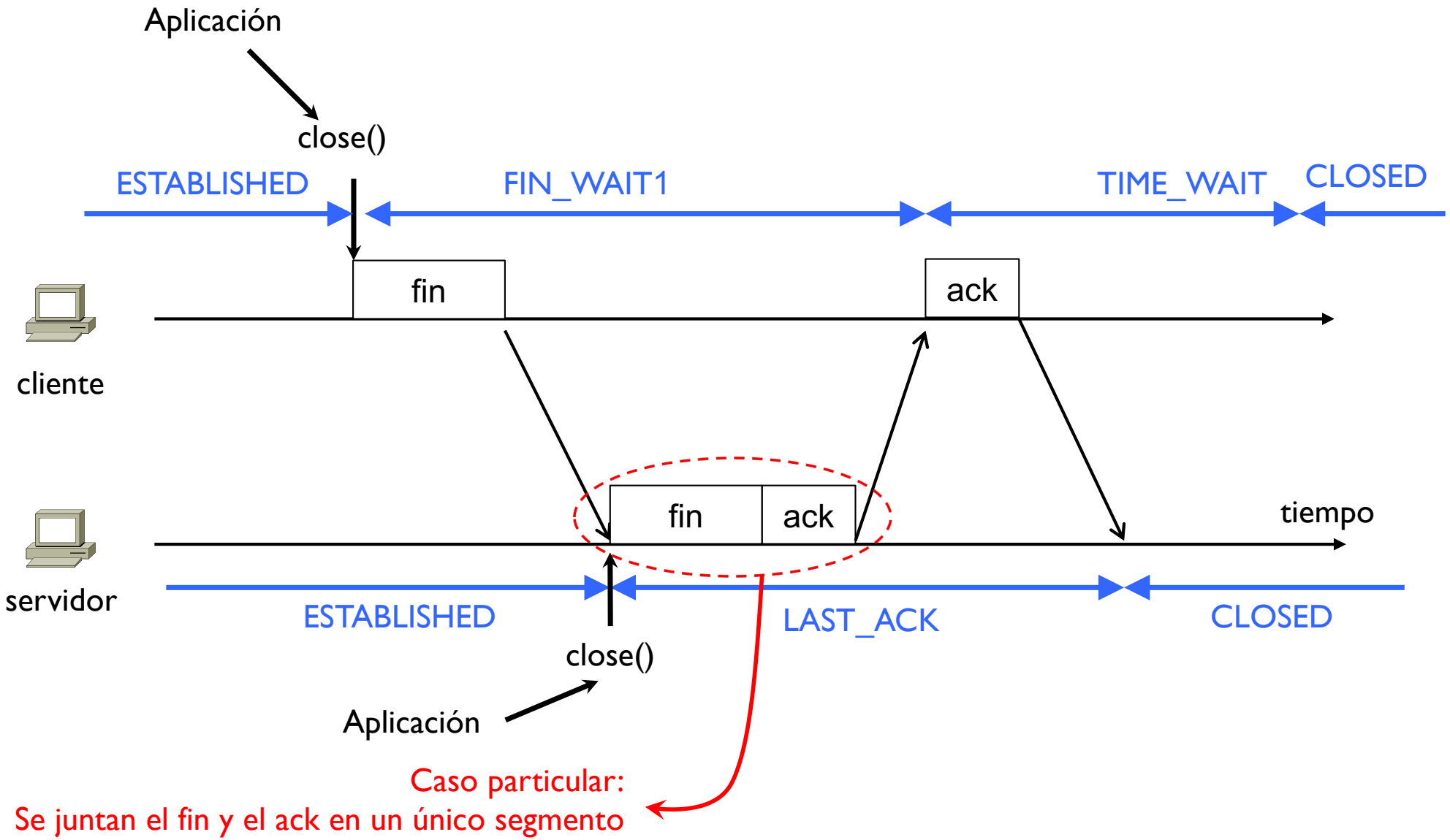
Tema 3 – Terminación de una conexión



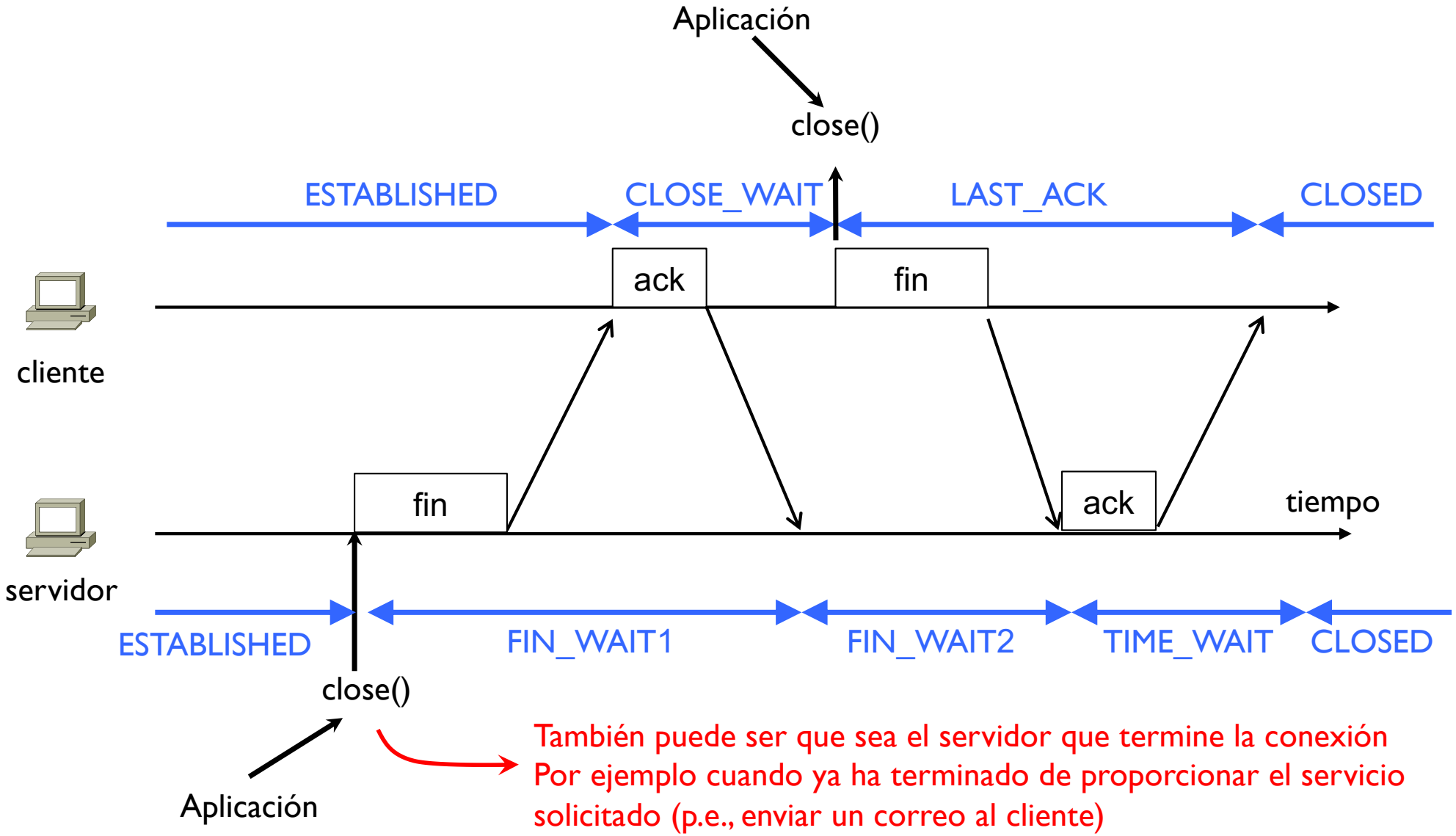
Tema 3 – Terminación de una conexión



Tema 3 – Terminación de una conexión



Tema 3 – Terminación de una conexión



Tema 3 – Ejemplos con netstat

```
$ netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 192.168.1.2:49058       173.255.230.5:80        ESTABLISHED
tcp        0      0 192.168.1.2:33324       173.194.36.117:443      ESTABLISHED
tcp6       0      0 ::1:631                 :::*                    LISTEN
```

```
dync-35-211:~ davidecareglio$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  dync-35-211.ac.u.52838 pclabxc.ac.upc.e.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52837 pclabxc.ac.upc.e.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52826 stackoverflow.co.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52815 104.16.88.254.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52807 185.63.147.10.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52597 mad06s10-in-f6.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52576 mad06s09-in-f6.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52203 91.213.30.155.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52185 mrs04s10-in-f8.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52168 mad06s09-in-f6.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52156 mad01s25-in-f1.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52150 mad01s24-in-f2.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52139 mad06s10-in-f1.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52138 91.213.30.155.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52137 91.213.30.155.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52133 mad06s10-in-f13..https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52128 wo-in-f154.1e100.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52126 mad01s24-in-f2.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52125 mad01s24-in-f2.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52111 91.213.30.155.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52106 91.213.30.155.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52071 ash-rb4-13b.sjc..https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52059 mad01s24-in-f2.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52058 mad01s24-in-f4.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52053 mad01s25-in-f10..https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52044 91.213.30.155.https    ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.52007 104.244.42.8.https     ESTABLISHED
tcp4      37      0  dync-35-211.ac.u.51986 server-54-192-62.https CLOSE_WAIT
tcp4      37      0  dync-35-211.ac.u.51970 108.160.172.193.https  CLOSE_WAIT
tcp4       0      0  dync-35-211.ac.u.51946 mad01s24-in-f3.1.https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.51860 ec2-52-22-240-14.https CLOSE_WAIT
tcp4       0      0  dync-35-211.ac.u.51802 lis01s14-in-f14..https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.51801 91.213.30.155.https    ESTABLISHED
tcp4      37      0  dync-35-211.ac.u.51736 server-54-192-62.https CLOSE_WAIT
tcp4       0      0  dync-35-211.ac.u.51726 114.255.178.107..https ESTABLISHED
tcp4       0      0  dync-35-211.ac.u.51723 91.213.30.155.https    ESTABLISHED
```

Tema 3 – Tools

- Hay varias herramientas que permiten capturar y analizar el intercambio de información durante una comunicación
- Las más usadas son:
  - tcpdump es una herramienta (command-line) para capturar información usada en linux y macOS
  - windump es la versión para Windows
  - Wireshark es una herramienta grafica disponible para los tres OS

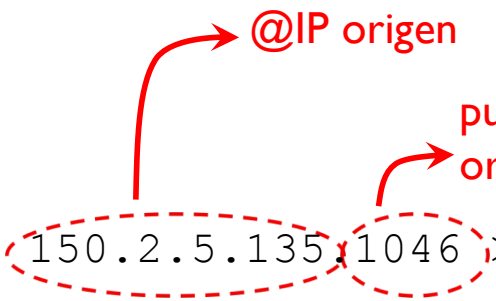
Tema 3 – tcpdump

- Ejemplo

```
tcpdump –ni Ethernet0

150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 …
```

Tema 3 – tcpdump

- Ejemplo




```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 …
```

Tema 3 – tcpdump

- Ejemplo

@IP origen

puerto origen

```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 …
```

@IP: 150.2.5.135
Puerto: 1046 → cliente

Tema 3 – tcpdump

- Ejemplo

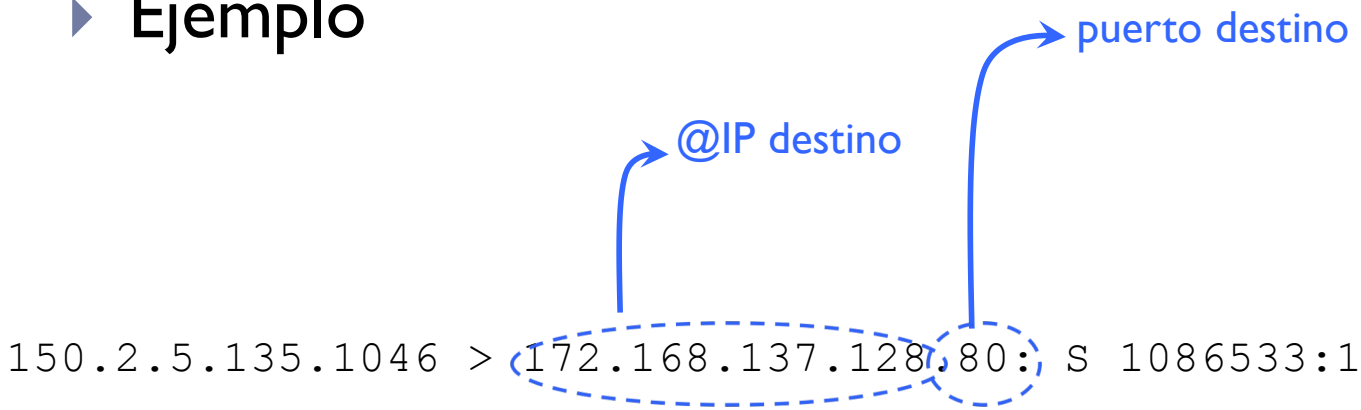
Está enviando

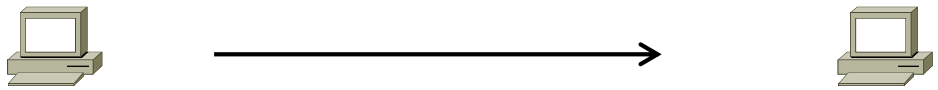
```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 …
```

@IP: 150.2.5.135
Puerto: 1046 → cliente

Tema 3 – tcpdump

- Ejemplo



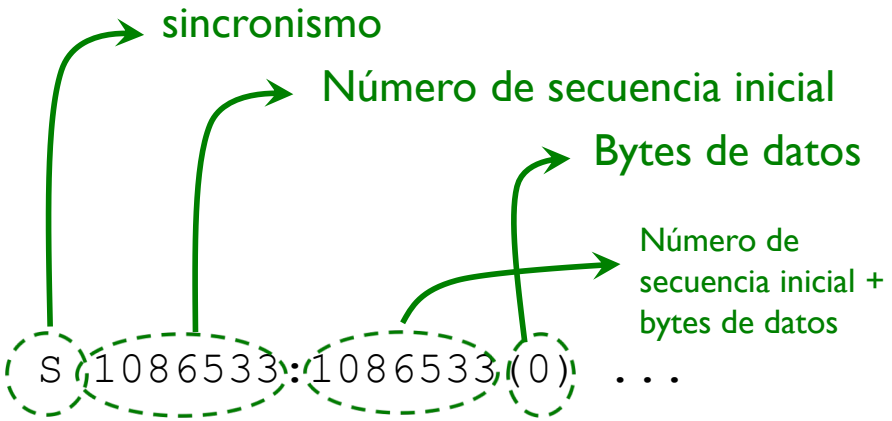


@IP: 150.2.5.135
Puerto: 1046 → cliente

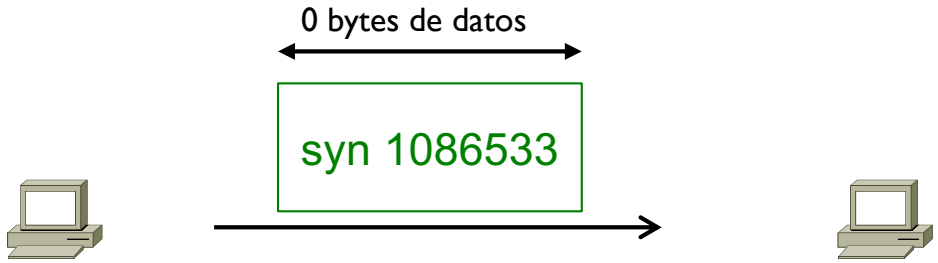
@IP: 172.168.137.128
Puerto: 80 → servidor HTTP

Tema 3 – tcpdump

- Ejemplo




```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 …
```



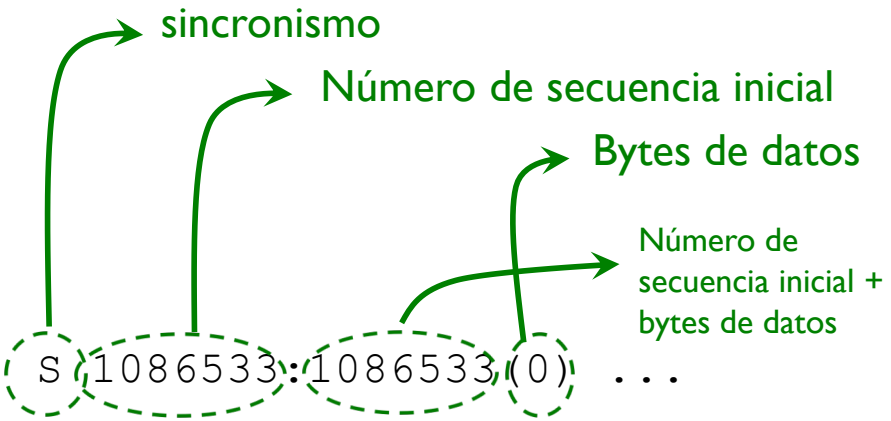
@IP: 150.2.5.135
Puerto: 1046 → cliente

@IP: 172.168.137.128
Puerto: 80 → servidor HTTP

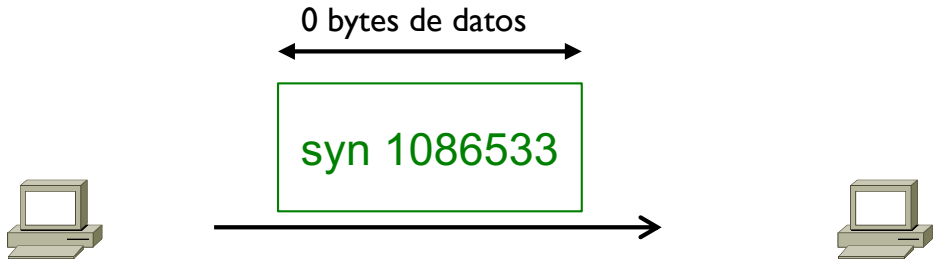
Tema 3 – tcpdump

- Ejemplo

Primer paso del 3WH




```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 …
```



@IP: 150.2.5.135
Puerto: 1046 → cliente

@IP: 172.168.137.128
Puerto: 80 → servidor HTTP

Tema 3 – Protocolo TCP

- El número de secuencia inicial (ISN) del TCP es un número aleatorio entre 0 y 2^{32}-1
- Se usa un número aleatorio por razones de seguridad
  - Si siempre se empezara por 0, sería relativamente fácil hacerse pasar por el otro extremo contestando siempre con un ack 1 o bloquear cualquier comunicación enviando constantemente ack 1

Tema 3 – tcpdump

- Ejemplo

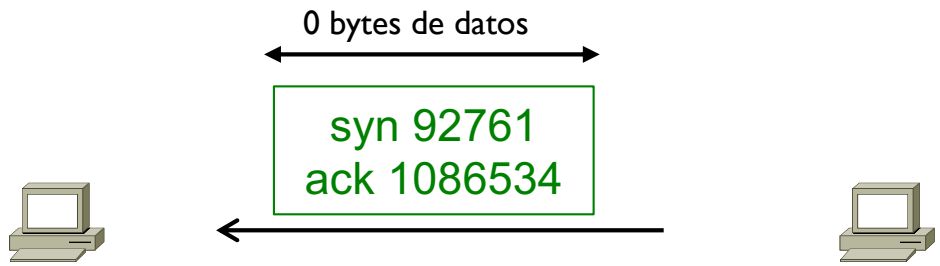
Segundo paso del 3WH

```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...
172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...
150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 …
```

sincronismo

Número de secuencia inicial para la comunicación en el otro sentido

ack del syn del 1er paso
Es el número de secuencia + 1 ya que lleva 0 bytes de datos



@IP: 150.2.5.135
Puerto: 1046 → cliente

@IP: 172.168.137.128
Puerto: 80 → servidor HTTP

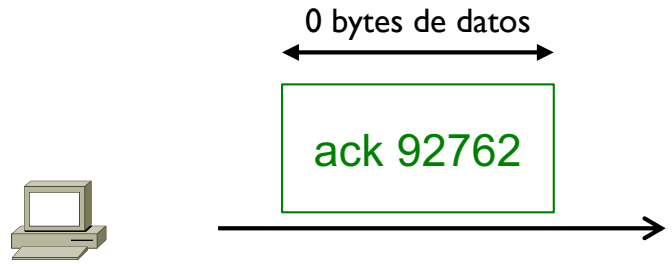
Tema 3 – tcpdump

- Ejemplo

Tercer paso del 3WH

```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 ...
```

ack del syn del 2o paso
Es el número de secuencia del syn del servidor + 1



@IP: 150.2.5.135
Puerto: 1046 → cliente

@IP: 172.168.137.128
Puerto: 80 → servidor HTTP

Tema 3 – Tools

- En muchas implementaciones de tcpdump, hay una simplificación para analizar con más facilidad una captura de intercambio de información
- Este simplificación consiste en visualizar números de secuencias relativos, es decir números de secuencias a los cuales se ha restado el número de secuencia inicial
- De esta forma, el número de secuencia inicial relativo es siempre 0 para ambas direcciones y el primer número de secuencia relativo que se usa para transmitir datos es siempre 1

- IMPORTANTE
  - Es una simplificación del tcpdump para facilitar la lectura de una traza
  - TCP envía el número real

Tema 3 – tcpdump

- Por ejemplo, siguiendo con el ejemplo anterior, una vez pasado el 3WH, empezaría el envío de datos

```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 ...

150.2.5.135.1046 > 172.168.137.128.80: . 1:351(350) ...

172.168.137.128.80 > 150.2.5.135.1046: ack 351 ...

172.168.137.128.80 > 150.2.5.135.1046: . 1:1461(1460) ...
```

Este valor es en realidad 1086534 (1 + el número de secuencia inicial del cliente)

Es el cliente que transmite, se resta su número de secuencia inicial

Tema 3 – tcpdump

- Por ejemplo, siguiendo con el ejemplo anterior, una vez pasado el 3WH, empezaría el envío de datos

```
150.2.5.135.1046 > 172.168.137.128.80: S 1086533:1086533(0) ...

172.168.137.128.80 > 150.2.5.135.1046: S 92761:92761(0) ack 1086534 ...

150.2.5.135.1046 > 172.168.137.128.80: . ack 92762 ...

150.2.5.135.1046 > 172.168.137.128.80: . 1:351(350) ...

172.168.137.128.80 > 150.2.5.135.1046: ack 351 ...

172.168.137.128.80 > 150.2.5.135.1046: . 1:1461(1460) ...
```

Este valor es en realidad 1086534 (1 + el número de secuencia inicial del cliente)

Es el servidor que transmite, se resta su número de secuencia inicial

Este valor es en realidad 92762 (1 + el número de secuencia inicial del servidor)

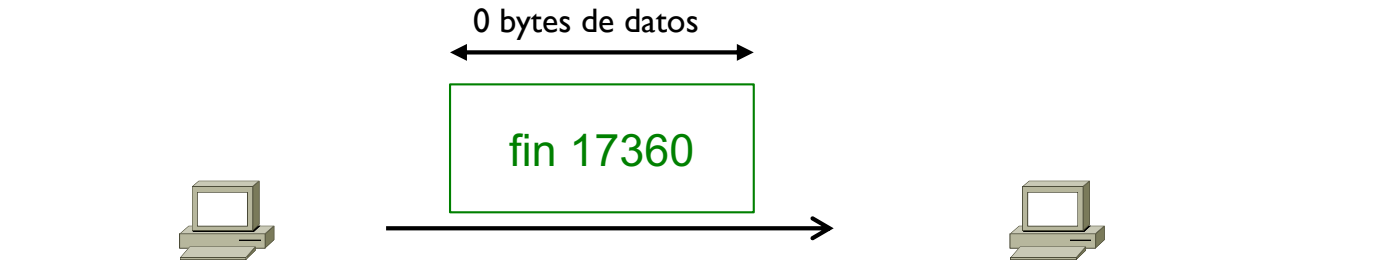
Tema 3 – tcpdump

- Ejemplo

terminación

Número de secuencia final

```
150.2.5.135.1046 > 172.168.137.128.80: F 17360:17360(0) ...

172.168.137.128.80 > 150.2.5.135.1046: . ack 17361 ...

172.168.137.128.80 > 150.2.5.135.1046: F 4234000:4234000(0) ack 17361 …

150.2.5.135.1046 > 172.168.137.128.80: . ack 4234001 …
```



@IP: 150.2.5.135
Puerto: 1046 → cliente

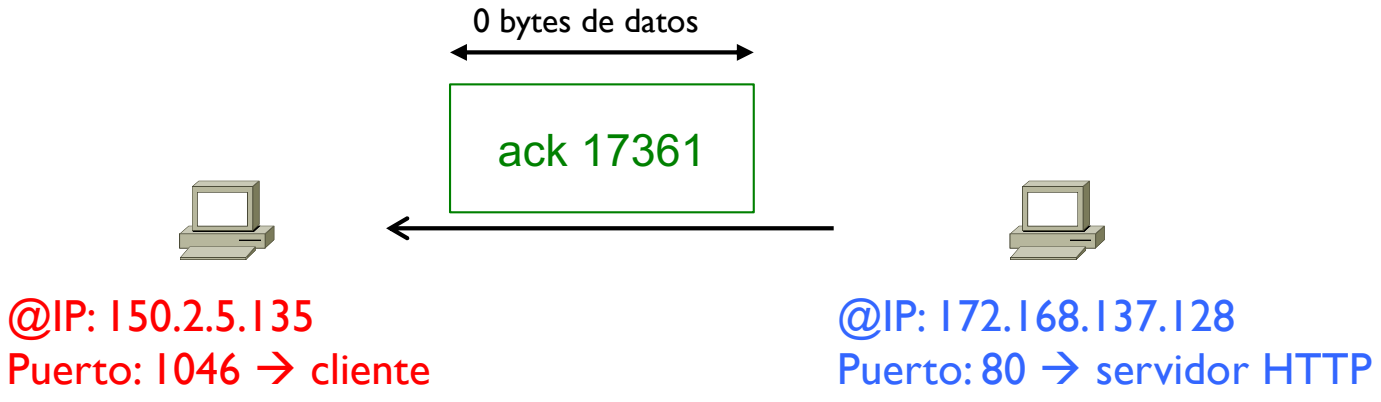
@IP: 172.168.137.128
Puerto: 80 → servidor HTTP

Tema 3 – tcpdump

- Ejemplo

```
150.2.5.135.1046 > 172.168.137.128.80: F 17360:17360(0) ...
172.168.137.128.80 > 150.2.5.135.1046: . ack 17361 ...
172.168.137.128.80 > 150.2.5.135.1046: F 4234000:4234000(0) ack 17361 …
150.2.5.135.1046 > 172.168.137.128.80: . ack 4234001 …
```

ack del servidor al cliente

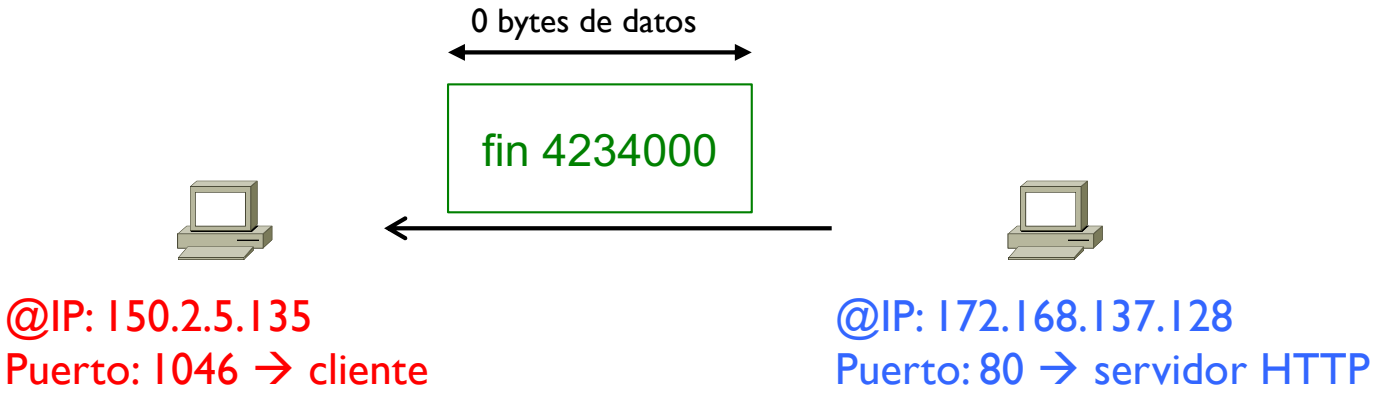


Tema 3 – tcpdump

- Ejemplo

```
150.2.5.135.1046 > 172.168.137.128.80: F 17360:17360(0) ...

172.168.137.128.80 > 150.2.5.135.1046: . ack 17361 ...

172.168.137.128.80 > 150.2.5.135.1046: F 4234000:4234000(0) ack 17361 …

150.2.5.135.1046 > 172.168.137.128.80: . ack 4234001 …
```

Número de secuencia final

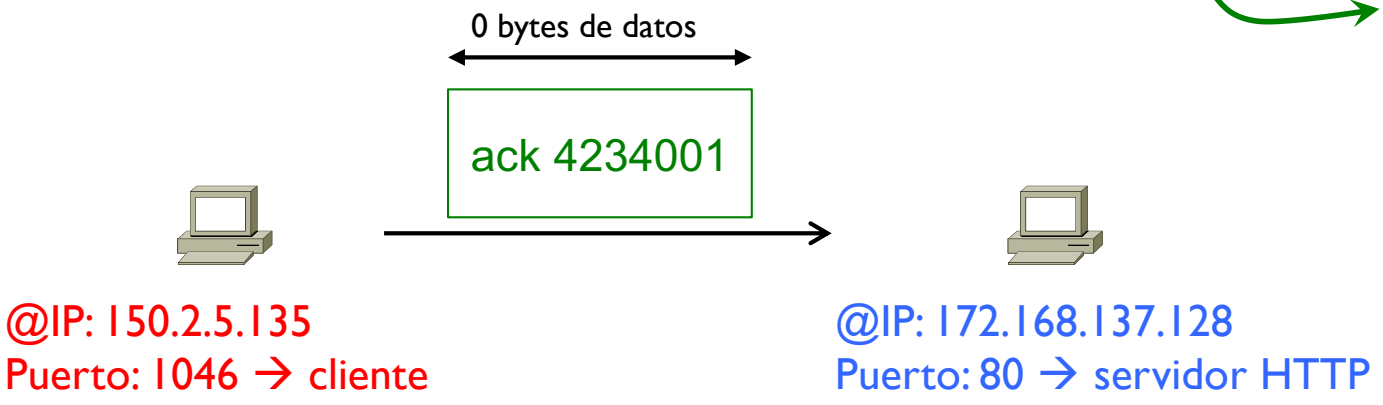


Tema 3 – tcpdump

- Ejemplo

```
150.2.5.135.1046 > 172.168.137.128.80: F 17360:17360(0) ...

172.168.137.128.80 > 150.2.5.135.1046: . ack 17361 ...

172.168.137.128.80 > 150.2.5.135.1046: F 4234000:4234000(0) ack 17361 …

150.2.5.135.1046 > 172.168.137.128.80: . ack 4234001 …
```

ack del cliente al servidor



Tema 3 – tcpdump

- Ejemplo

```
150.2.5.135.1046 > 172.168.137.128.80: F 17360:17360(0) ...

172.168.137.128.80 > 150.2.5.135.1046: . ack 17361 ...

172.168.137.128.80 > 150.2.5.135.1046: F 4234000:4234000(0) ack 17361 …

150.2.5.135.1046 > 172.168.137.128.80: . ack 4234001 …
```

Pregunta:

- ¿cuántos bytes de datos habrá enviado durante toda la conexión el cliente al servidor?
- ¿y cuántos del servidor al cliente?

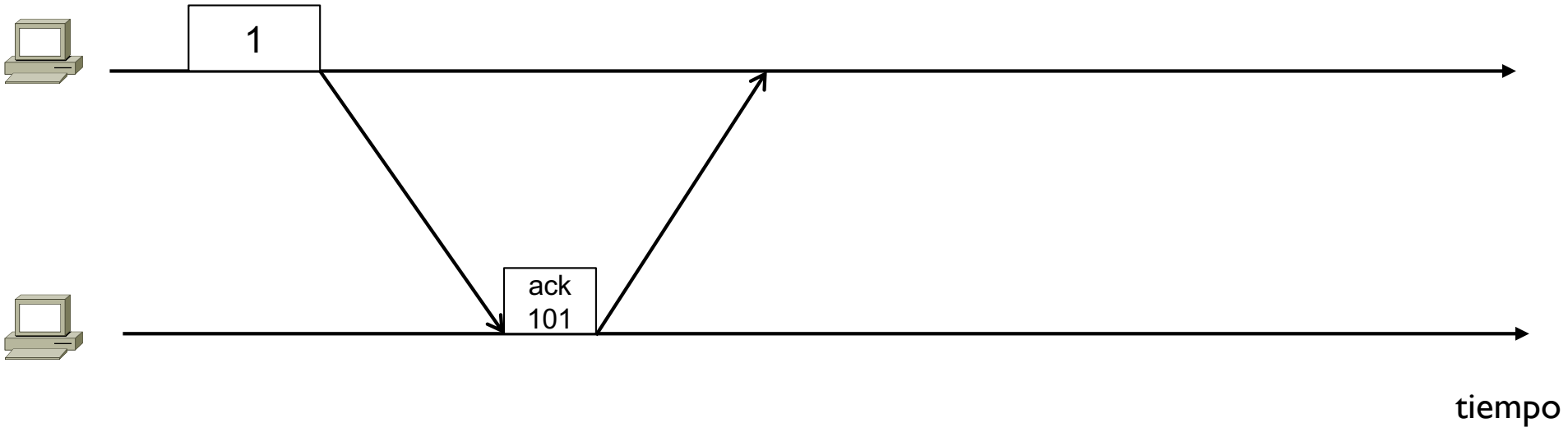
Tema 3 – Protocolos UDP y TCP

- a) Introducción
- b) El protocolo UDP
- **c) El protocolo TCP**
  - Arquitectura
  - El MSS
  - Números de secuencia
  - Establecimiento y terminación de una conexión TCP
  - **Funcionamiento durante la transmisión**
    - Control de flujo
    - Control de congestión
  - Cabecera TCP

Tema 3 – Durante la transmisión

- Se usa una transmisión continua

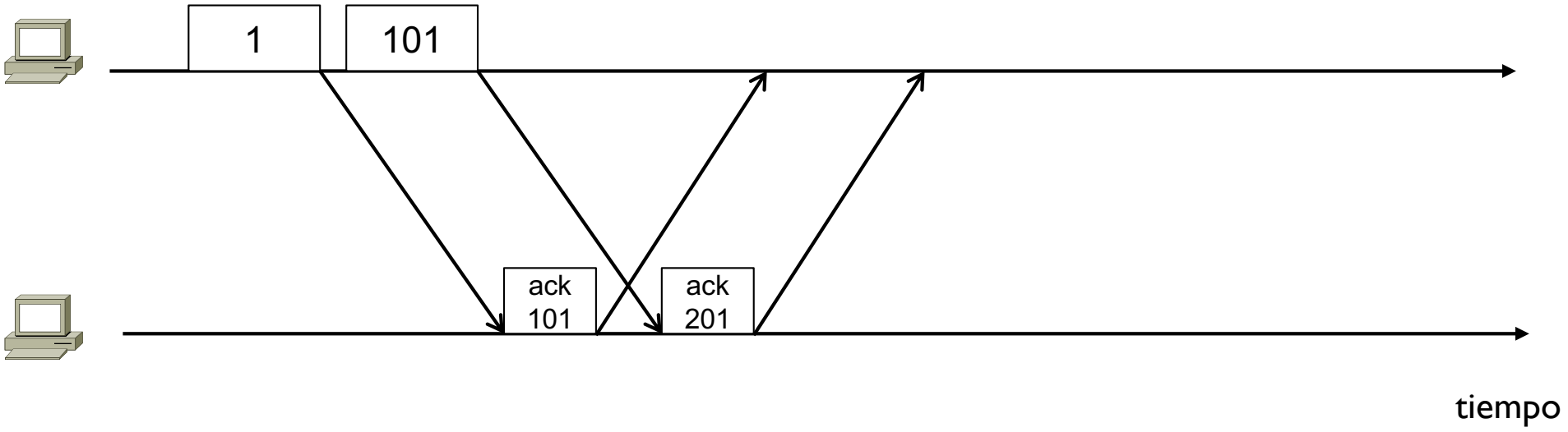
MSS = 100 bytes



Tema 3 – Durante la transmisión

- Se usa una transmisión continua

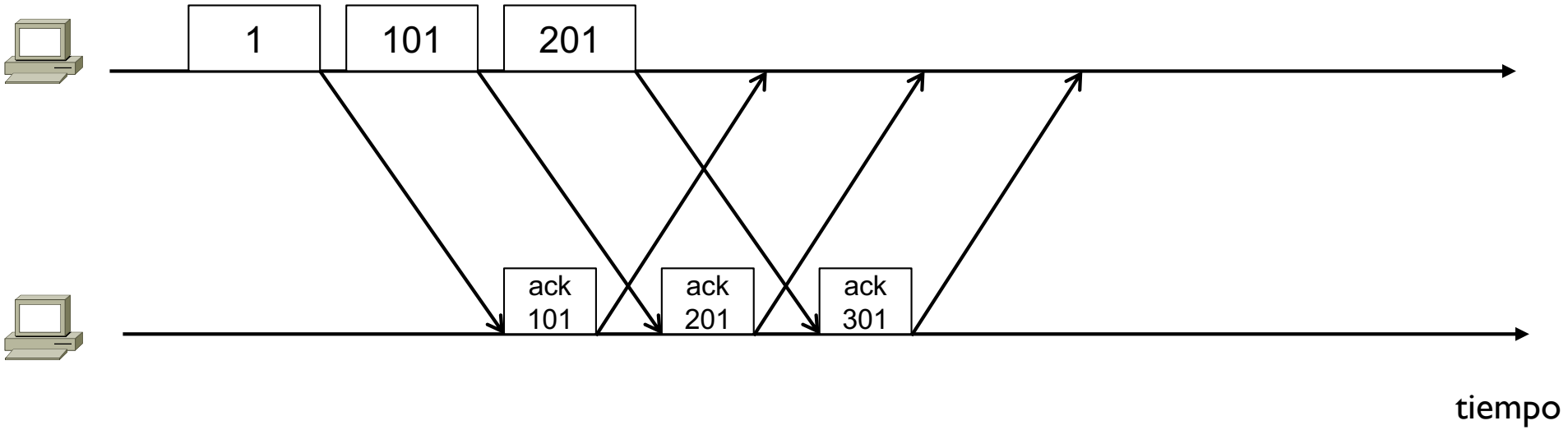
MSS = 100 bytes



Tema 3 – Durante la transmisión

- Se usa una transmisión continua

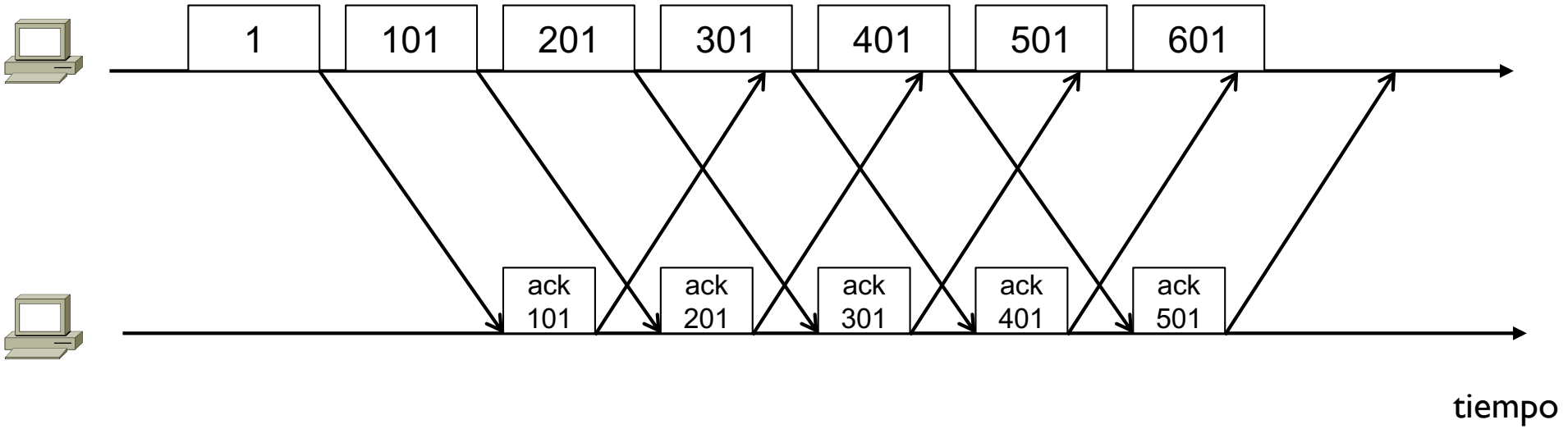
MSS = 100 bytes



Tema 3 – Durante la transmisión

- Se usa una transmisión continua

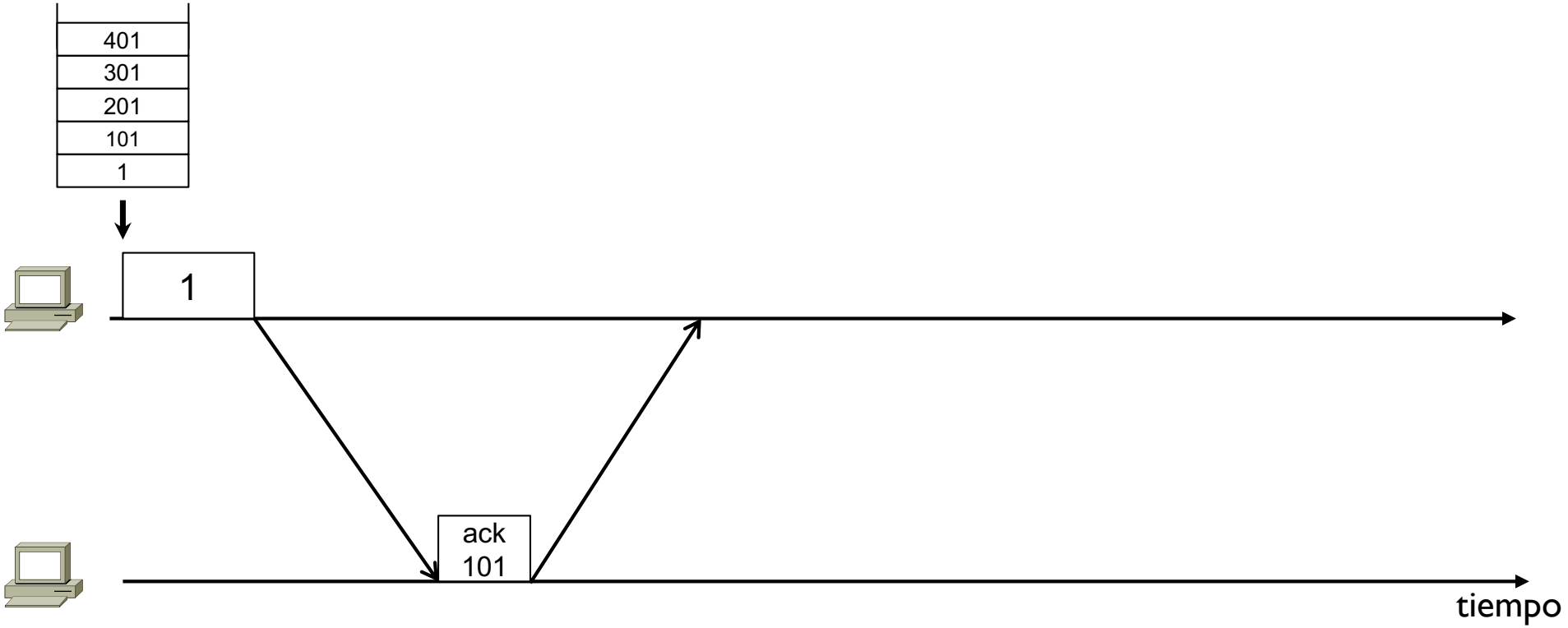
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

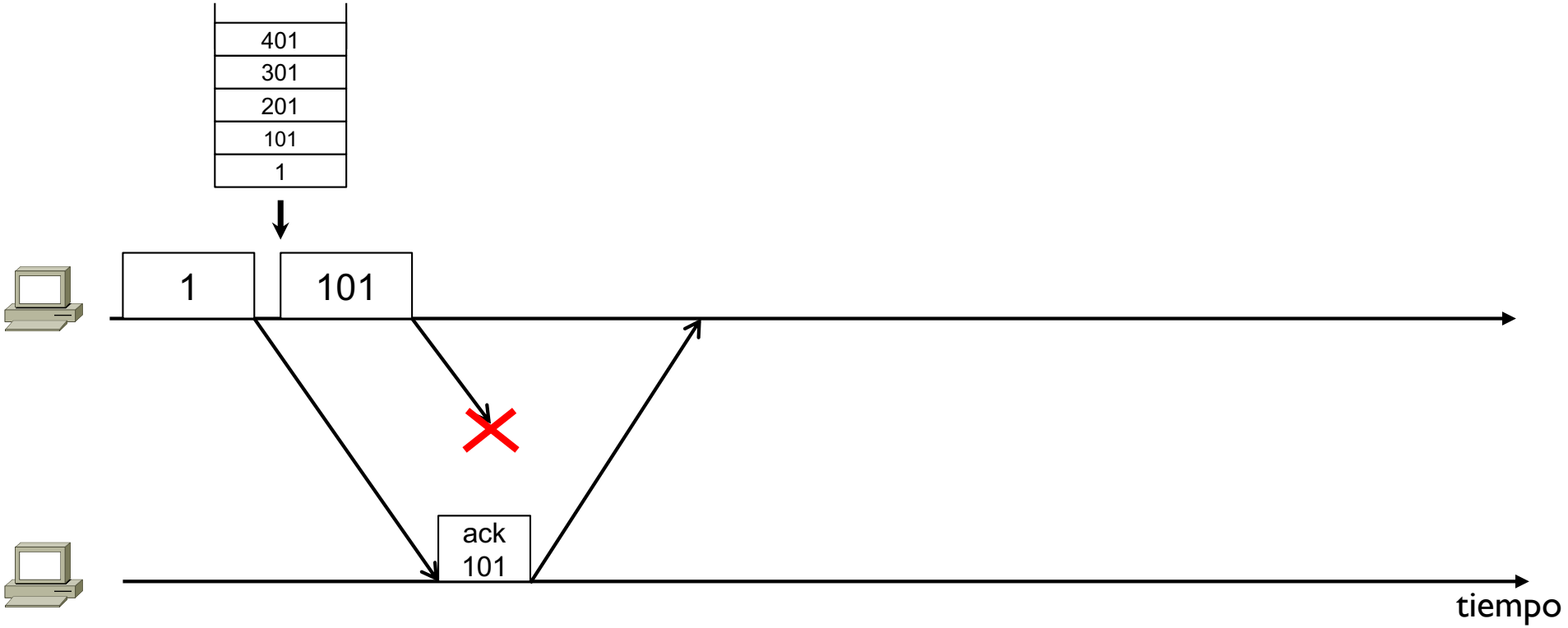
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

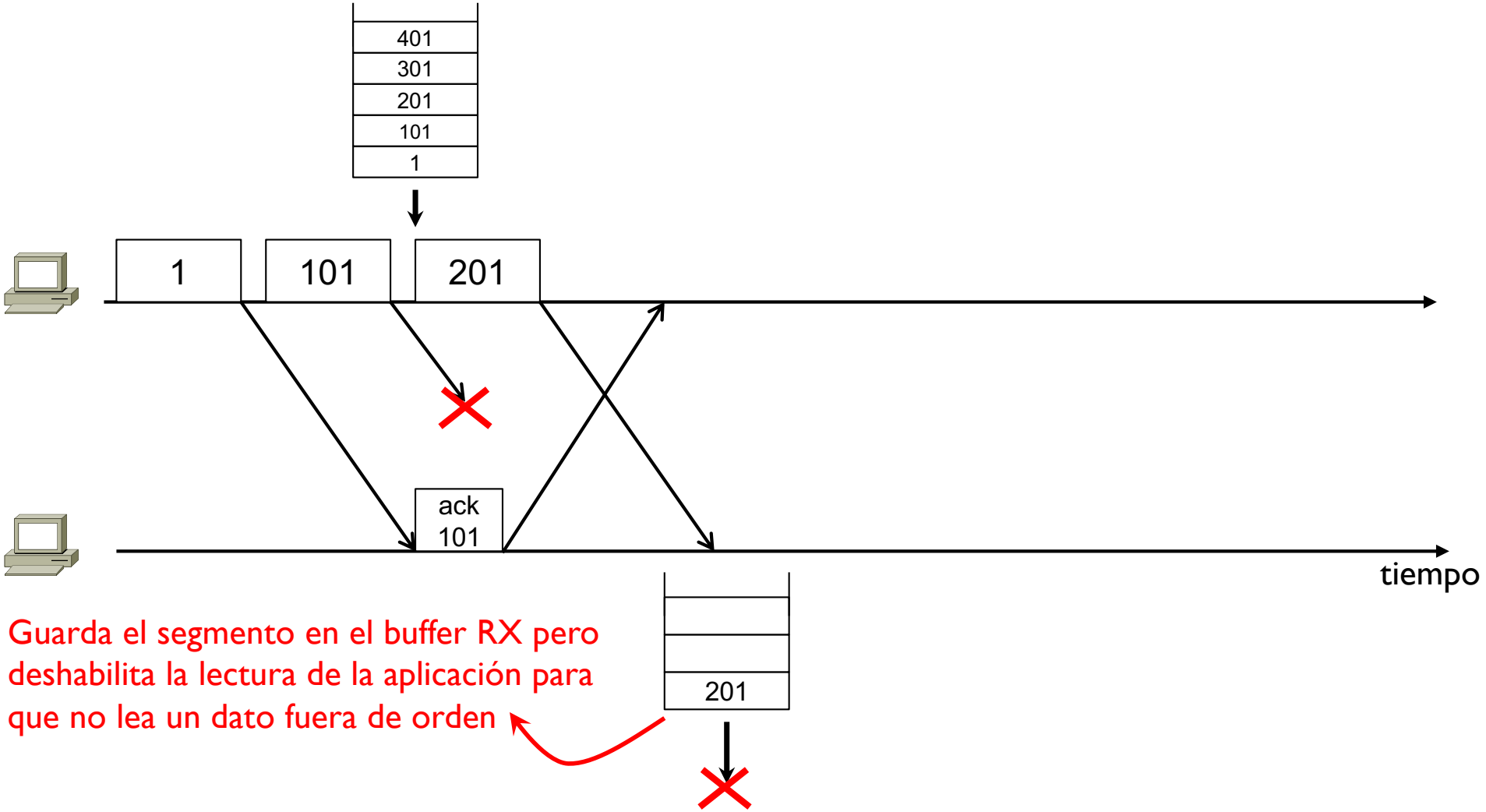
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

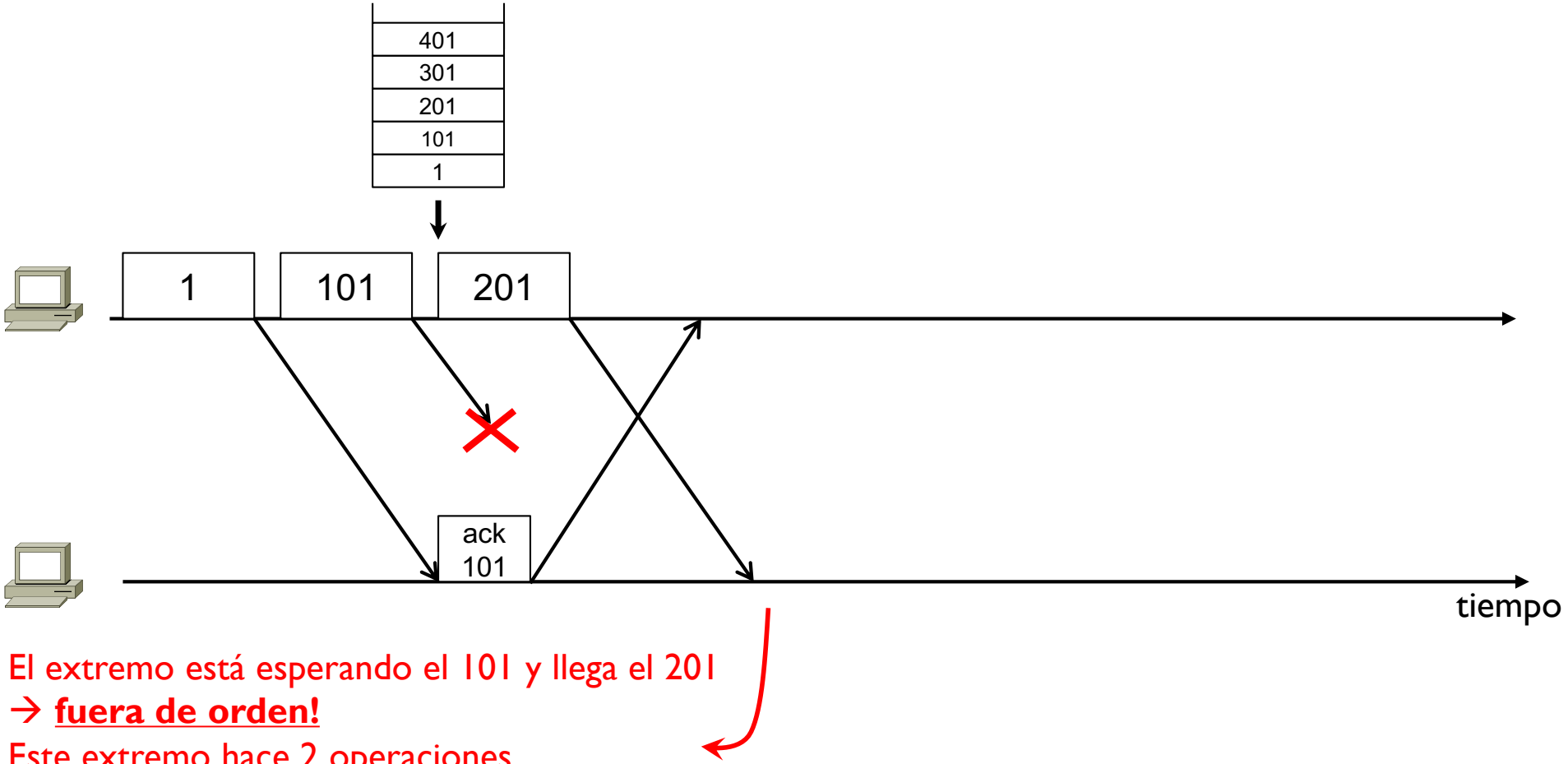
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

MSS = 100 bytes

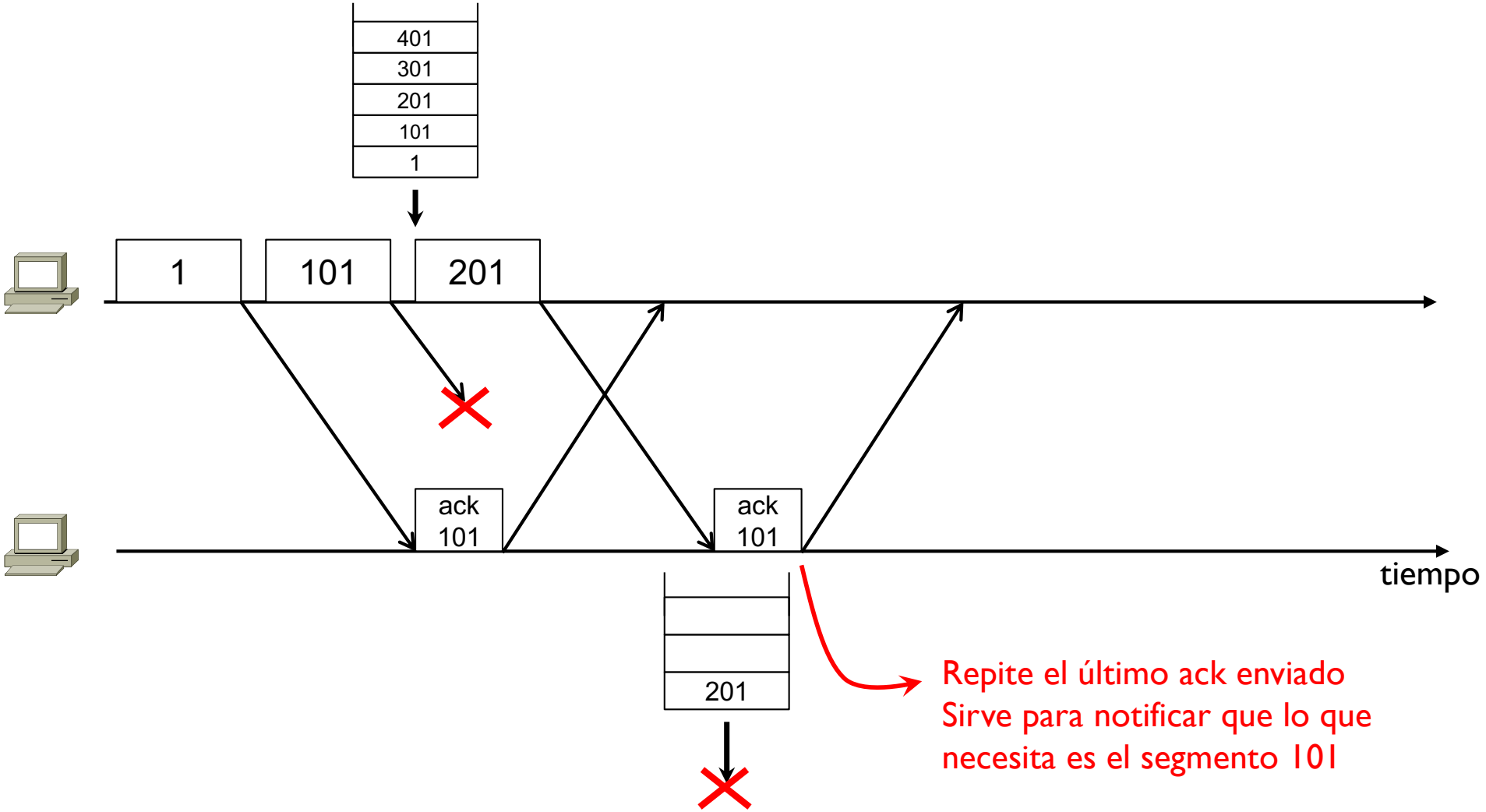


El extremo está esperando el 101 y llega el 201
→ **fuera de orden!**
Este extremo hace 2 operaciones

Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

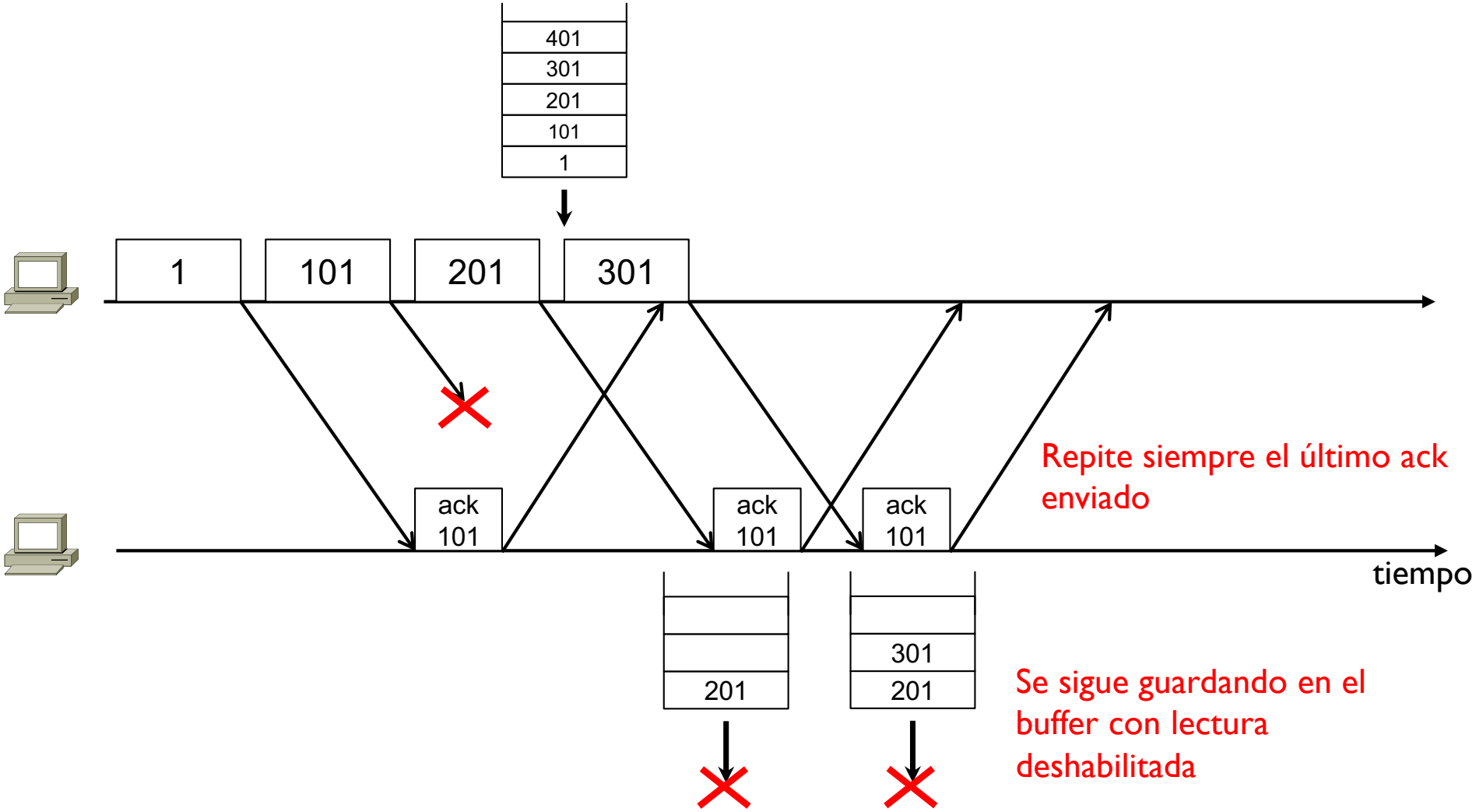
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

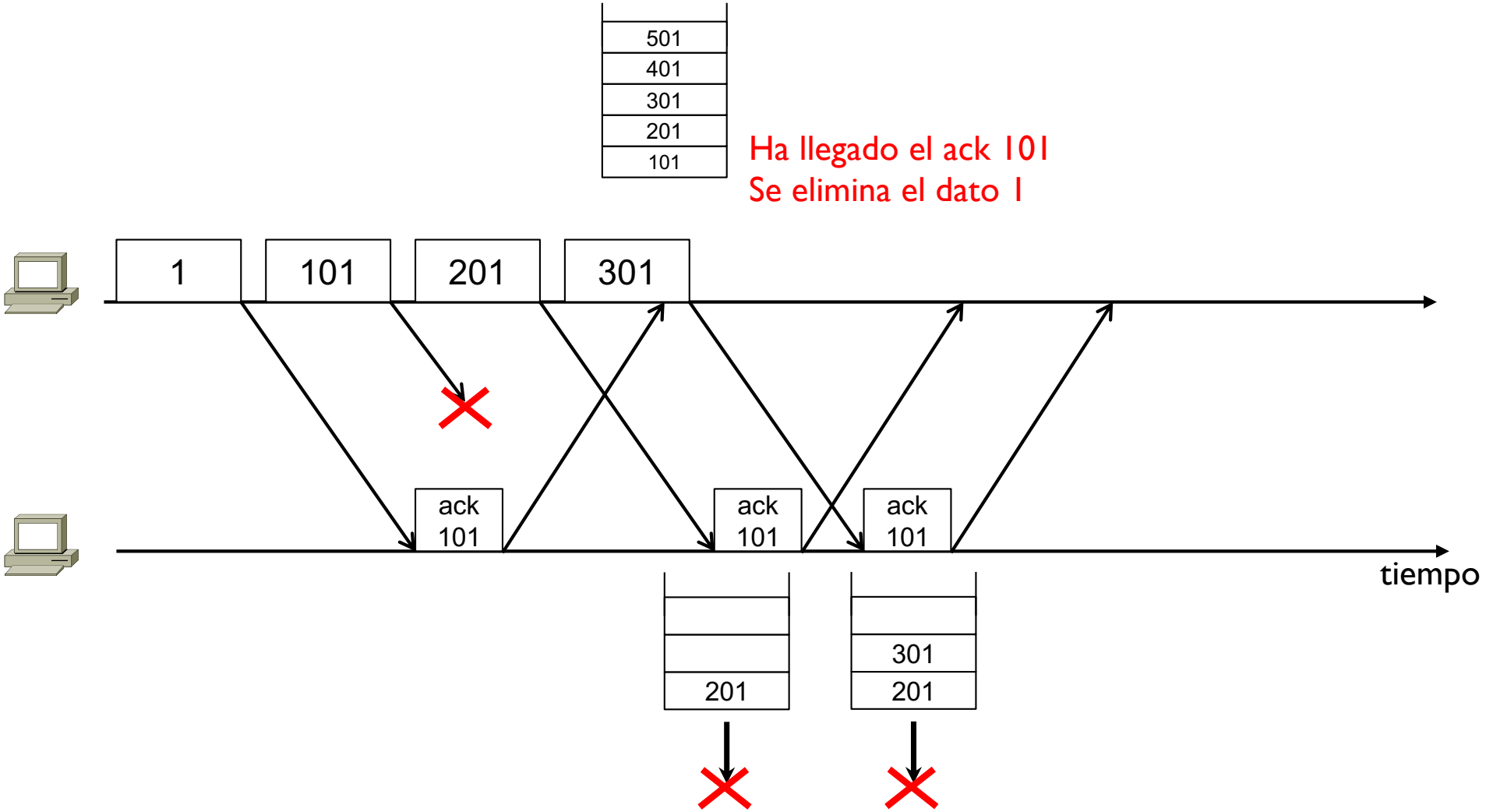
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

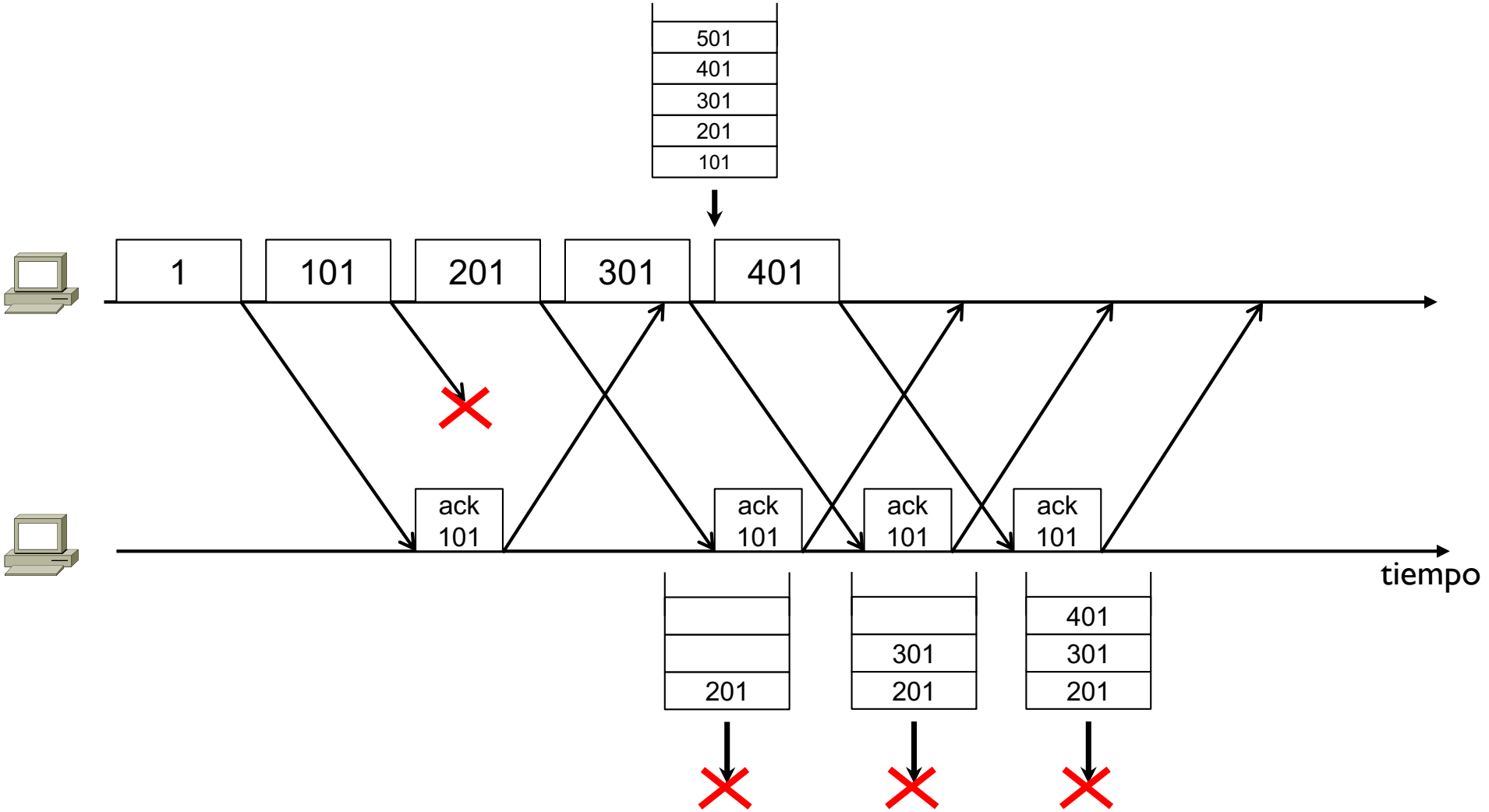
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

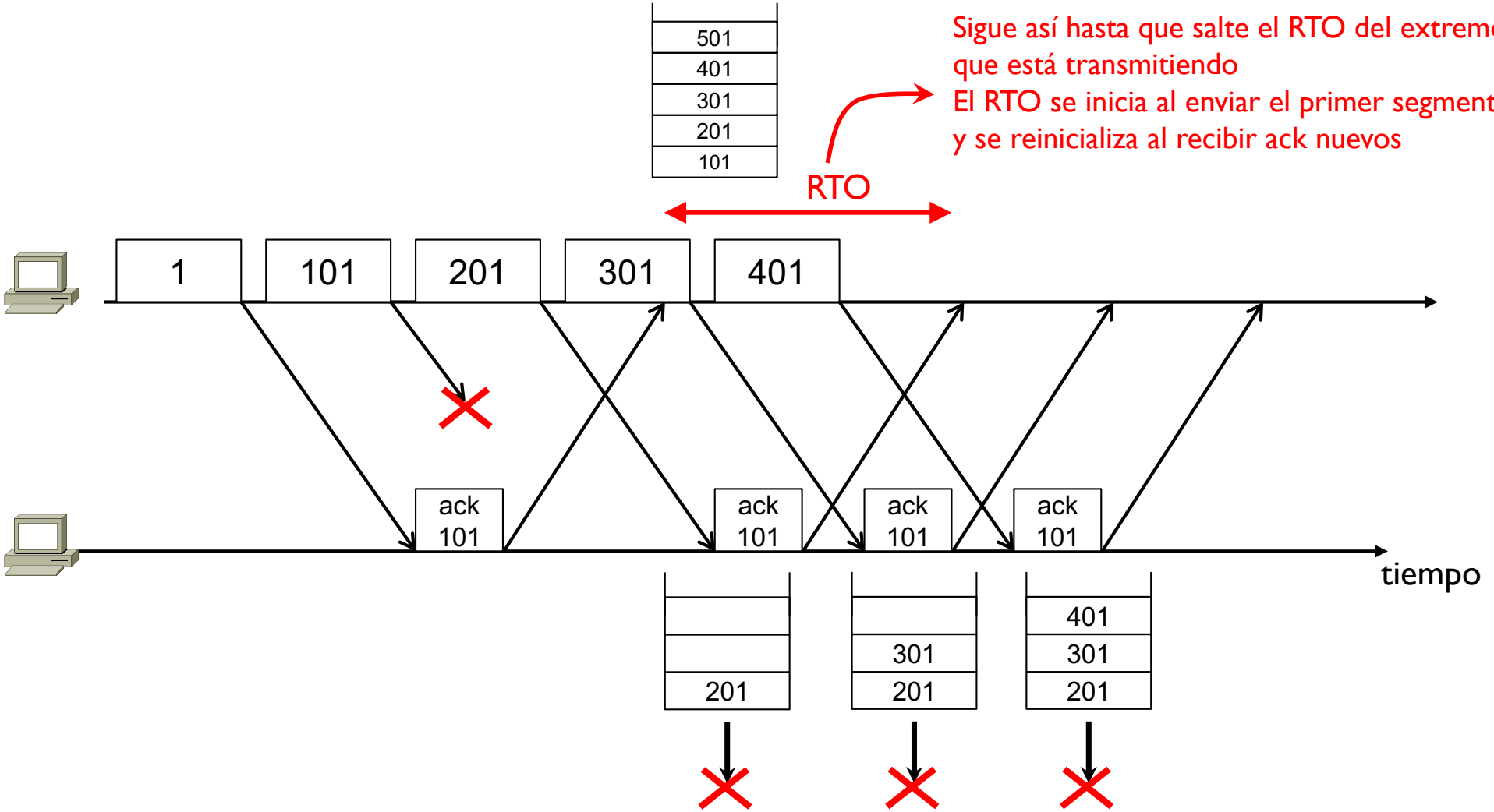
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

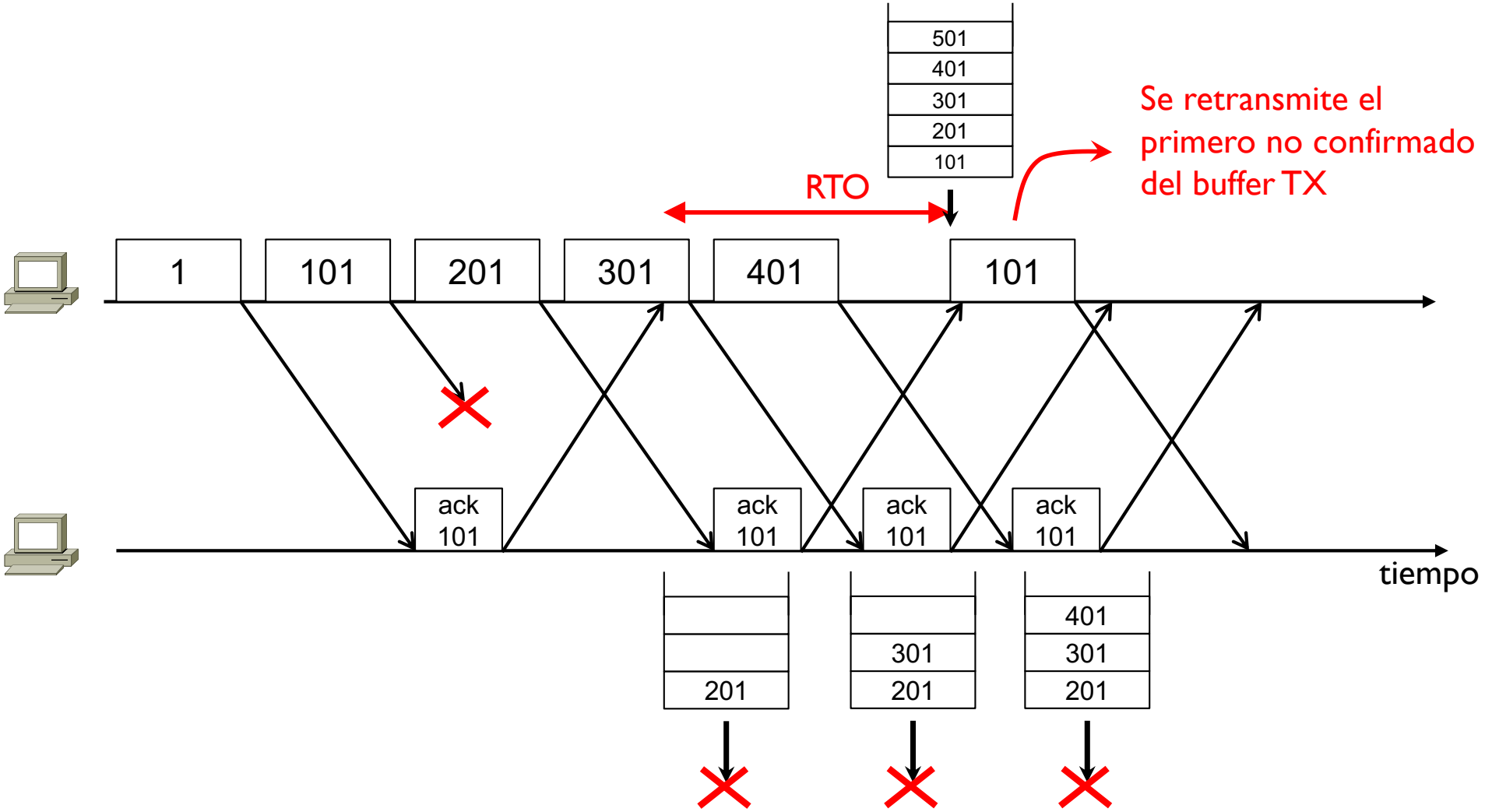
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

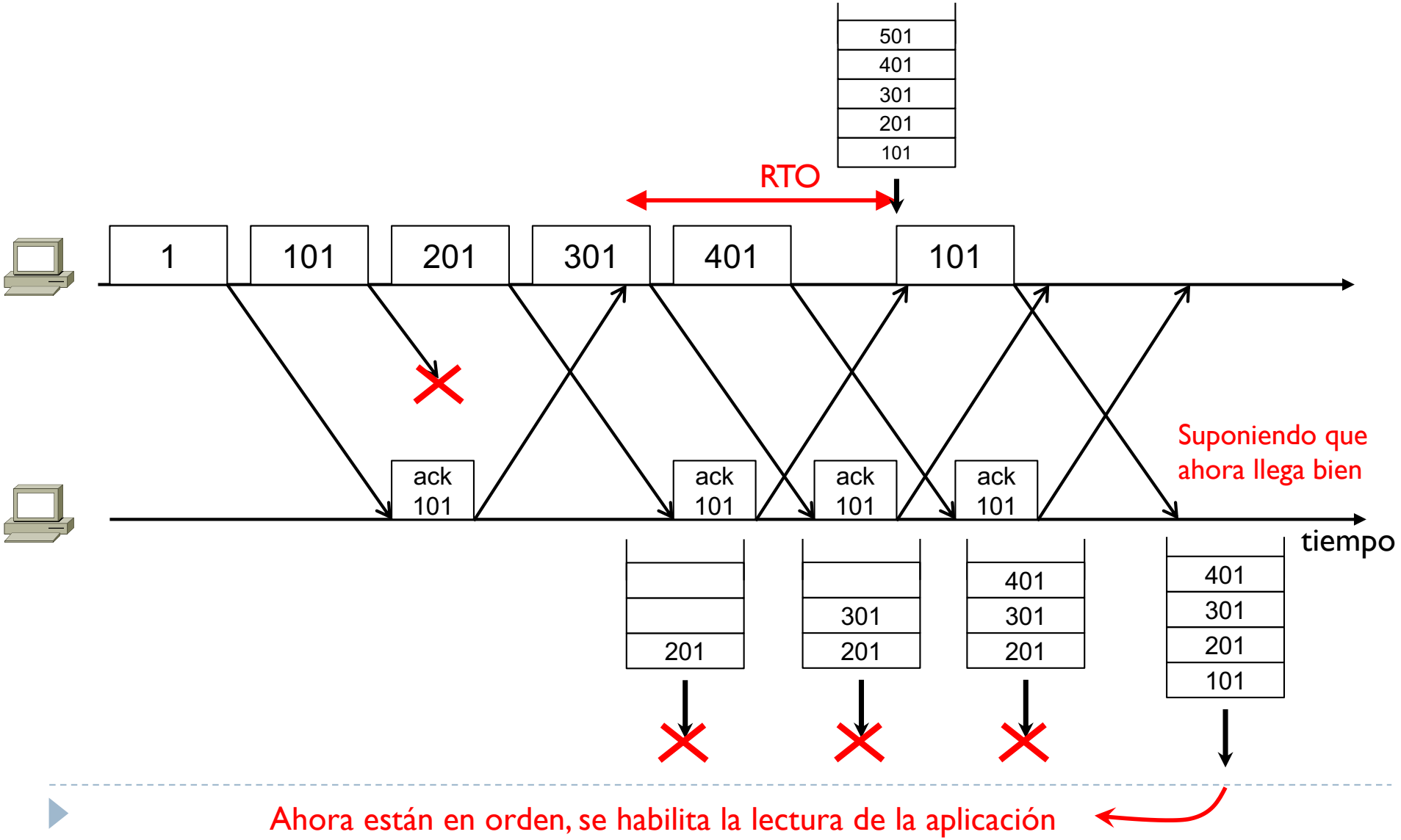
MSS = 100 bytes



Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

MSS = 100 bytes

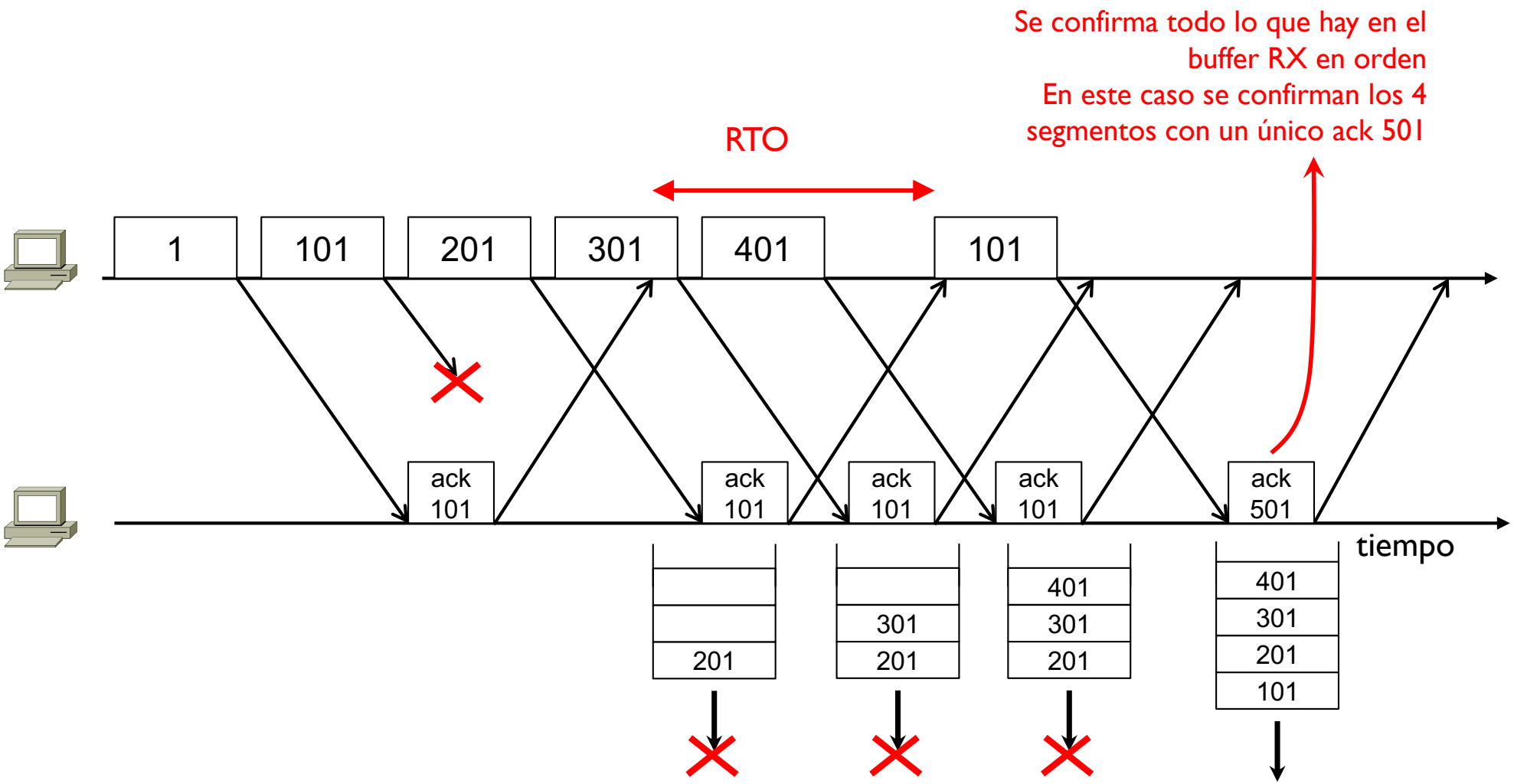


Ahora están en orden, se habilita la lectura de la aplicación

Tema 3 – Durante la transmisión

- ¿Y en caso de perdidas?

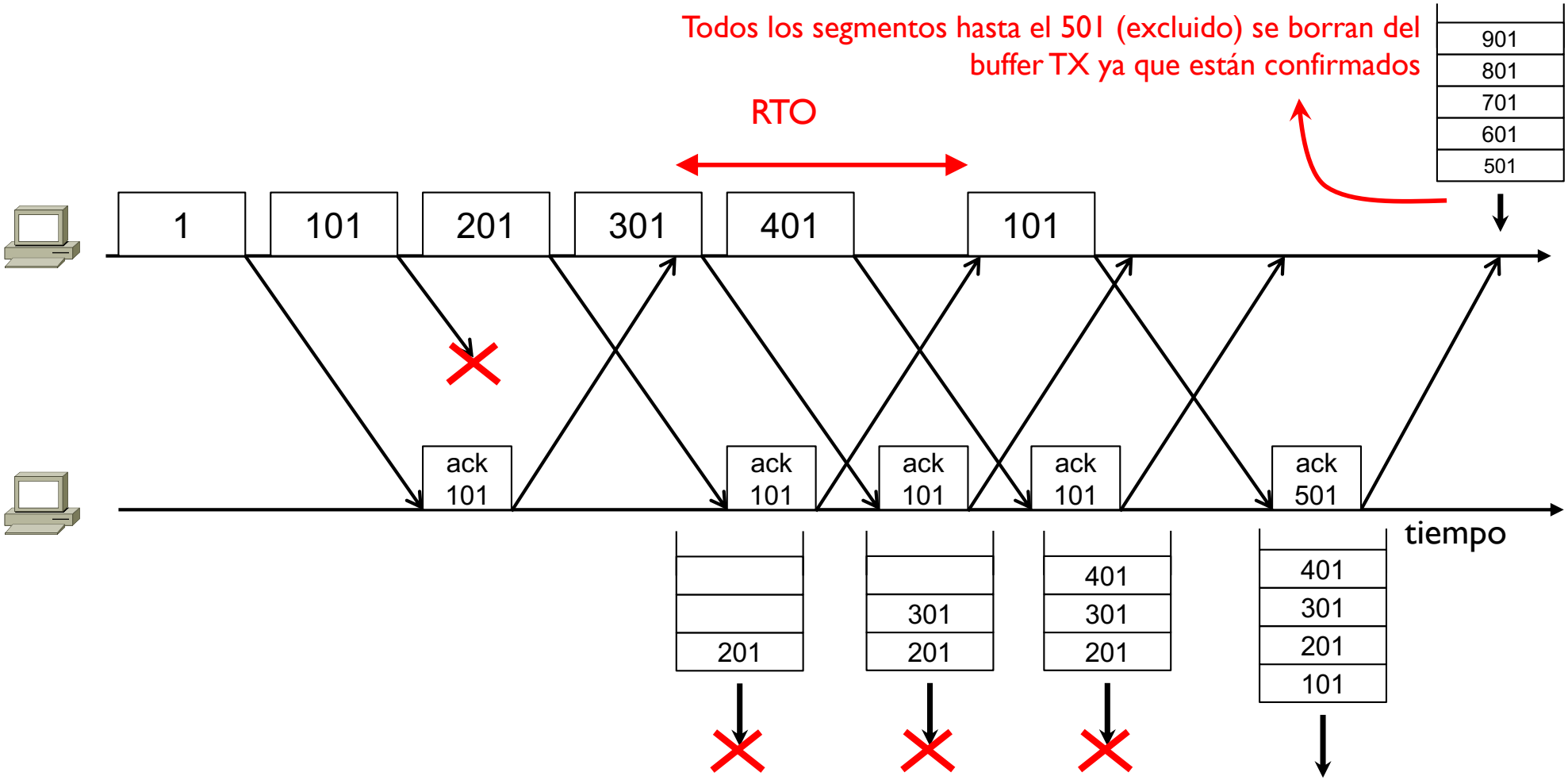
MSS = 100 bytes



Tema 3 – Durante la transmisión

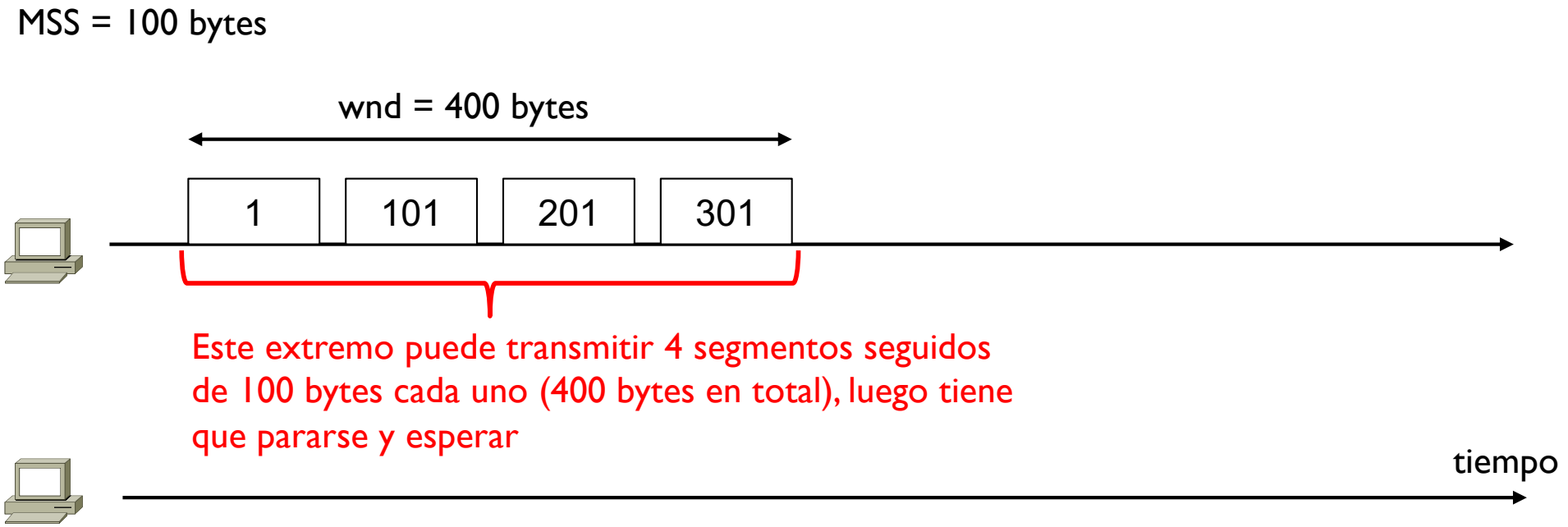
- ¿Y en caso de perdidas?

MSS = 100 bytes



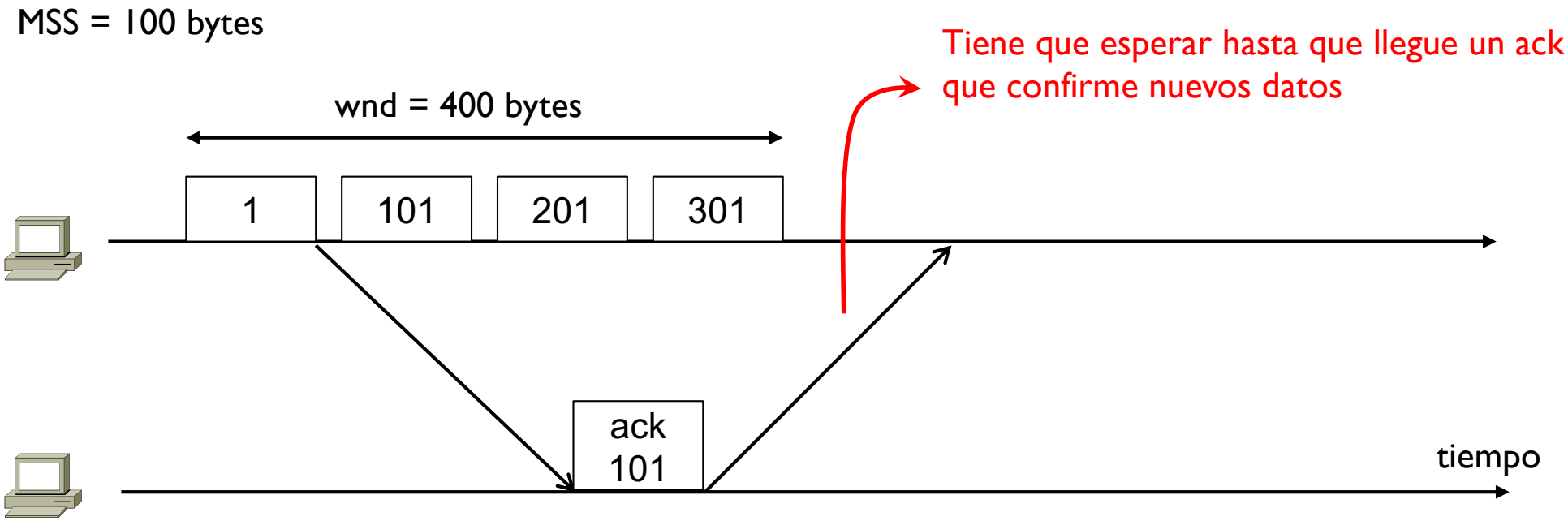
Tema 3 – Durante la transmisión

- La transmisión continua está pero limitada por una ventana deslizante, es decir un extremo transmite tantos segmentos seguidos hasta alcanzar el valor de esta ventana
- Esta ventana se indica con wnd y se define como el máximo número de bytes no confirmados que se pueden transmitir



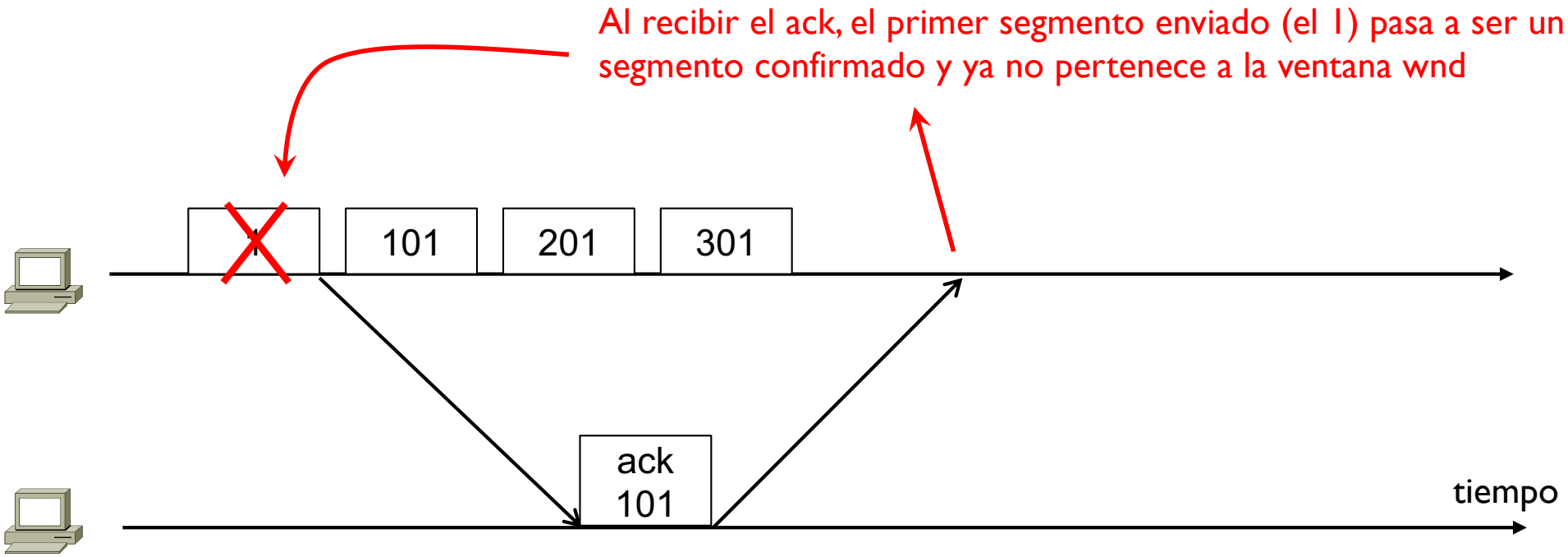
Tema 3 – Durante la transmisión

- La transmisión continua está pero limitada por una ventana deslizante, es decir un extremo transmite tantos segmentos seguidos hasta alcanzar el valor de esta ventana
- Esta ventana se indica con wnd y se define como el máximo número de bytes no confirmados que se pueden transmitir



Tema 3 – Durante la transmisión

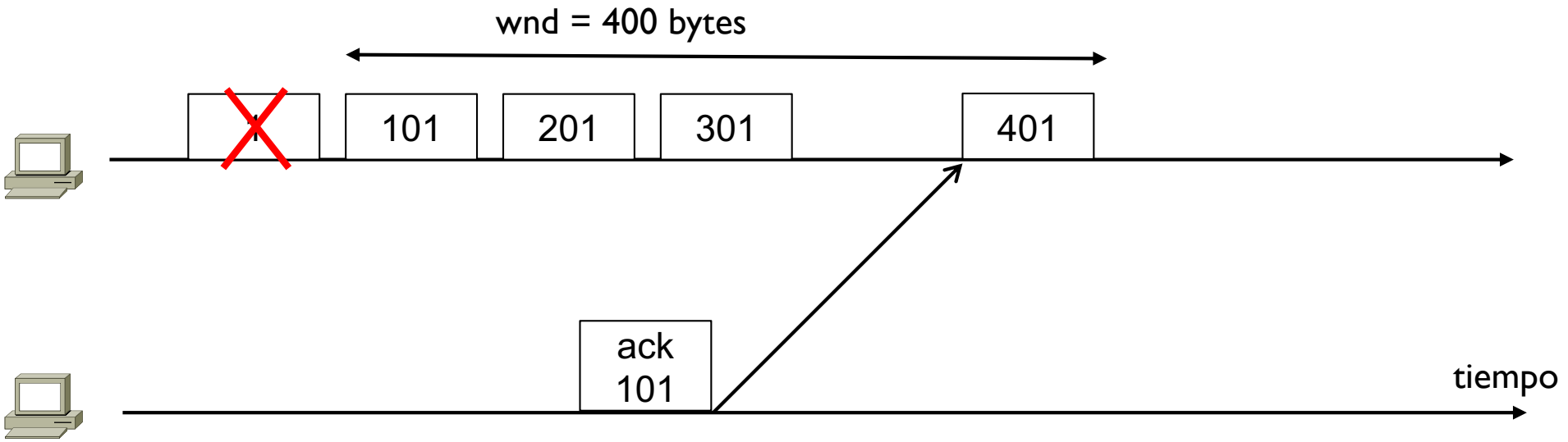
- La transmisión continua está pero limitada por una ventana deslizante, es decir un extremo transmite tantos segmentos seguidos hasta alcanzar el valor de esta ventana
- Esta ventana se indica con wnd y se define como el máximo número de bytes no confirmados que se pueden transmitir



Tema 3 – Durante la transmisión

- La transmisión continua está pero limitada por una ventana deslizante, es decir un extremo transmite tantos segmentos seguidos hasta alcanzar el valor de esta ventana
- Esta ventana se indica con wnd y se define como el máximo número de bytes no confirmados que se pueden transmitir

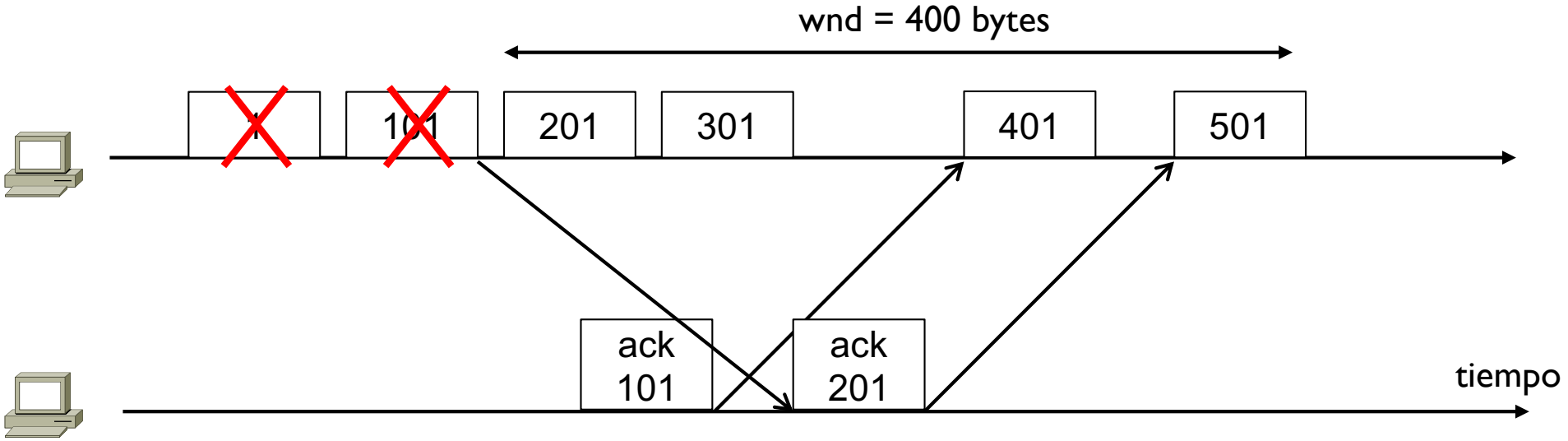
La ventana wnd se dice que se desliza hacía adelante
Es decir permite que se transmitan otros 100 bytes de datos



Tema 3 – Durante la transmisión

- La transmisión continua está pero limitada por una ventana deslizante, es decir un extremo transmite tantos segmentos seguidos hasta alcanzar el valor de esta ventana
- Esta ventana se indica con wnd y se define como el máximo número de bytes no confirmados que se pueden transmitir

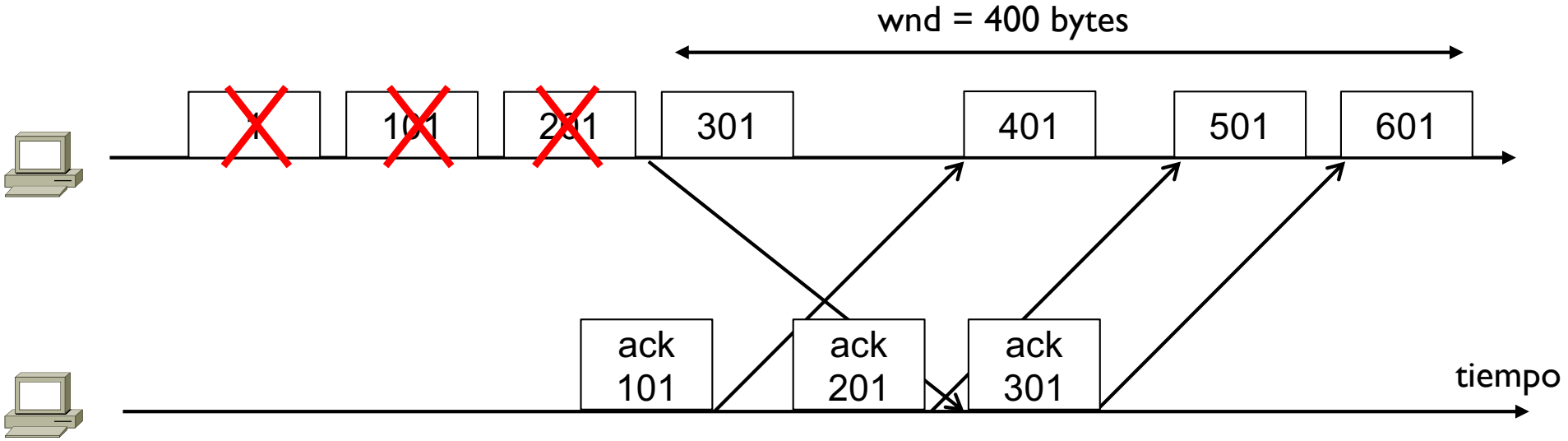
Y sigue así con todos los segmentos enviados
Cada vez que llega un ack nuevo, se puede transmitir un nuevo segmento de datos



Tema 3 – Durante la transmisión

- La transmisión continua está pero limitada por una ventana deslizante, es decir un extremo transmite tantos segmentos seguidos hasta alcanzar el valor de esta ventana
- Esta ventana se indica con wnd y se define como el máximo número de bytes no confirmados que se pueden transmitir

Y sigue así con todos los segmentos enviados
Cada vez que llega un ack nuevo, se puede transmitir un nuevo segmento de datos



Tema 3 – Durante la transmisión

- El valor de la ventana deslizante wnd es dinámico y depende de la aplicación de dos mecanismos

- Control de flujo
  - Adapta la tasa de envío* de datos a la capacidad del extremo receptor de guardar estos datos
  - Este mecanismo proporciona la ventana anunciada awnd
- Control de congestión
  - Adapta la tasa de envío* de datos al nivel de congestión de la red entre origen (transmisor) y destino (receptor)
  - Este mecanismo proporciona la ventana de congestión cwnd

- La ventana wnd es el mínimo entre las dos ventanas

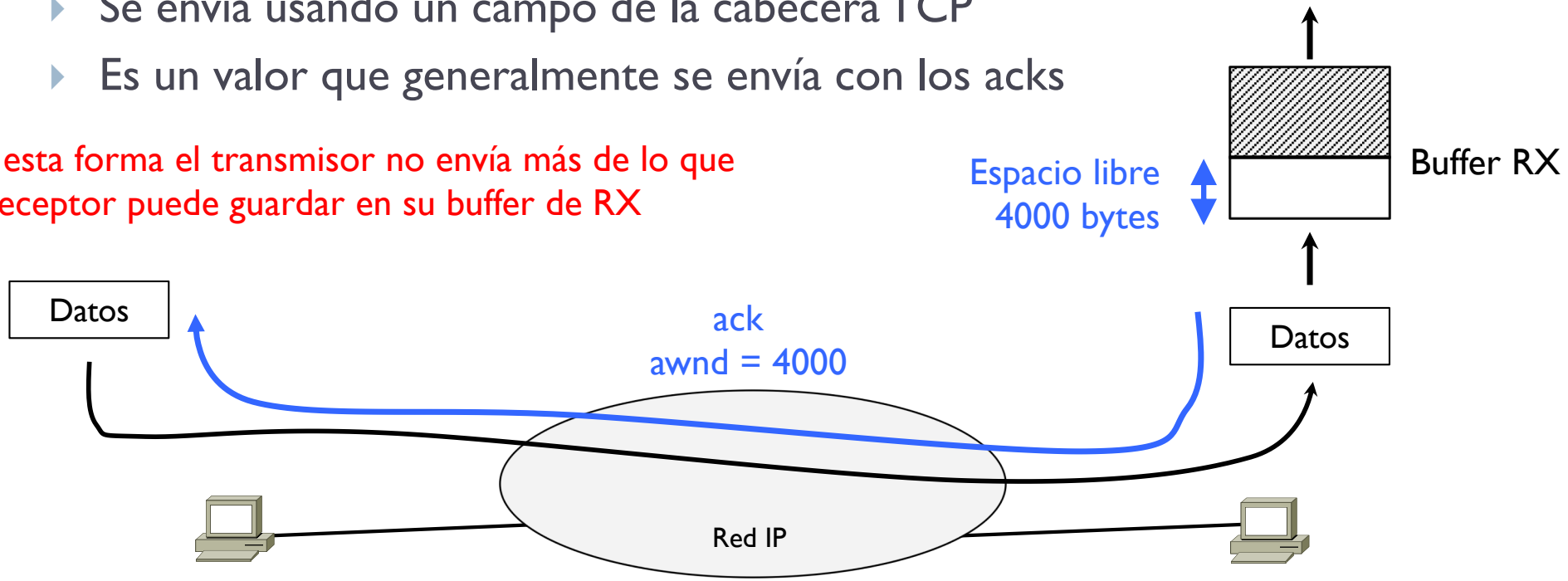
$$wnd = \min (cwnd, awnd)$$

* Una tasa de envío indica la cantidad de datos enviados por segundo

Tema 3 – Control de flujo

- Objetivo: adaptar la tasa de envío del transmisor a la capacidad de recepción del receptor
- awnd
  - Ventana anunciada por el extremo receptor al transmisor
  - Su valor corresponde al espacio libre en el buffer de RX
  - Se envía usando un campo de la cabecera TCP
  - Es un valor que generalmente se envía con los acks

De esta forma el transmisor no envía más de lo que el receptor puede guardar en su buffer de RX



Tema 3 – Control de congestión

- Objetivo: adaptar la tasa de envío del transmisor a la capacidad de la red
- cwnd
  - Es un valor que no se puede determinar con simplicidad como awnd
  - La red va cambiando su estado y su ocupación constantemente
  - Es un valor que el transmisor solo puede **estimar** usando un algoritmo
- Funcionamiento básico de estos algoritmos
  - La idea base de este algoritmo es empezar con una tasa muy baja de envíos e ir aumentando progresivamente
  - Se aumenta la tasa de envío (más datos por segundo) para ver hasta que punto la red aguanta (es decir no entra en congestión)
  - Se descubre que la red está en congestión cuando se pierde un dato (es decir la tasa de envío ahora es demasiado alta), en este caso el algoritmo vuelve al punto de partido y vuelve a comenzar

Tema 3 – Control de congestión

- Algoritmos más conocidos
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmission
  - Fast Recovery
  - Selective Ack

TCP 1983 → TCP Tahoe 1988 → TCP Reno 1990 → TCP Sack 1996

TCP Hybla 2004

TCP Westwood+ 2004

TCP CUBIC 2007

Compound TCP 2003

...

TCP New Vegas 2007

TCP BIC 2005

TCP New Reno 2012

TCP PRR 2014

Tema 3 – Control de congestión

- Algoritmos más conocidos
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmission
  - Fast Recovery
  - Selective Ack

TCP 1983 → TCP Tahoe 1988 → TCP Reno 1990 → TCP Sack 1996

TCP Hybla 2004

TCP Westwood+ 2004

TCP CUBIC 2007

Compound TCP 2003

...

TCP New Vegas 2007

TCP BIC 2005

TCP New Reno 2012

TCP PRR 2014

Tema 3 – Slow Start + Congestion Avoidance

Inicio

cwnd = 1 MSS

ssthresh = infinito

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)

Se reinicia el RTO

si cwnd < ssthresh → se aplica Slow Start (SS)

cwnd = cwnd + 1 MSS

si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)

cwnd = cwnd + MSS * (MSS / cwnd)

Si hay una perdida y salta el RTO

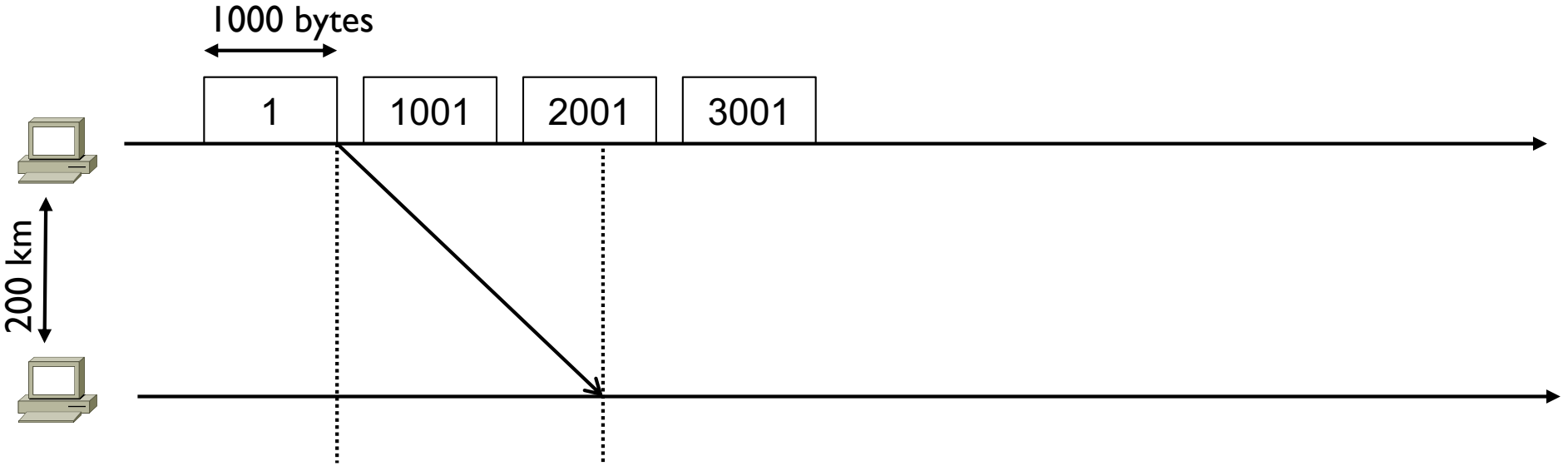
ssthresh = max(2 MSS, wnd / 2)

cwnd = 1 MSS

Se retransmite el dato perdido

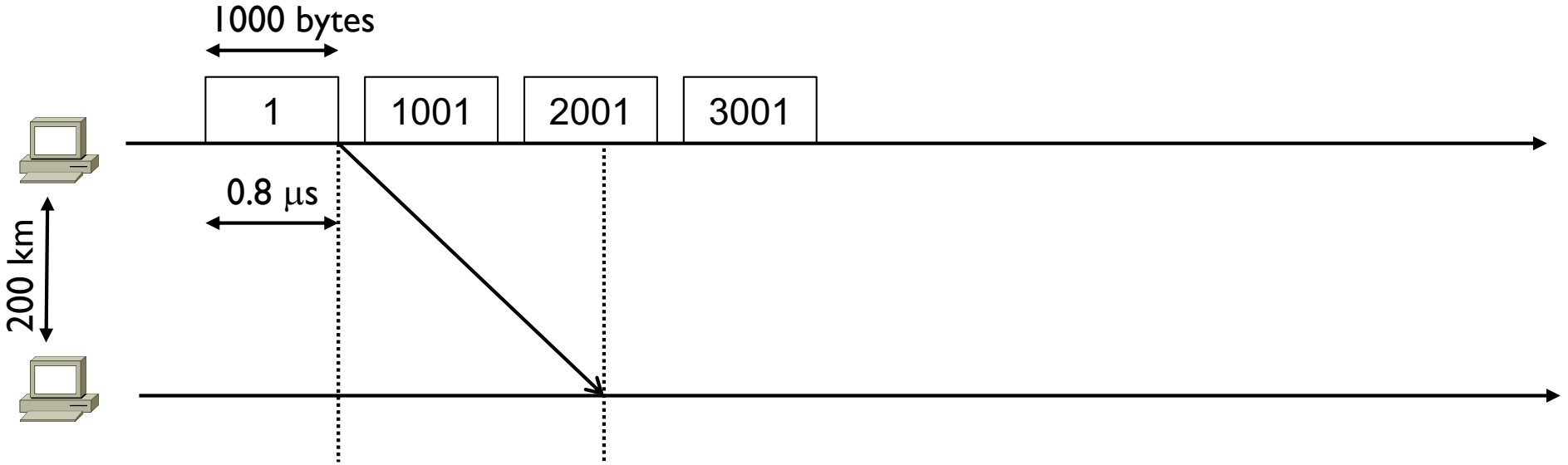
Tema 3 – Simplificación

- Hasta ahora se han representado los datos enviados de esta forma



Tema 3 – Simplificación

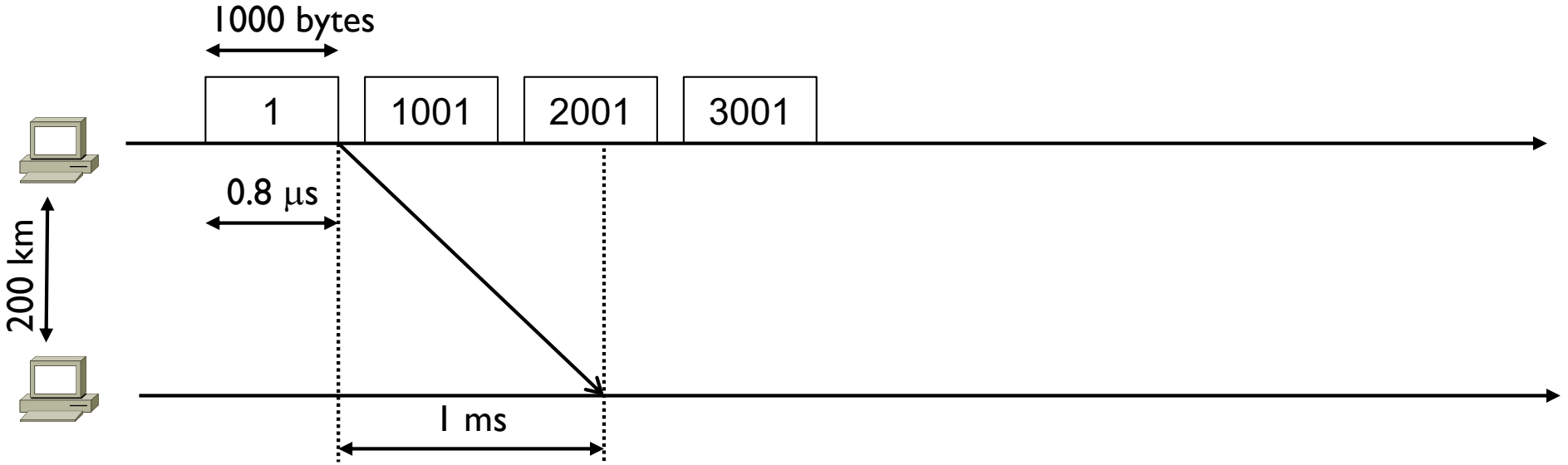
- Hasta ahora se han representado los datos enviados de esta forma



- Pero vamos a poner todo en su escala en un ejemplo
  - Se transmite a 10 Mbit/s
  - 1000 bytes entonces se transmiten en 1000 * 8 bits / 10 Mbit/s = 0.8 10^{-6}s = 0.8 μs

Tema 3 – Simplificación

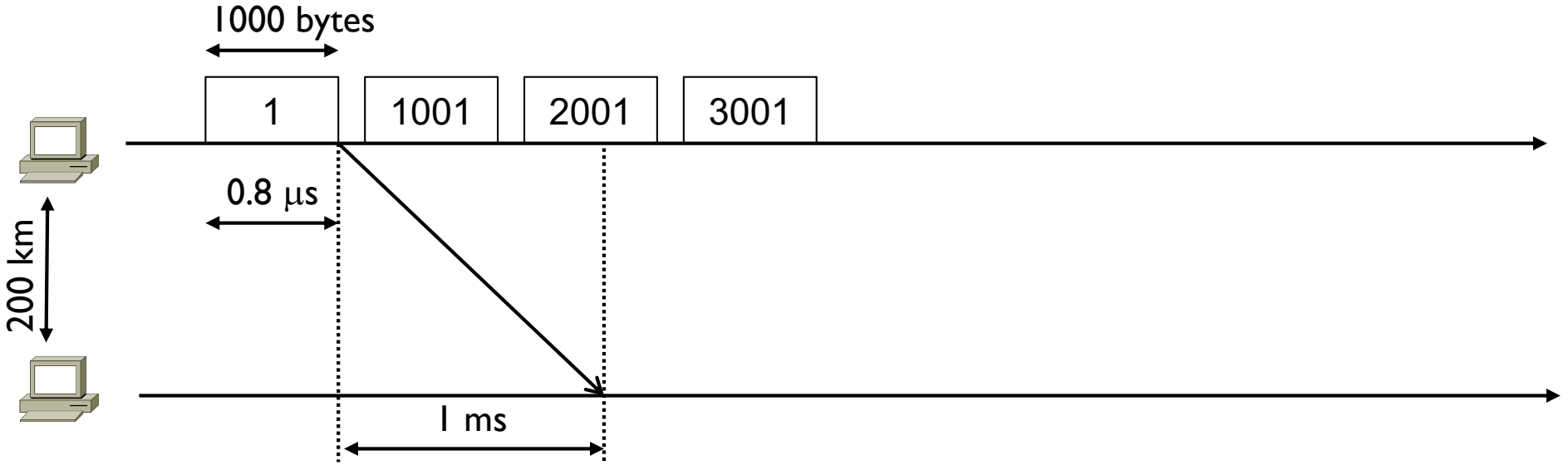
- Hasta ahora se han representado los datos enviados de esta forma



- Pero vamos a poner todo en su escala en un ejemplo
  - Se transmite a 10 Mbit/s
  - 1000 bytes entonces se transmiten en 1000 * 8 bits / 10 Mbit/s = 0.8 10^{-6}s = 0.8 μs
  - Suponiendo una distancia de 200 km entre los extremos, en el mejor de los casos la velocidad de la información viaja a 2/3 de la velocidad de la luz, es decir 2 10^{8} m/s
  - 200 km / 2 10^{8} m/s = 1 ms

Tema 3 – Simplificación

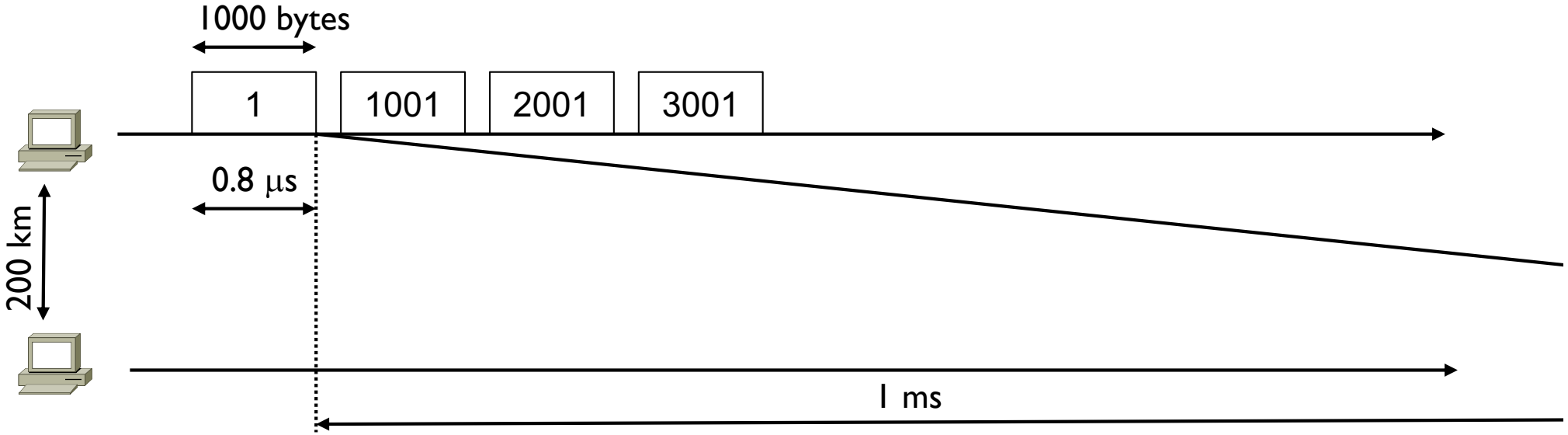
- Hasta ahora se han representado los datos enviados de esta forma



- Pero vamos a poner todo en su escala en un ejemplo
  - Se transmite a 10 Mbit/s
  - 1000 bytes entonces se transmiten en 1000 * 8 bits / 10 Mbit/s = 0.8 10^{-6}s = 0.8 μs
  - Suponiendo una distancia de 200 km entre los extremos, en el mejor de los casos la velocidad de la información viaja a 2/3 de la velocidad de la luz, es decir 2 10^8 m/s
  - 200 km / 2 10^8 m/s = 1 ms
  - 1 ms es más de 1000 veces más grande de 0.8 μs

Tema 3 – Simplificación

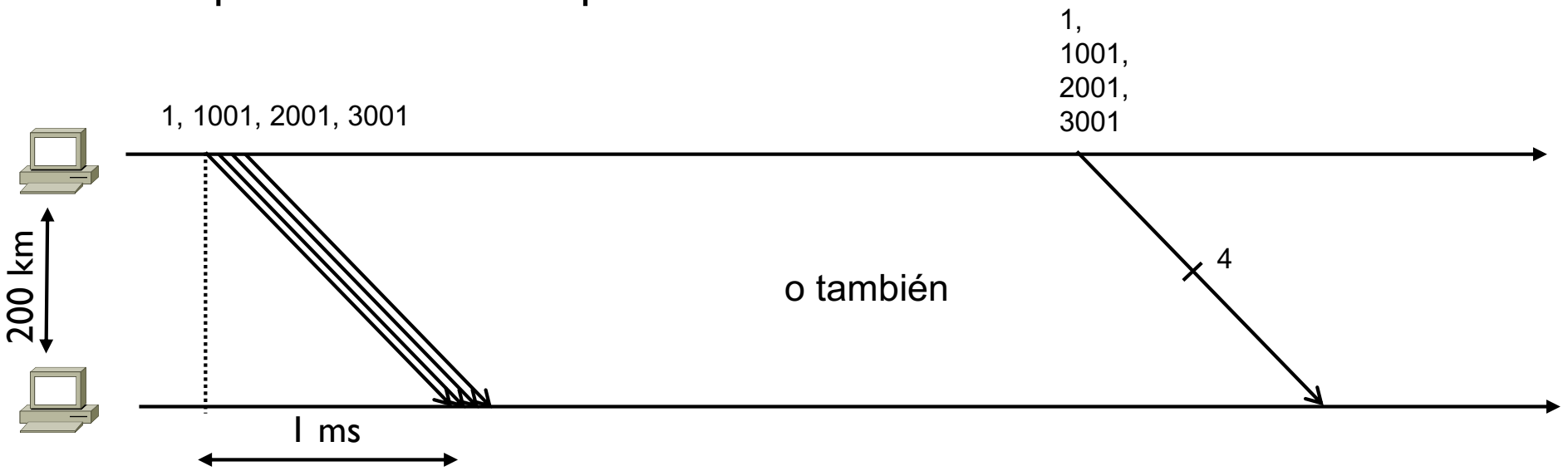
- Hasta ahora se han representado los datos enviados de esta forma



- Pero vamos a poner todo en su escala en un ejemplo
  - Se transmite a 10 Mbit/s
  - 1000 bytes entonces se transmiten en 1000 * 8 bits / 10 Mbit/s = 0.8 10^{-6}s = 0.8 μs
  - Suponiendo una distancia de 200 km entre los extremos, en el mejor de los casos la velocidad de la información viaja a 2/3 de la velocidad de la luz, es decir 2 10^{8} m/s
  - 200 km / 2 10^{8} m/s = 1 ms
  - 1 ms es más de 1000 veces más grande de 0.8 μs

Tema 3 – Simplificación

- Lo que realmente se supone es eso



- Como en el caso anterior se transmiten 4 segmentos de 1000 bytes cada uno
- Para ponerlo en escala real, la duración de los 4 es despreciable respecto a lo que se tarda para llegar al otro extremo
- Por eso se pueden simplemente representar como de duración (casi) nula

Tema 3 – Slow Start + Congestion Avoidance

Inicio

cwnd = 1 MSS

ssthresh = infinito

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)

Se reinicia el RTO

si cwnd < ssthresh → se aplica Slow Start (SS)

cwnd = cwnd + 1 MSS

si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)

cwnd = cwnd + MSS * (MSS / cwnd)

Si hay una perdida y salta el RTO

ssthresh = max(2 MSS, wnd / 2)

cwnd = 1 MSS

Se retransmite el dato perdido

Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

awnd > cwnd → por simplicidad
ssthresh = infinito

cwnd = 1 MSS

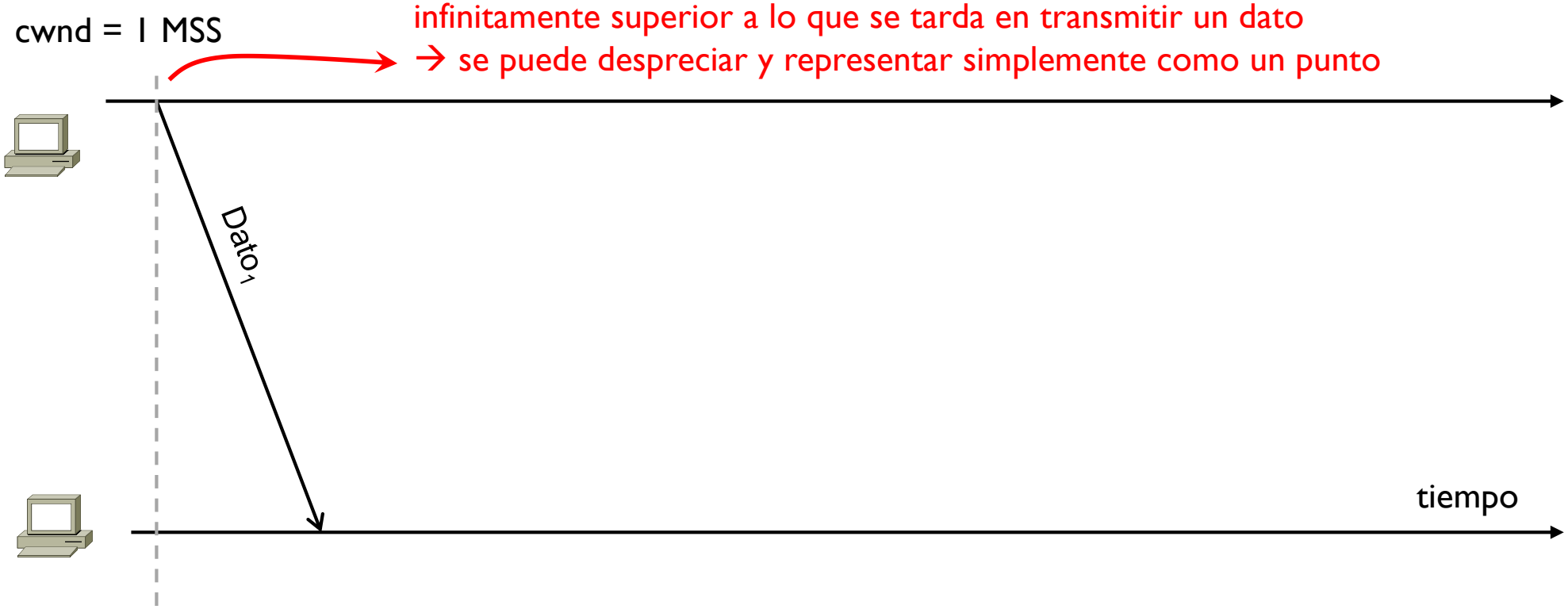
Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

awnd > cwnd → por simplicidad
ssthresh = infinito

El tiempo que se tarda para llegar al otro extremo es generalmente infinitamente superior a lo que se tarda en transmitir un dato
→ se puede despreciar y representar simplemente como un punto



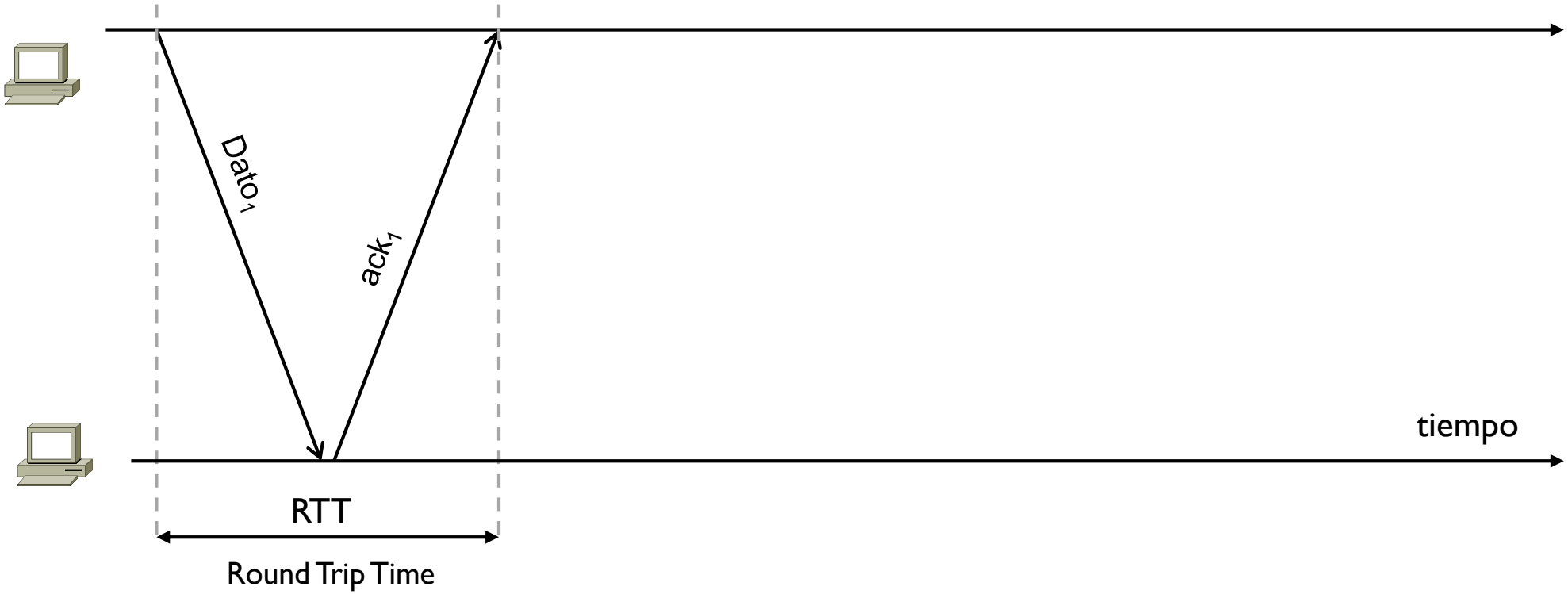
Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

awnd > cwnd → por simplicidad
ssthresh = infinito

cwnd = 1 MSS



Tema 3 – Slow Start + Congestion Avoidance

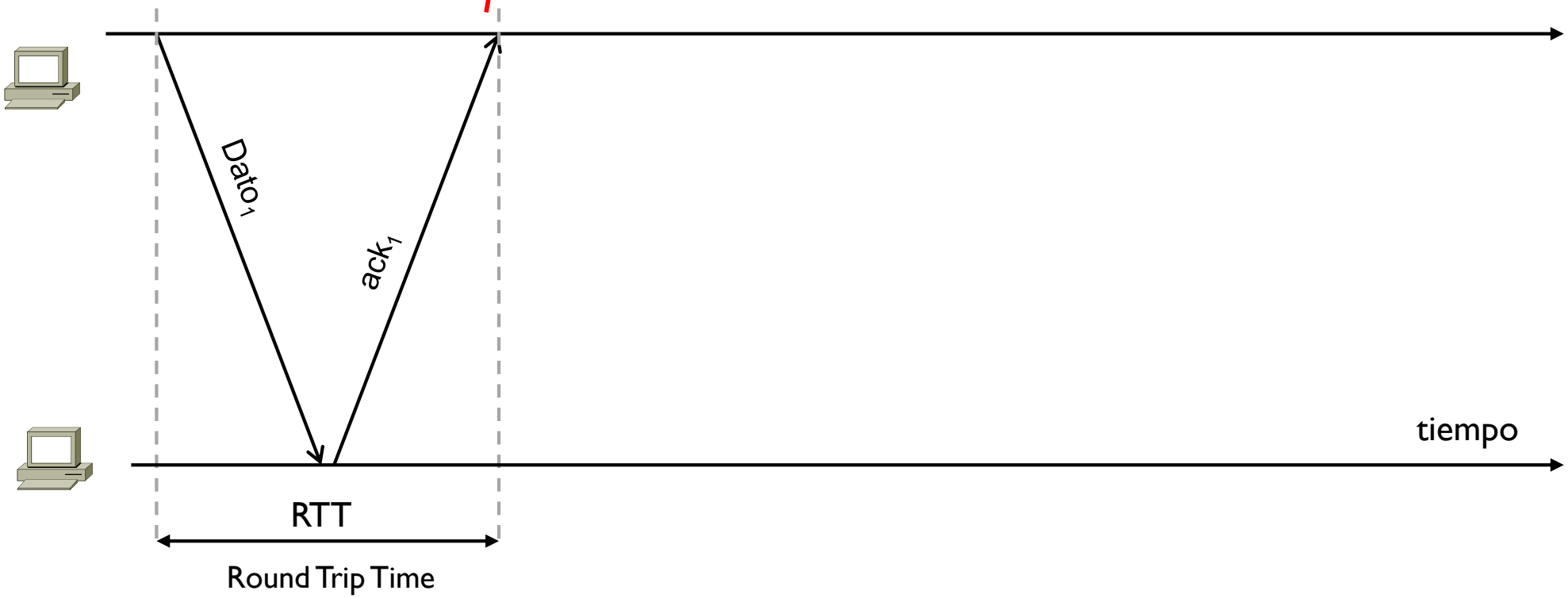
Ejemplo

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

awnd > cwnd → por simplicidad
ssthresh = infinito

Es un ack nuevo
cwnd ≤ ssthresh → se aplica Slow Start (SS)
cwnd ahora vale 1 MSS + 1 MSS = 2 MSS

cwnd = 1 MSS



Tema 3 – Slow Start + Congestion Avoidance

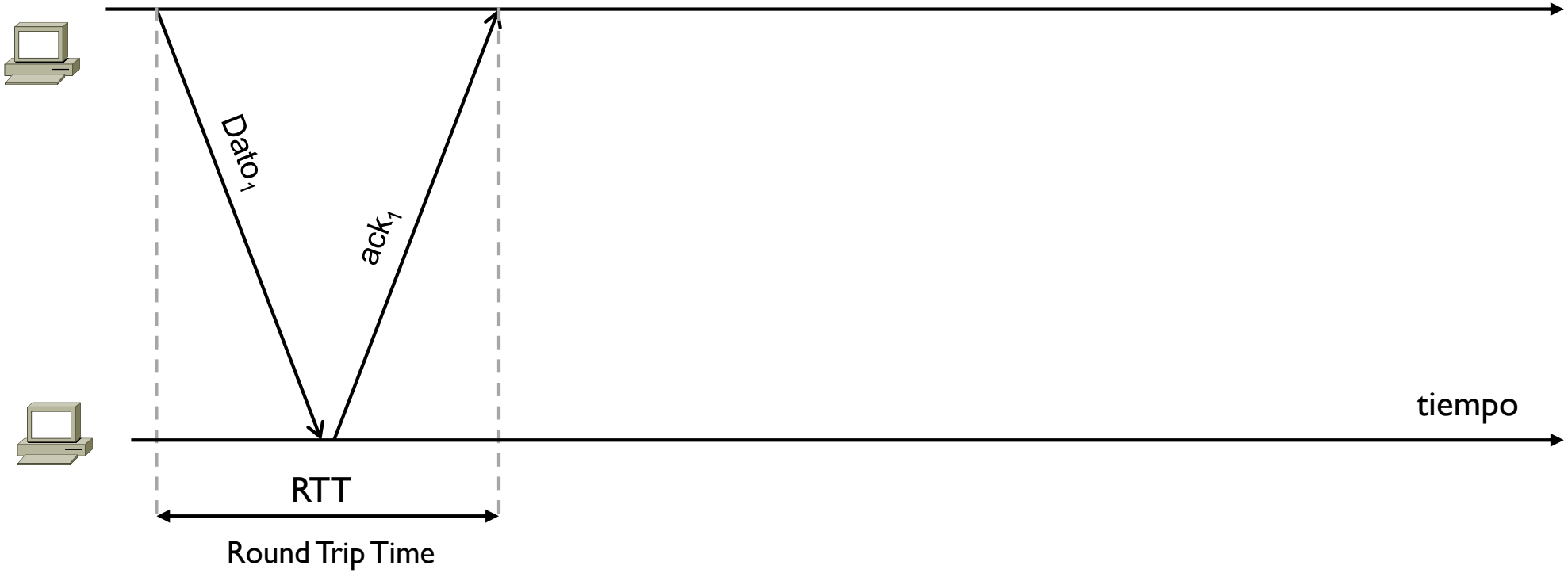
Ejemplo

awnd > cwnd → por simplicidad
ssthresh = infinito

cwnd = 1 MSS

cwnd = 1 + 1 = 2 MSS

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)



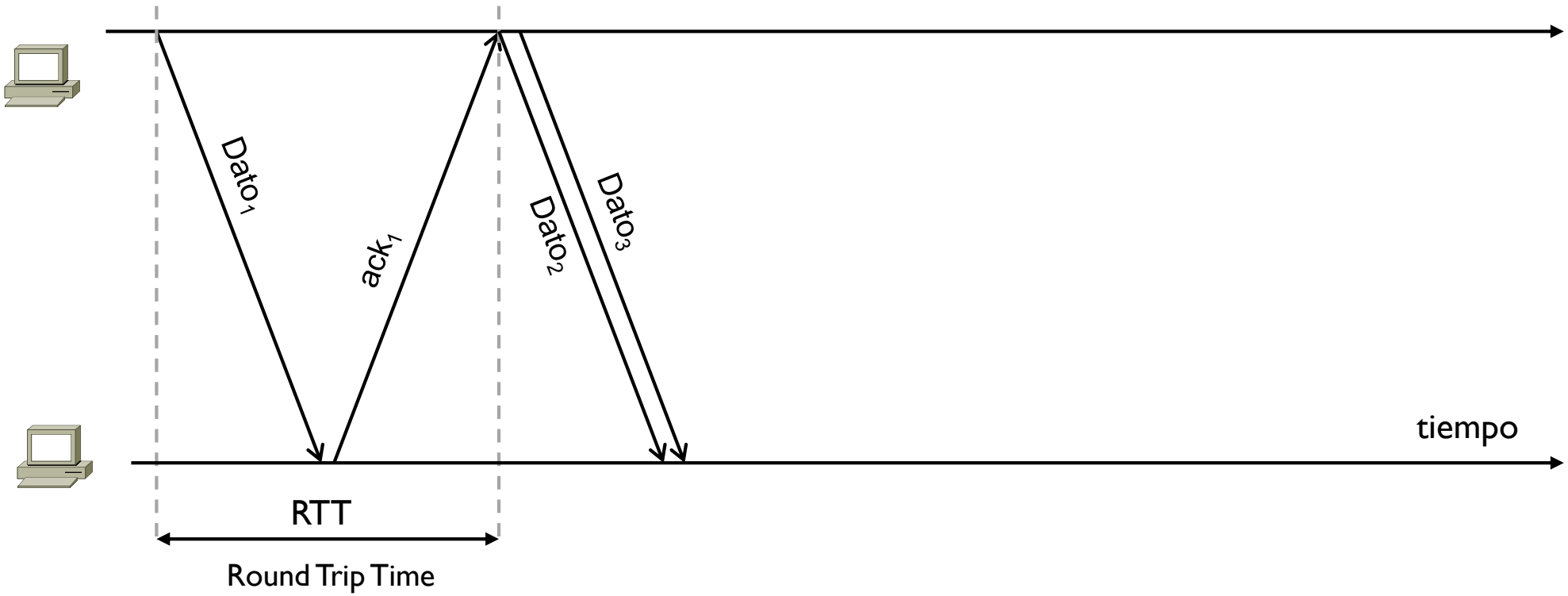
Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

awnd > cwnd → por simplicidad
ssthresh = infinito

Siendo wnd = min(cwnd, awnd) = cwnd = 2 MSS
Se pueden transmitir 2 datos uno seguido del otro

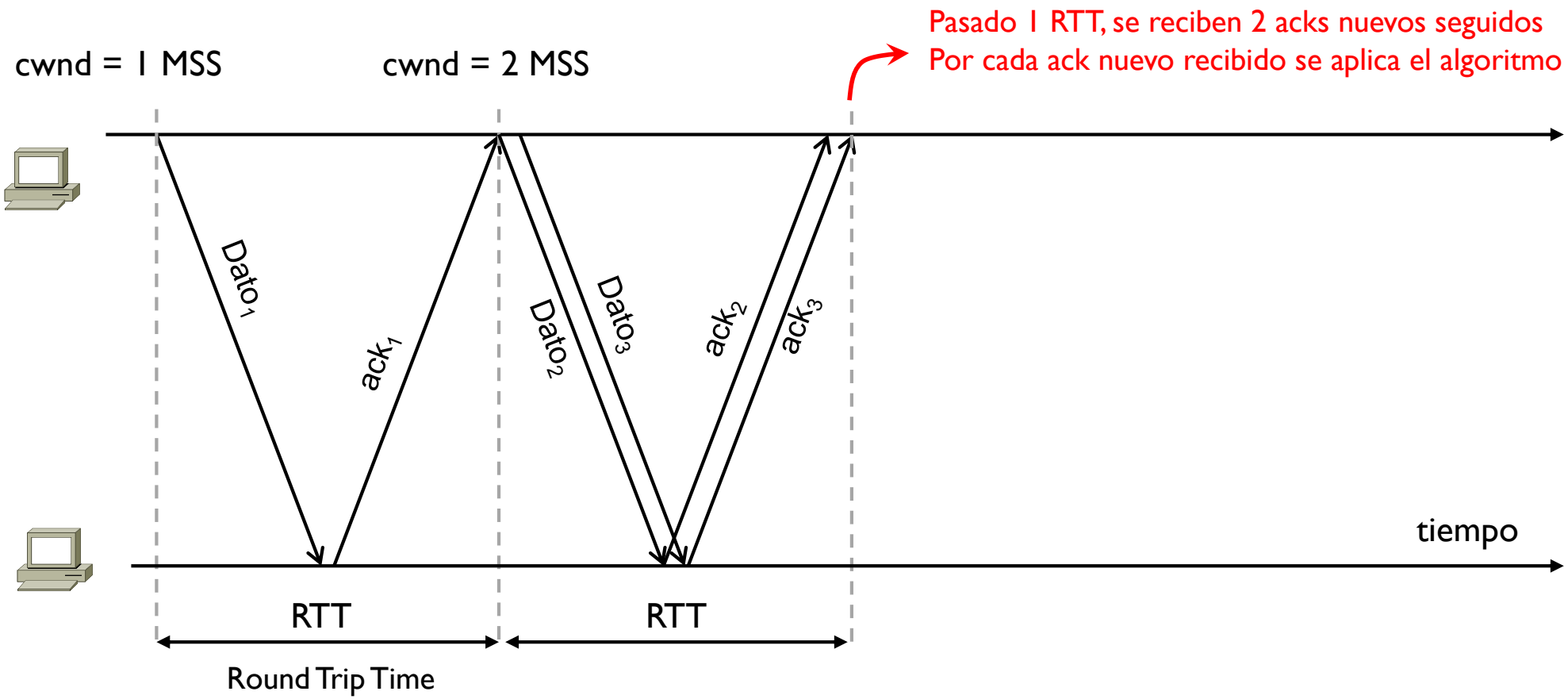


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

awnd > cwnd → por simplicidad
ssthresh = infinito

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

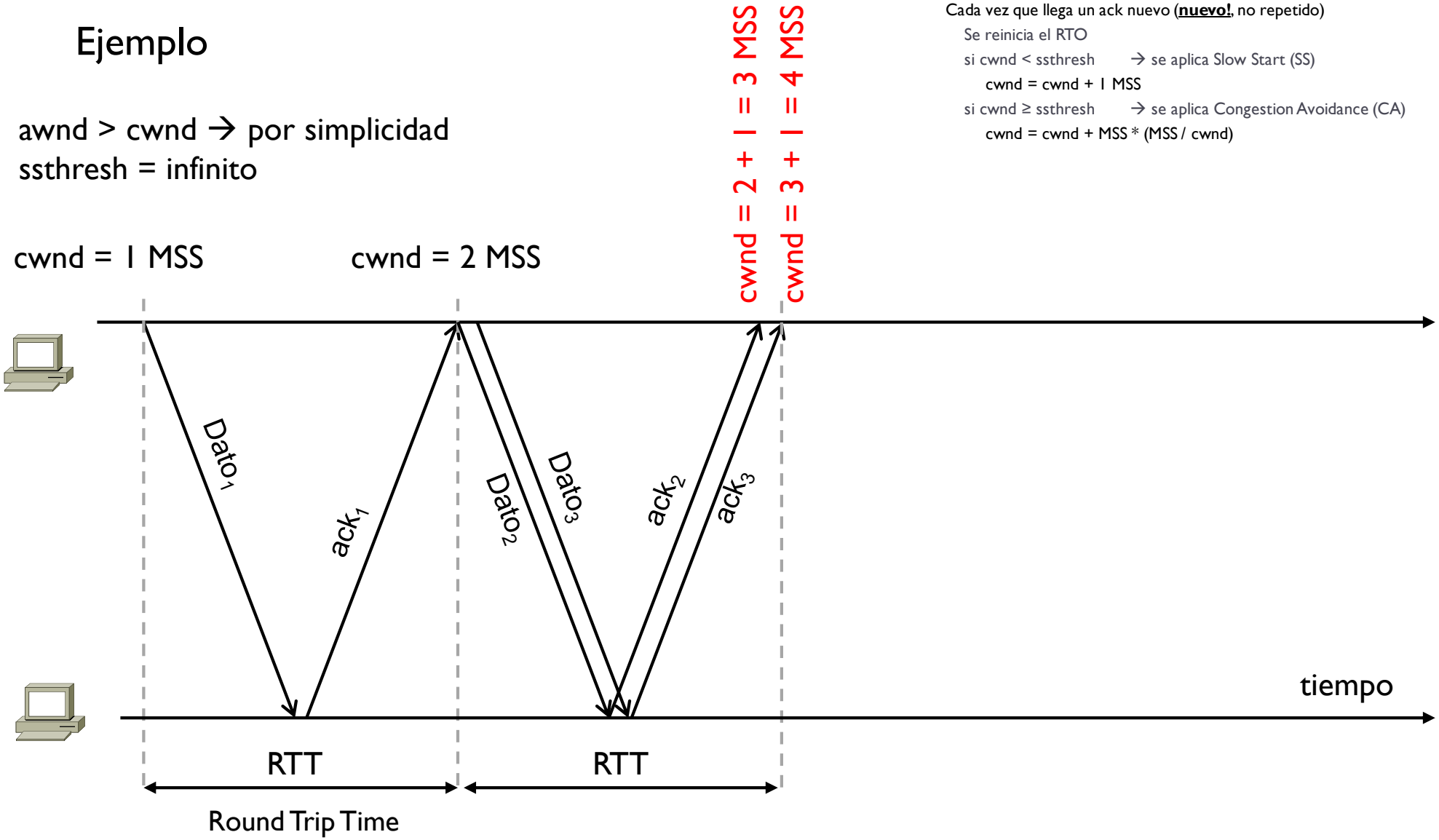


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

awnd > cwnd → por simplicidad
ssthresh = infinito

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

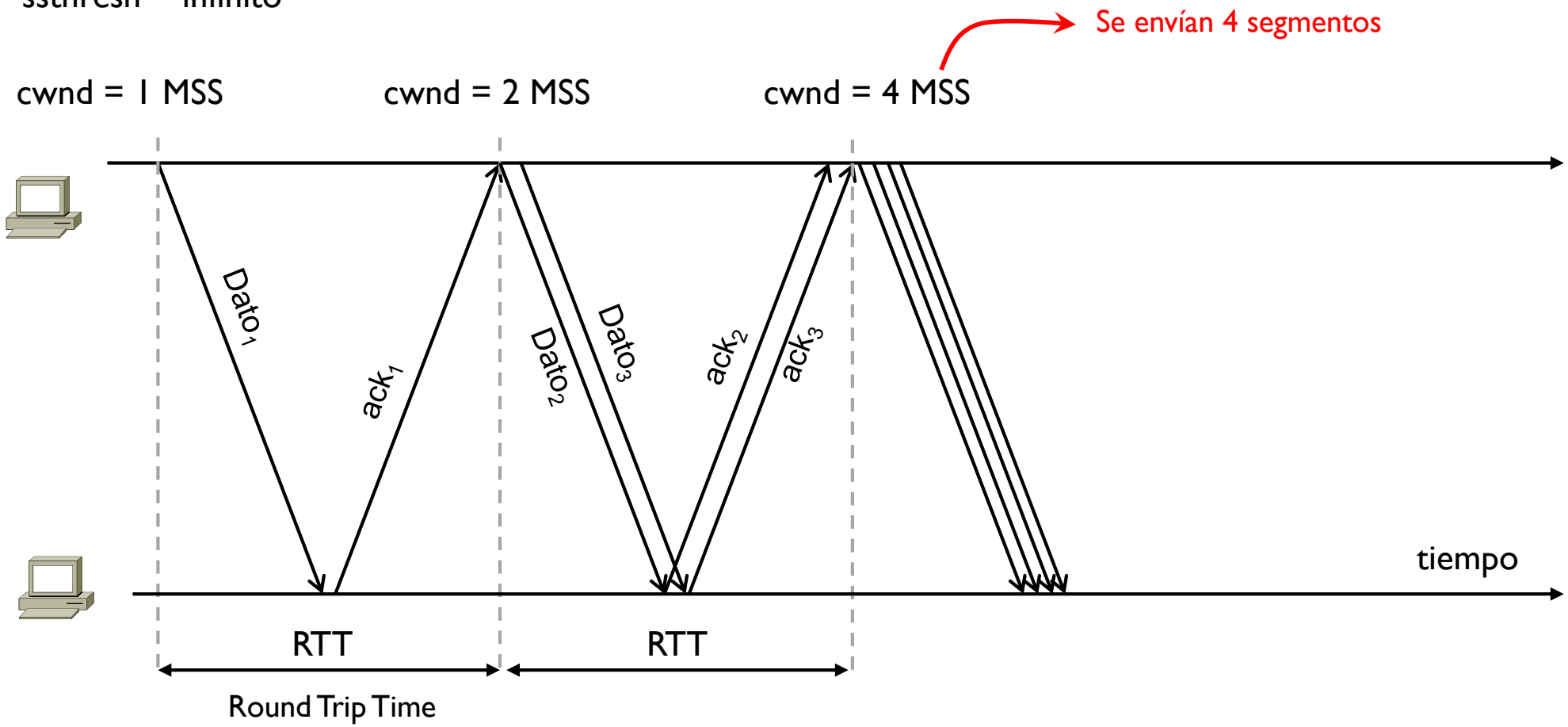


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

awnd > cwnd → por simplicidad
ssthresh = infinito

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)



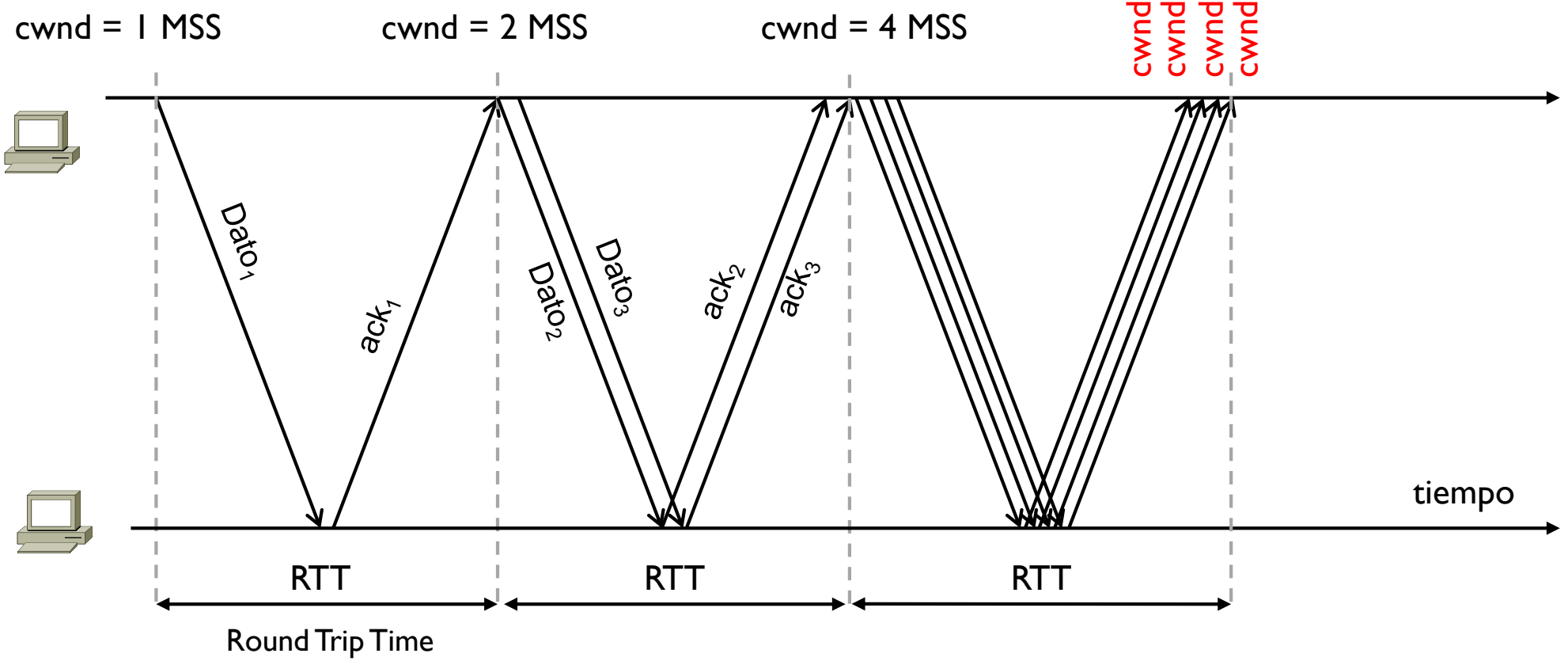
Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

awnd > cwnd → por simplicidad
ssthresh = infinito

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

cwnd = 4 + 1 = 5 MSS
cwnd = 5 + 1 = 6 MSS
cwnd = 6 + 1 = 7 MSS
cwnd = 7 + 1 = 8 MSS

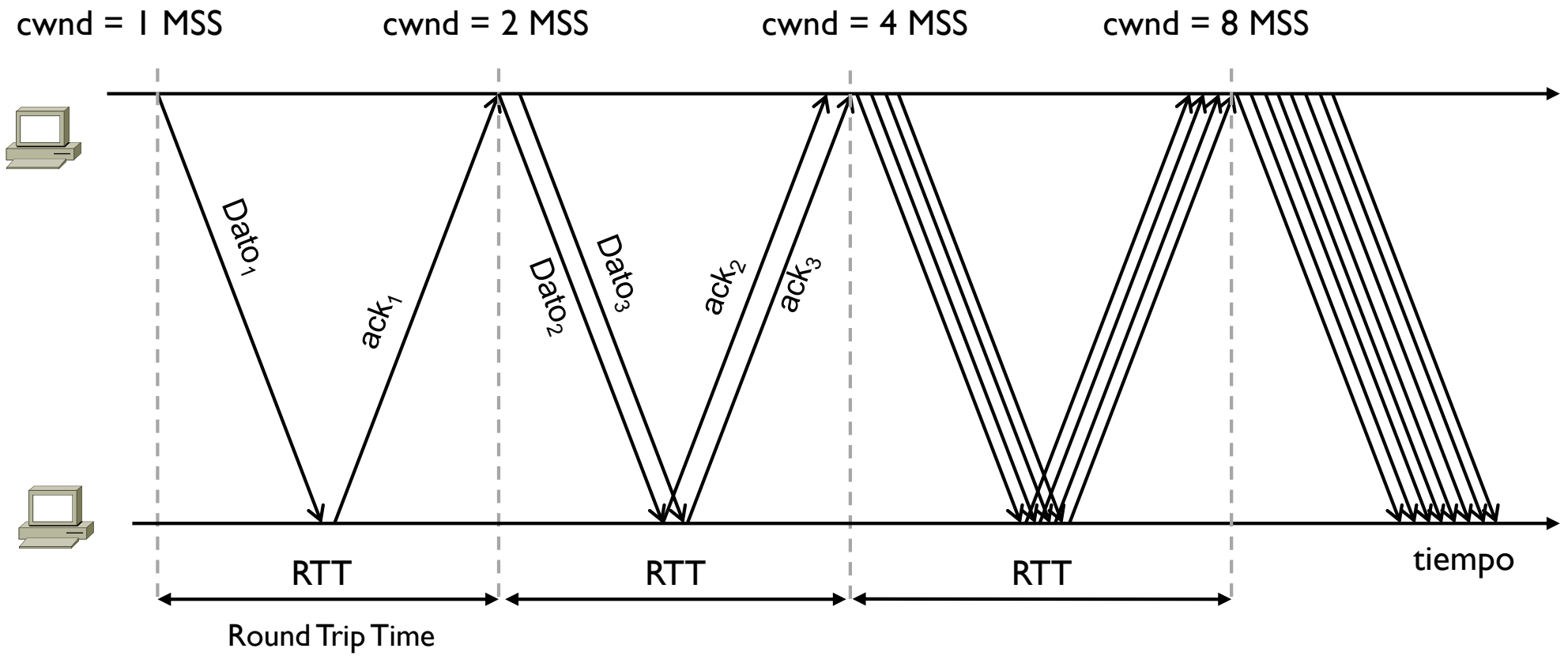


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo

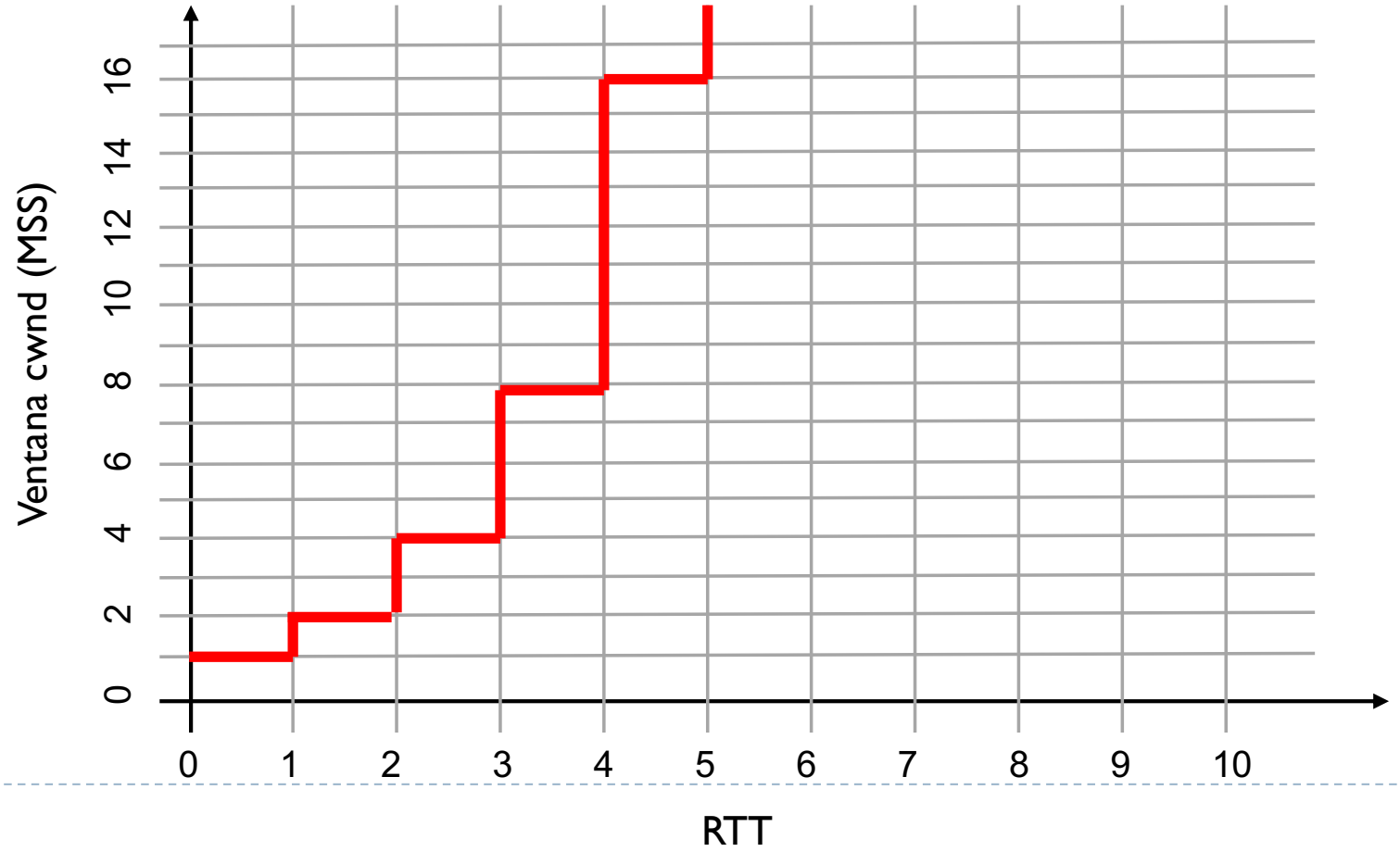
awnd > cwnd → por simplicidad
ssthresh = infinito

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)



Tema 3 – Slow Start + Congestion Avoidance

- Si no hay perdidas, solo se aplica SS ya que ssthresh se mantiene a infinito
- SS hace que cada vez que pasa un RTT (se envían tantos datos cuanto una ventana entera) → la ventana cwnd se duplica cada RTT



Tema 3 – Slow Start + Congestion Avoidance

- ¿Puede cwnd subir infinitamente?

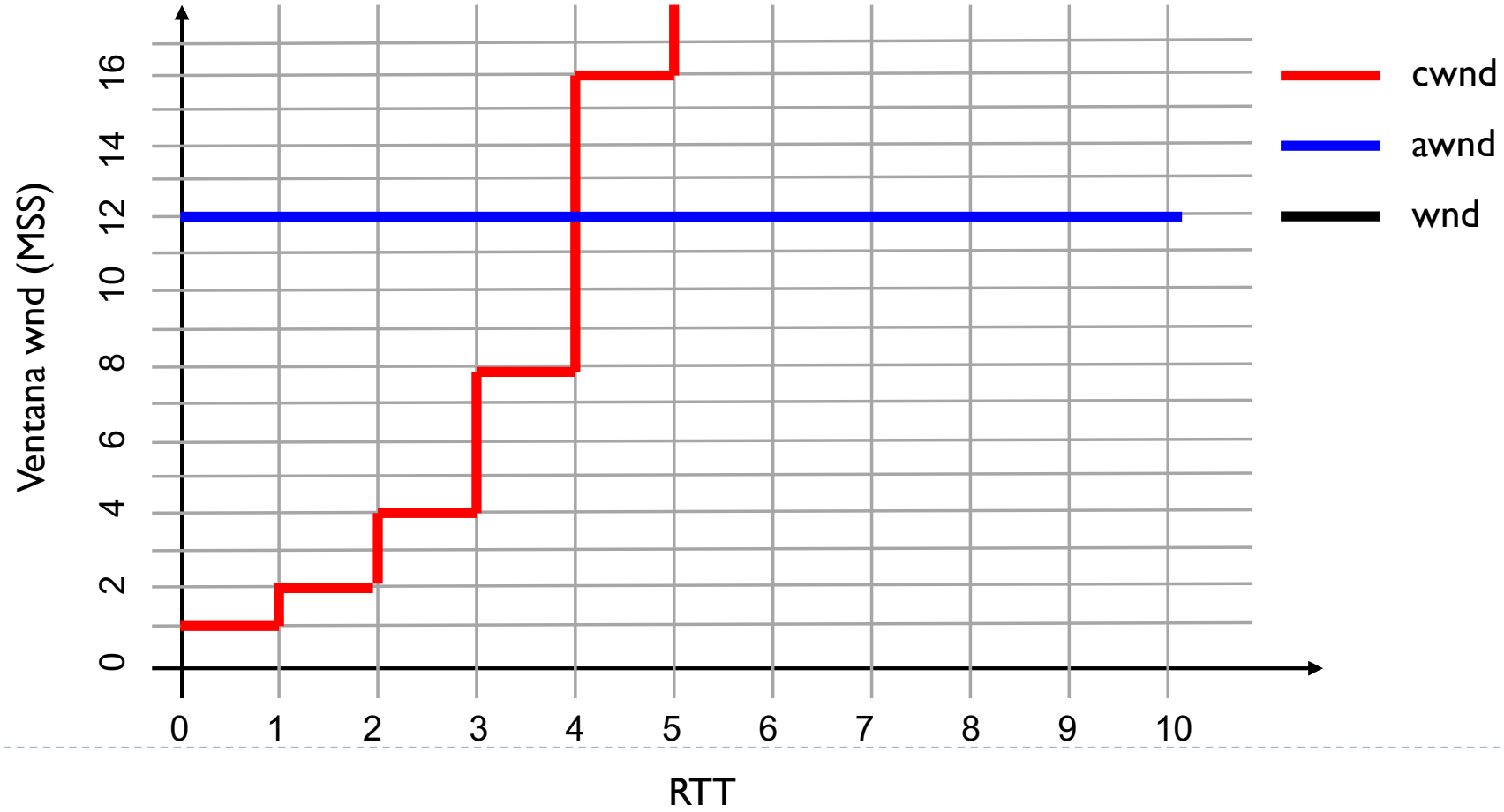
  No, su valor máximo es generalmente 65535 bytes

- Además hay que tener en cuenta que lo que se transmite (ventana wnd) depende de cwnd y también de awnd

  awnd es generalmente un valor bastante estable

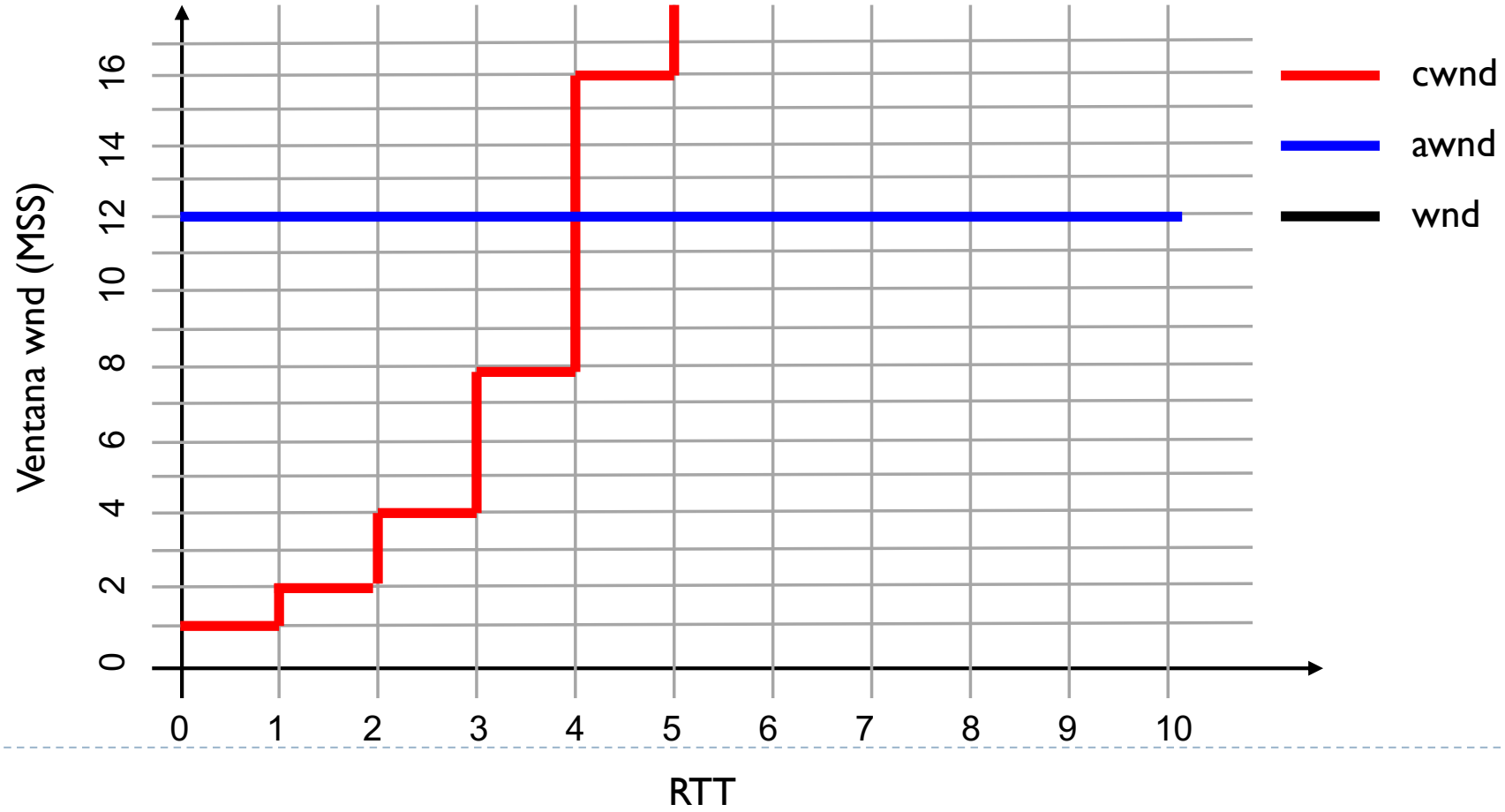
Tema 3 – Slow Start + Congestion Avoidance

- En resumen, si no hay perdidas, generalmente el TCP se comporta de esta manera



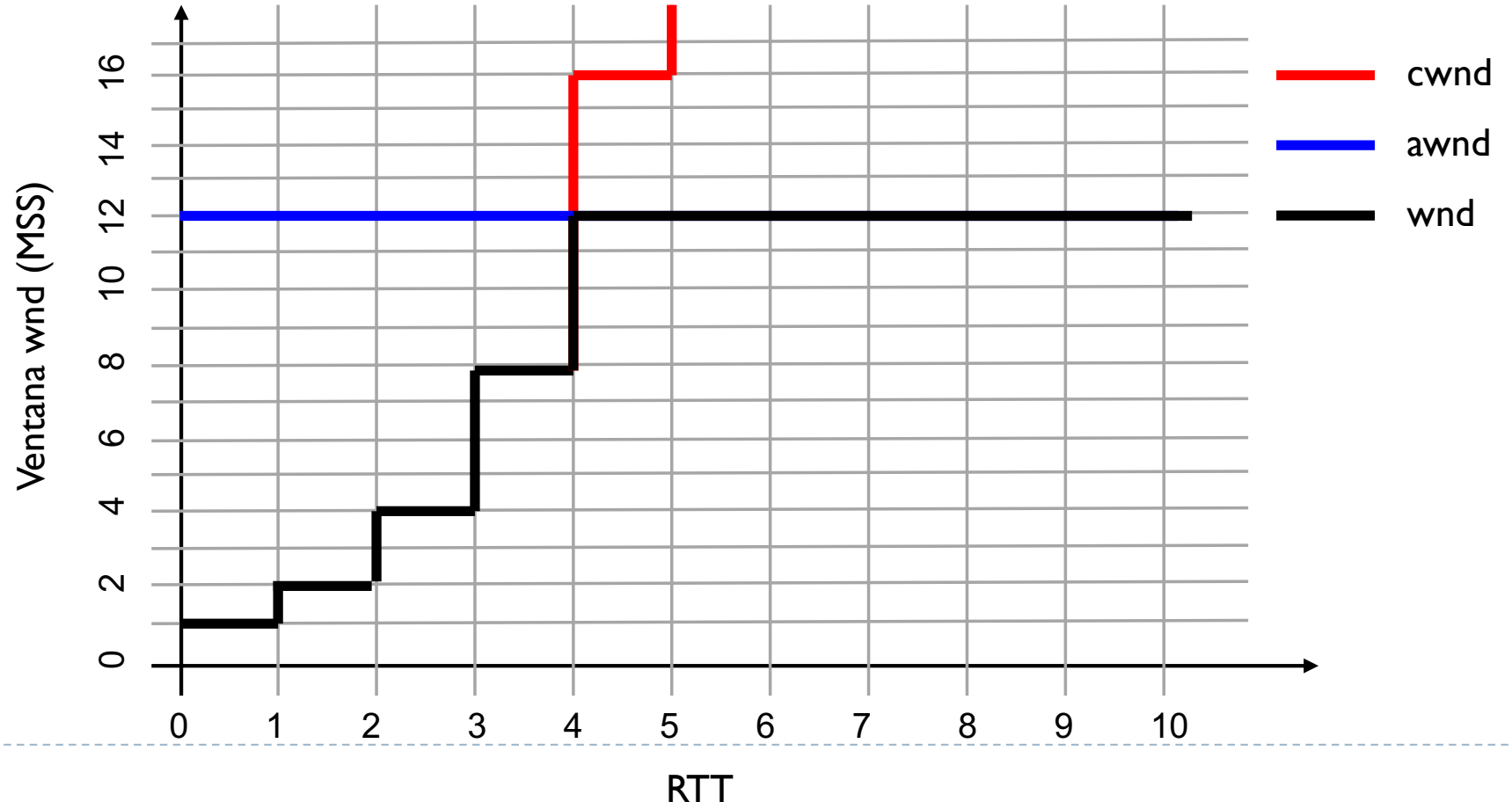
Tema 3 – Slow Start + Congestion Avoidance

- En resumen, si no hay perdidas, generalmente el TCP se comporta de esta manera
- ¿Cuanto vale la ventana de transmision wnd?



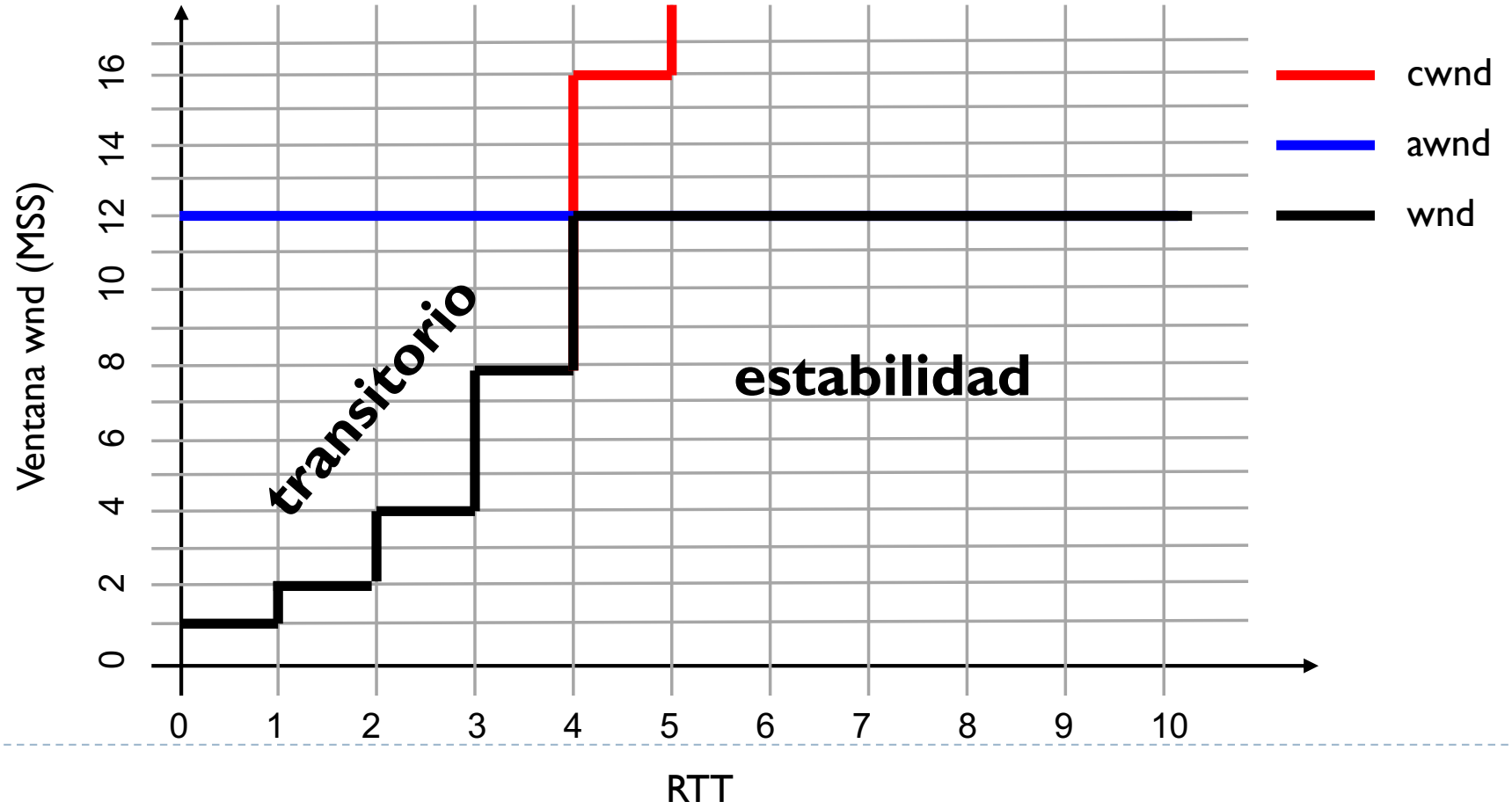
Tema 3 – Slow Start + Congestion Avoidance

- En resumen, si no hay perdidas, generalmente el TCP se comporta de esta manera
- La ventana wnd aumenta según cwnd hasta llegar a awnd, luego se estabiliza a este valor



Tema 3 – Slow Start + Congestion Avoidance

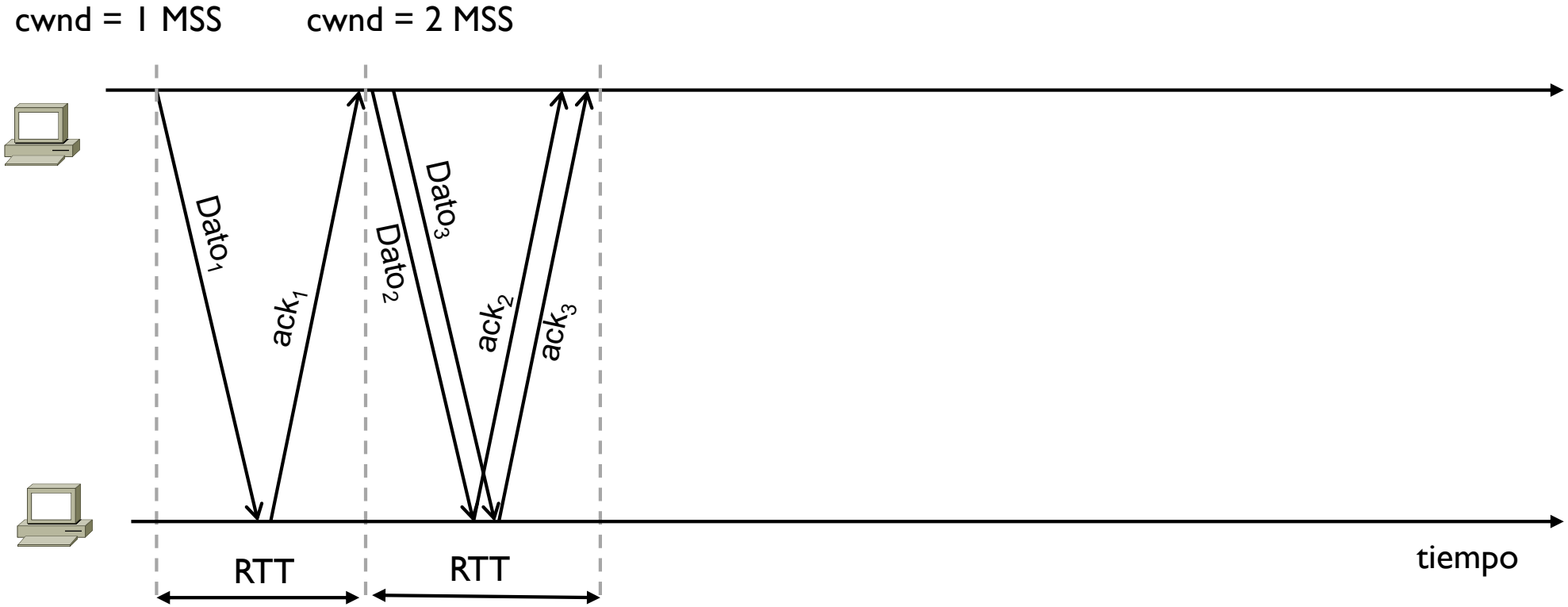
- En resumen, si no hay perdidas, generalmente el TCP se comporta de esta manera
- La ventana wnd aumenta según cwnd hasta llegar a awnd, luego se estabiliza a este valor



Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

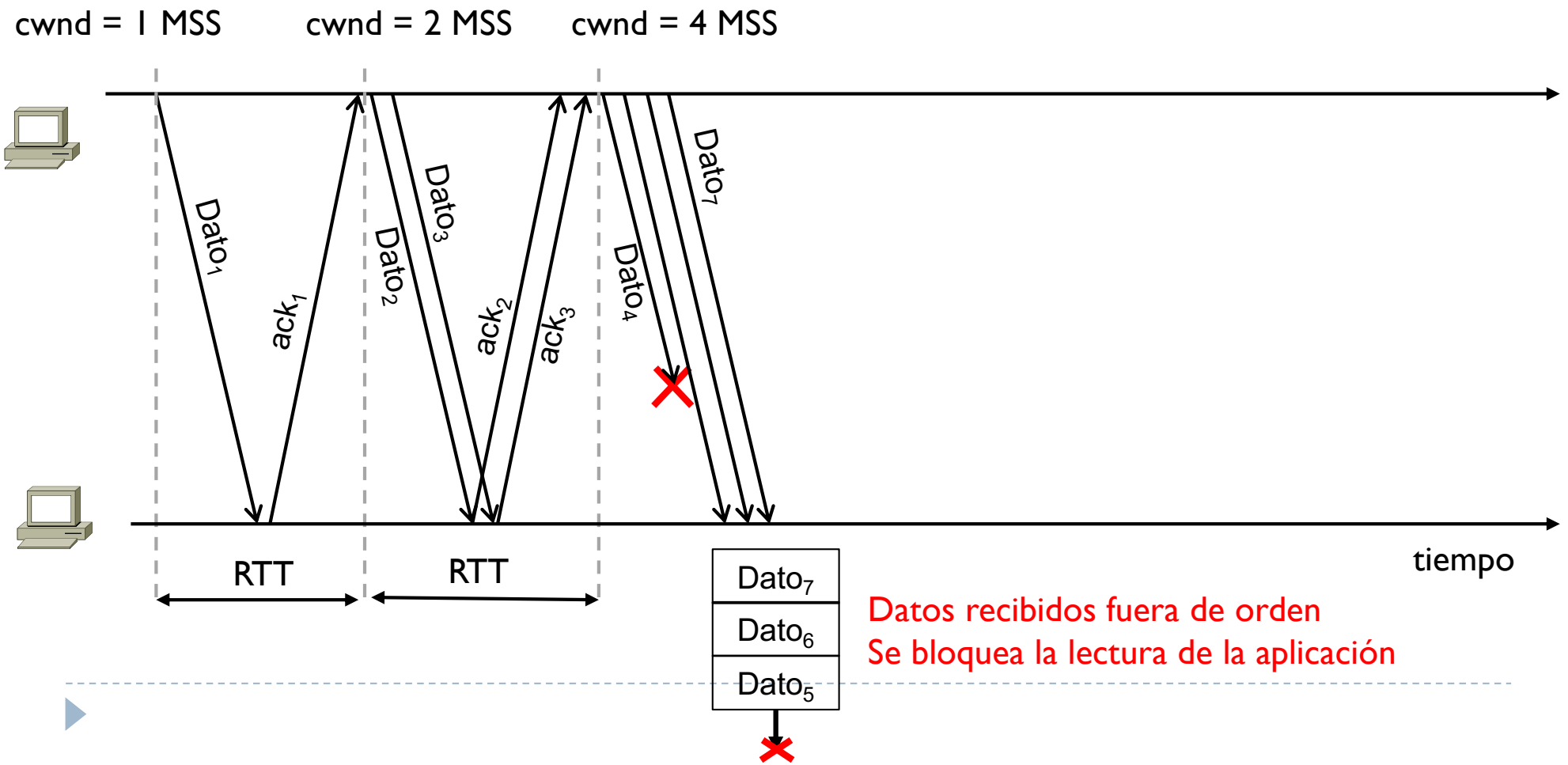
awnd > cwnd → por simplicidad
ssthresh = infinito



Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

awnd > cwnd → por simplicidad
ssthresh = infinito

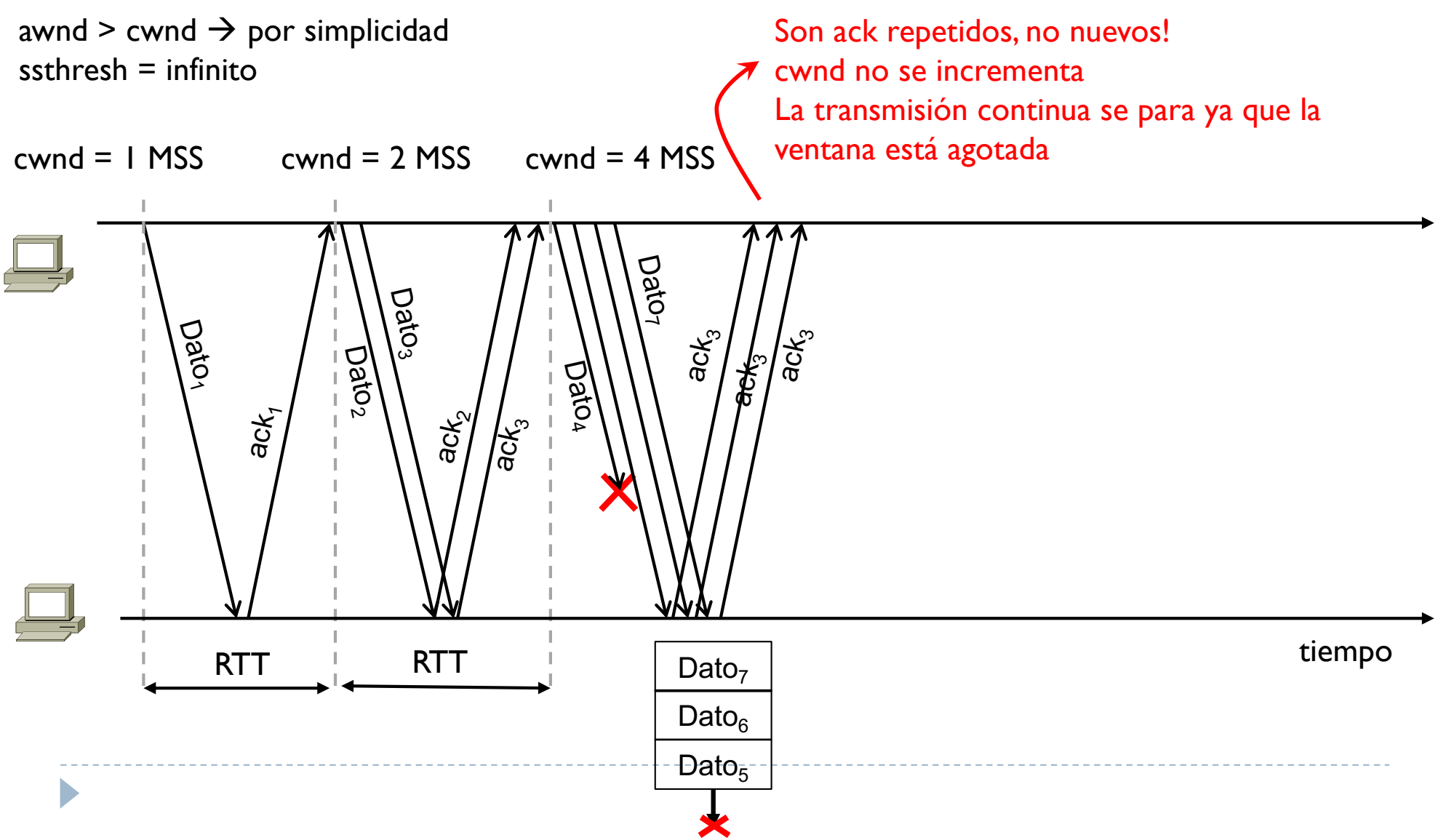


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

awnd > cwnd → por simplicidad
ssthresh = infinito

Son ack repetidos, no nuevos!
cwnd no se incrementa
La transmisión continua se para ya que la ventana está agotada

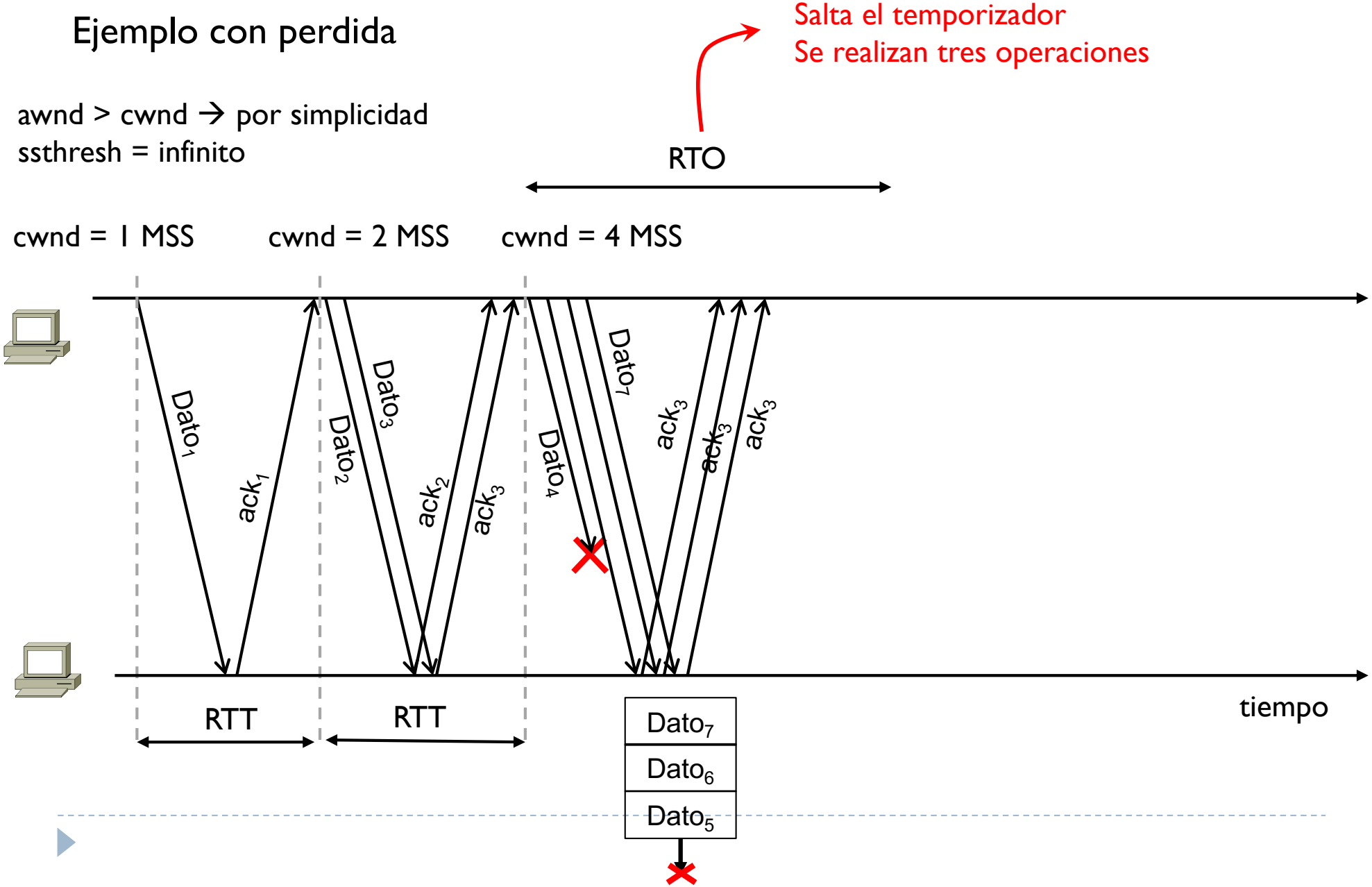


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

Salta el temporizador
Se realizan tres operaciones

awnd > cwnd → por simplicidad
ssthresh = infinito



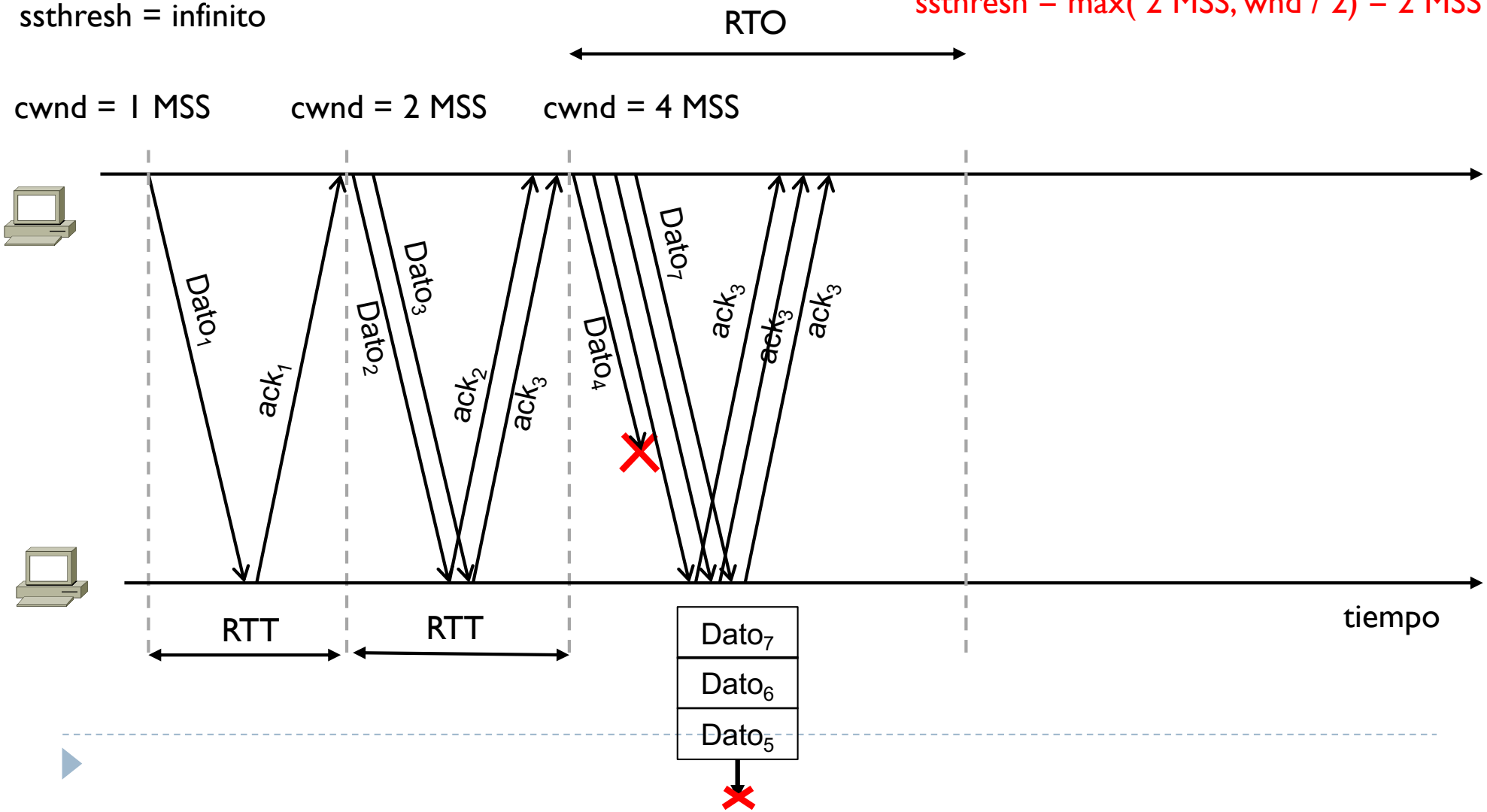
Tema 3 – Slow Start + Congestion Avoidance

(1)

Ejemplo con perdida

awnd > cwnd → por simplicidad
ssthresh = infinito

se calcula un nuevo valor de sshthresh
ssthresh = max(2 MSS, wnd / 2) = 2 MSS



Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

(2)

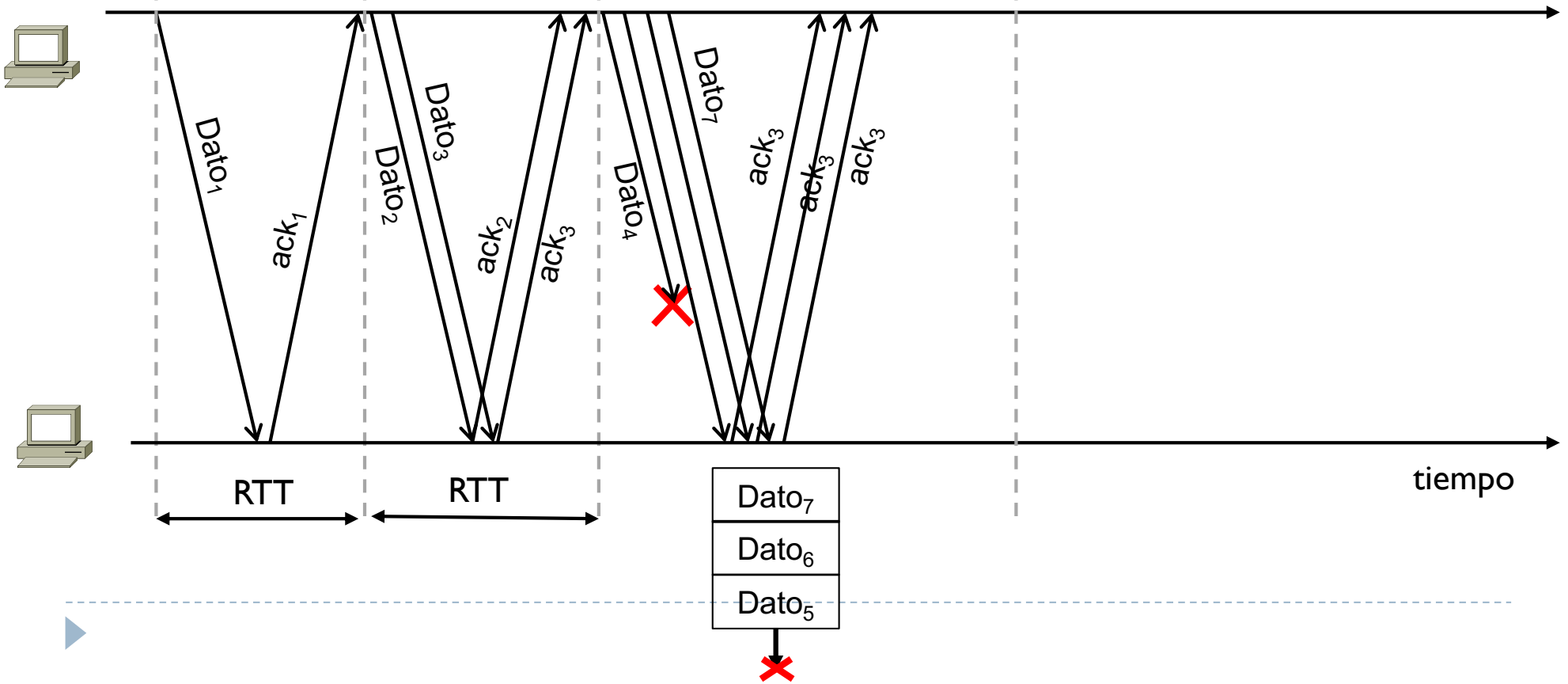
cwnd vuelve a empezar con 1 MSS

awnd > cwnd → por simplicidad
ssthresh = infinito

RTO

ssthresh = 2 MSS

cwnd = 1 MSS | cwnd = 2 MSS | cwnd = 4 MSS | cwnd = 1 MSS



Tema 3 – Slow Start + Congestion Avoidance

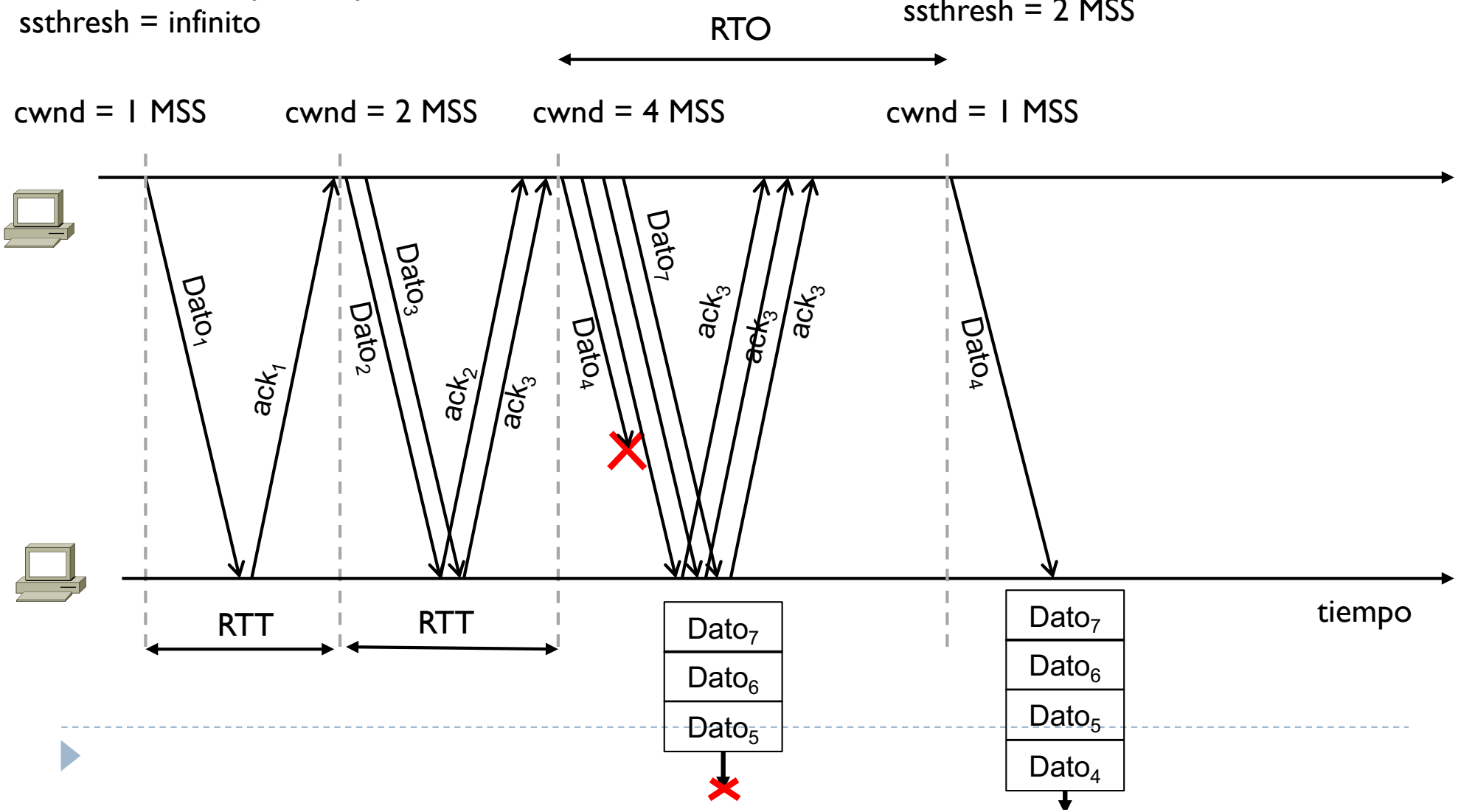
(3)

Ejemplo con perdida

awnd > cwnd → por simplicidad
ssthresh = infinito

Se retransmite el dato perdido

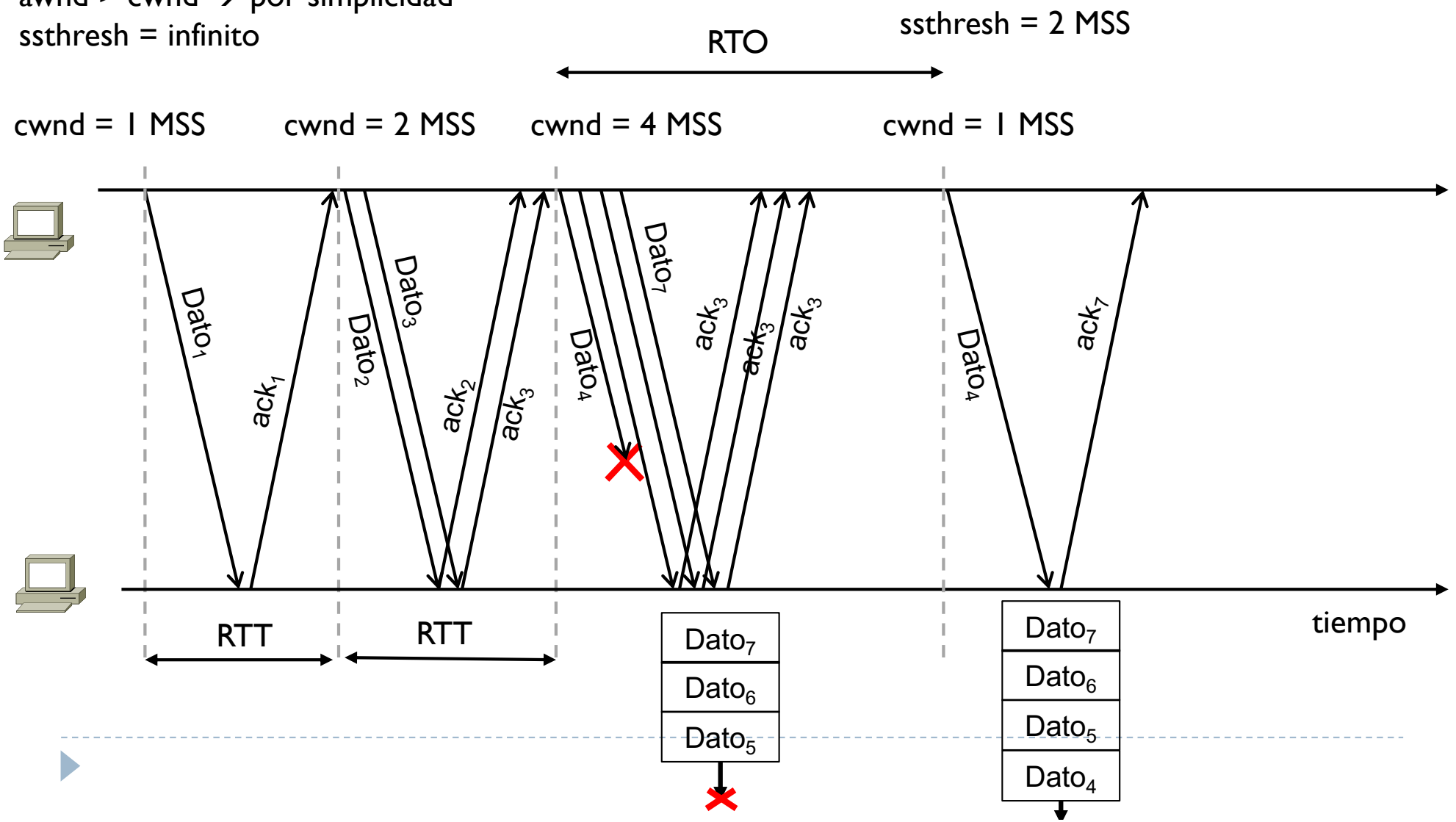
ssthresh = 2 MSS



Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

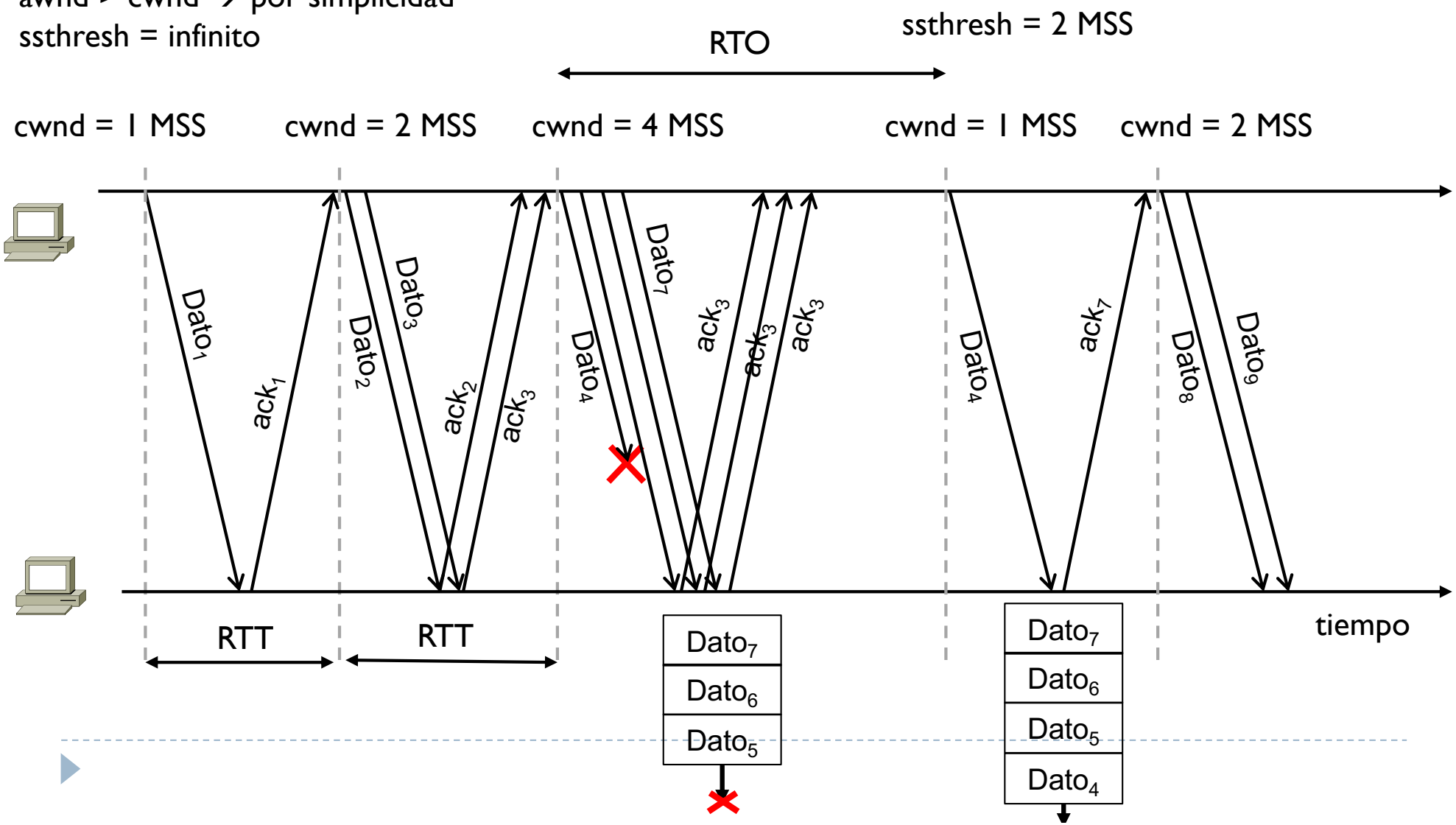
awnd > cwnd → por simplicidad
ssthresh = infinito



Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

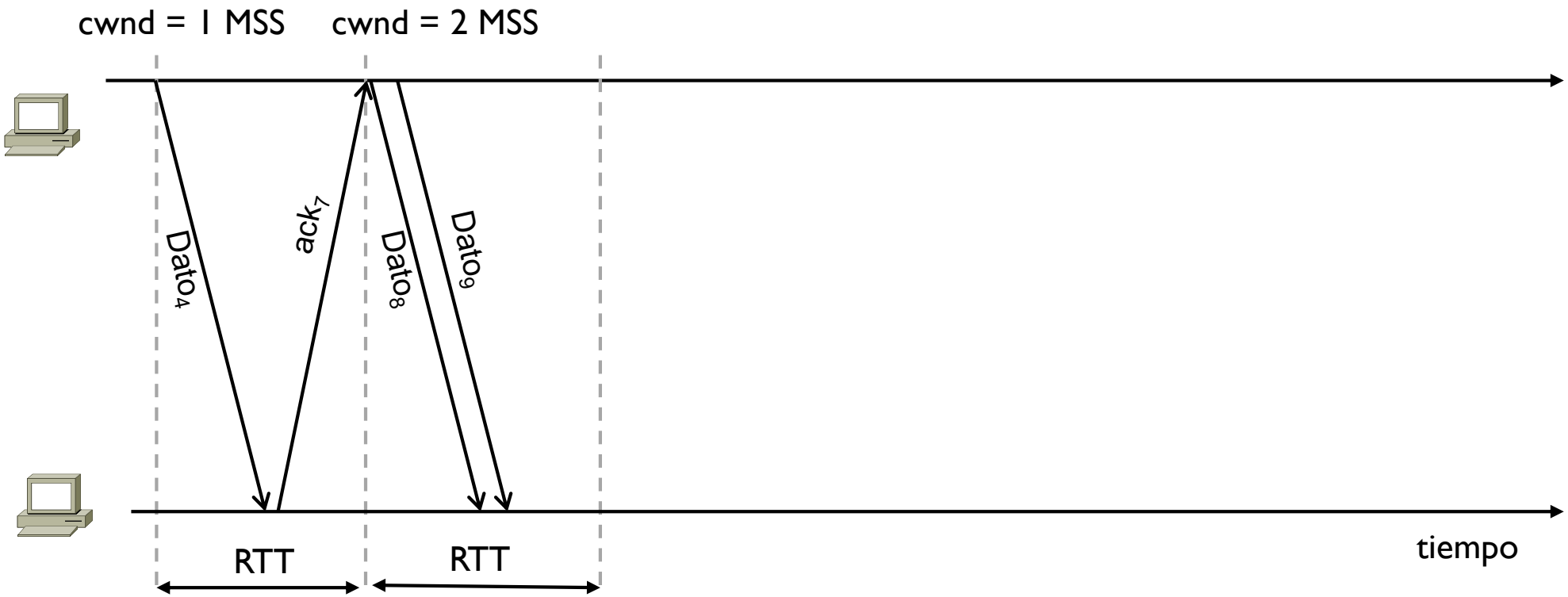
awnd > cwnd → por simplicidad
ssthresh = infinito



Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

awnd > cwnd → por simplicidad
ssthresh = 2 MSS



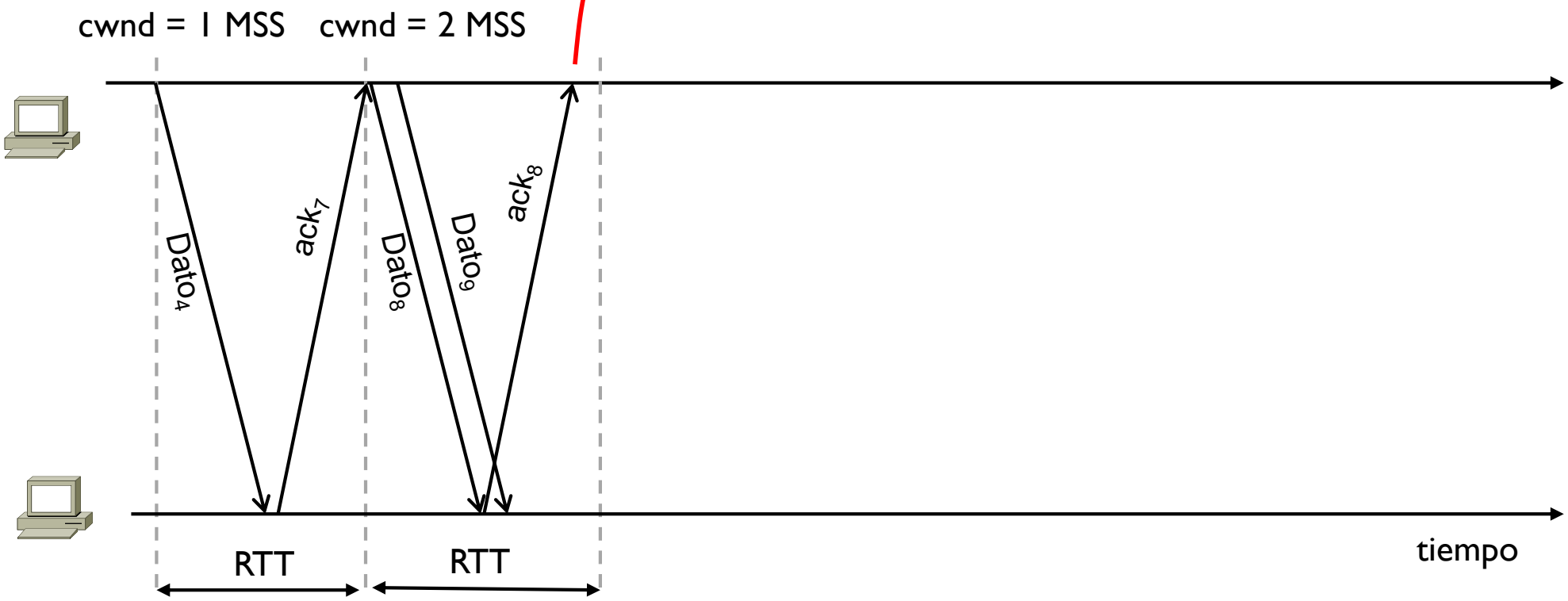
Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

awnd > cwnd → por simplicidad
ssthresh = 2 MSS

Se recibe un ack nuevo
Ahora pero cwnd = ssthresh → hay que aplicar CA
cwnd = 2 MSS + MSS * (MSS / 2 MSS) = 2.5 MSS

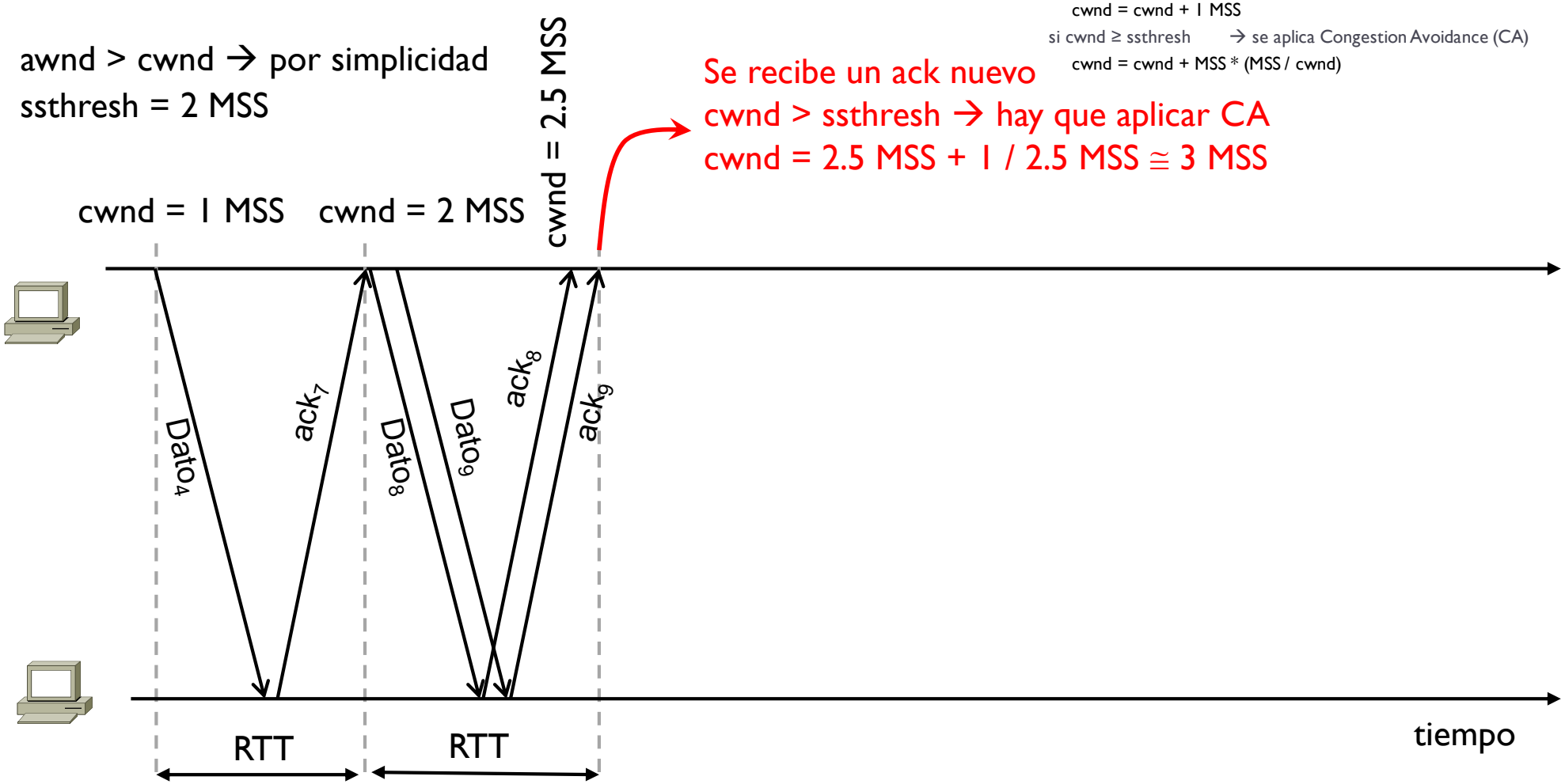


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

Cada vez que llega un ack nuevo (**nuevo!**, no repetido)
- Se reinicia el RTO
- si cwnd < ssthresh → se aplica Slow Start (SS)
  - cwnd = cwnd + 1 MSS
- si cwnd ≥ ssthresh → se aplica Congestion Avoidance (CA)
  - cwnd = cwnd + MSS * (MSS / cwnd)

awnd > cwnd → por simplicidad
ssthresh = 2 MSS

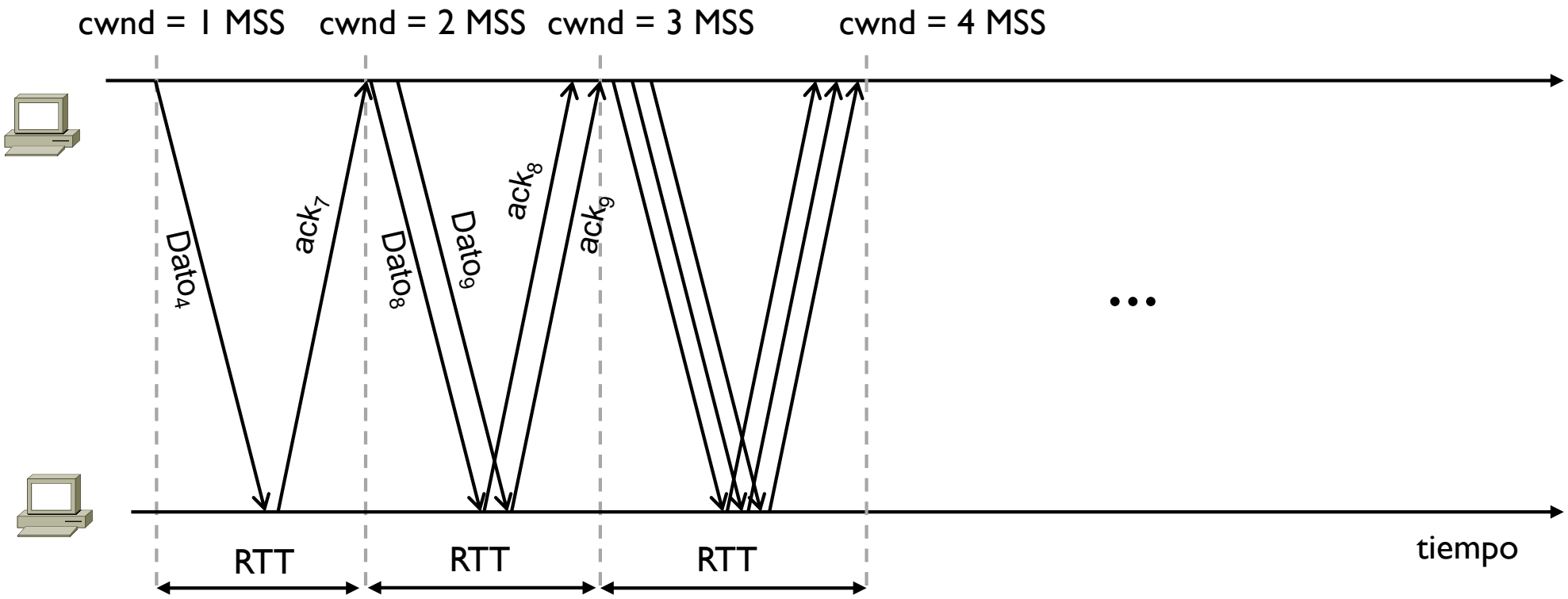


Tema 3 – Slow Start + Congestion Avoidance

Ejemplo con perdida

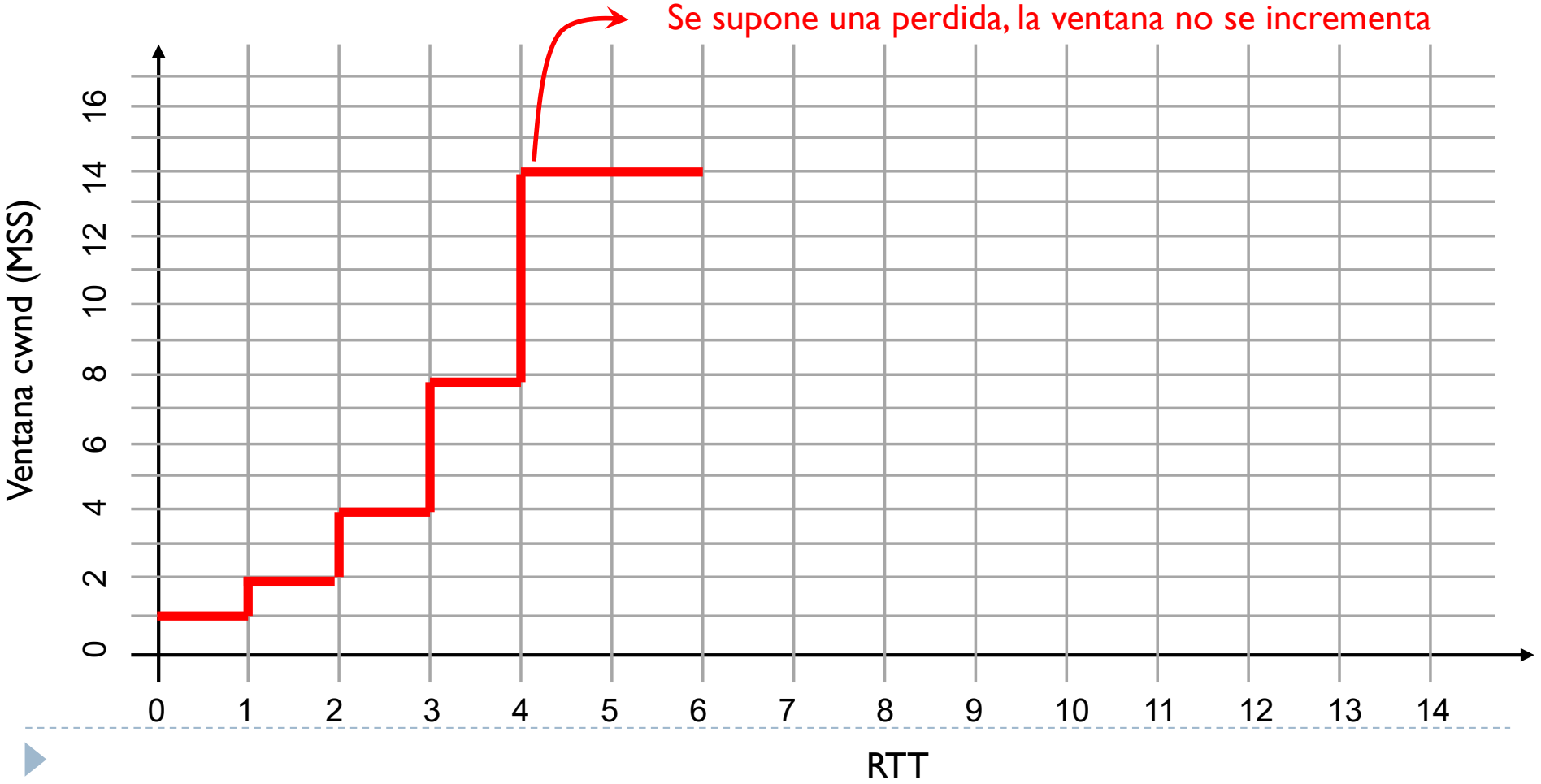
awnd > cwnd → por simplicidad
ssthresh = 2 MSS

Se envian 3 segmentos, se reciben 3 ack, cada uno más o menos incrementa 1/3, la ventana sube a 4



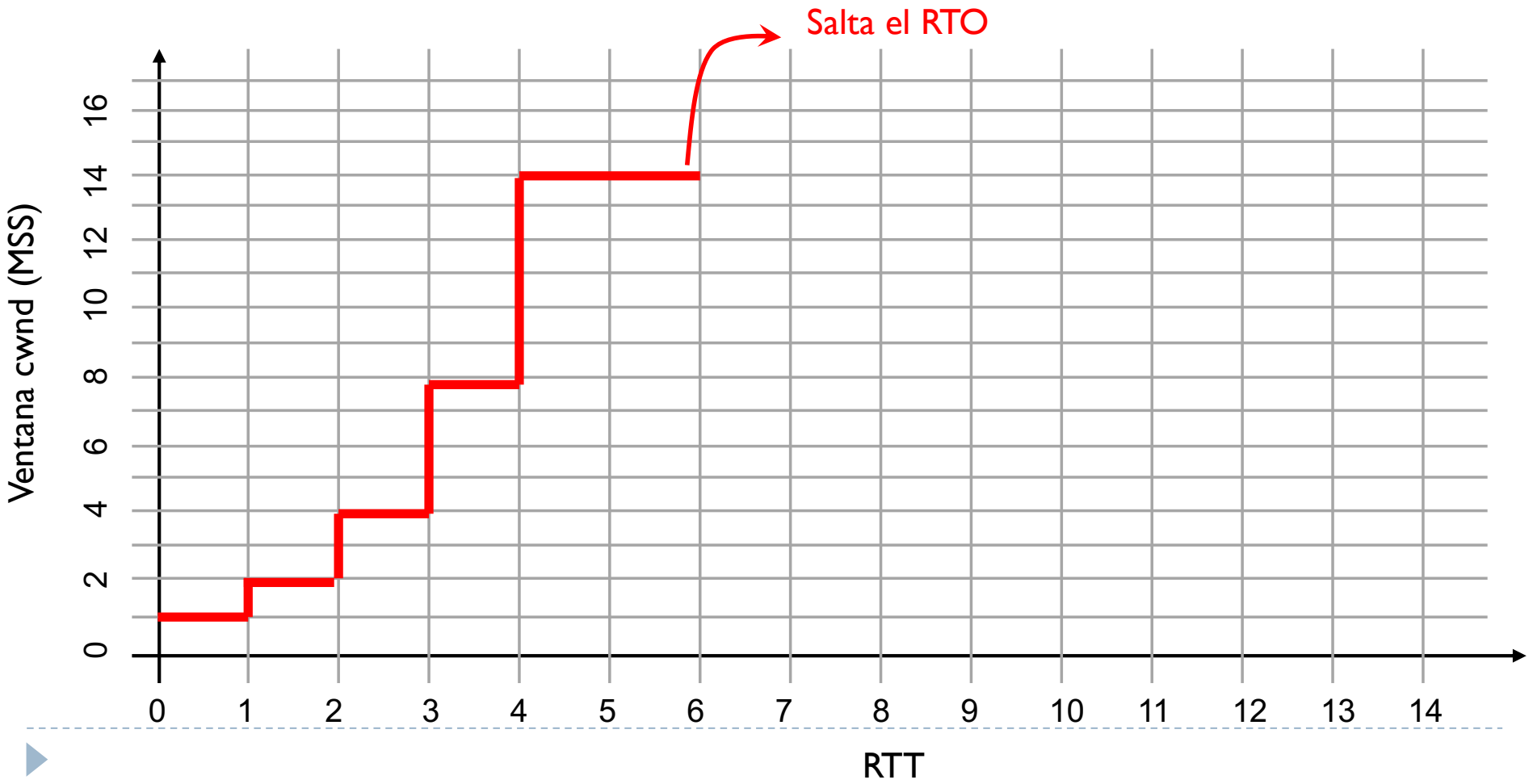
Tema 3 – Slow Start + Congestion Avoidance

- Si hay perdidas, se determina un nuevo valor por ssthresh
- Por debajo de este valor se aplica SS, por encima se aplica CA
- CA incrementa la ventana cwnd de 1 MSS por cada RTT



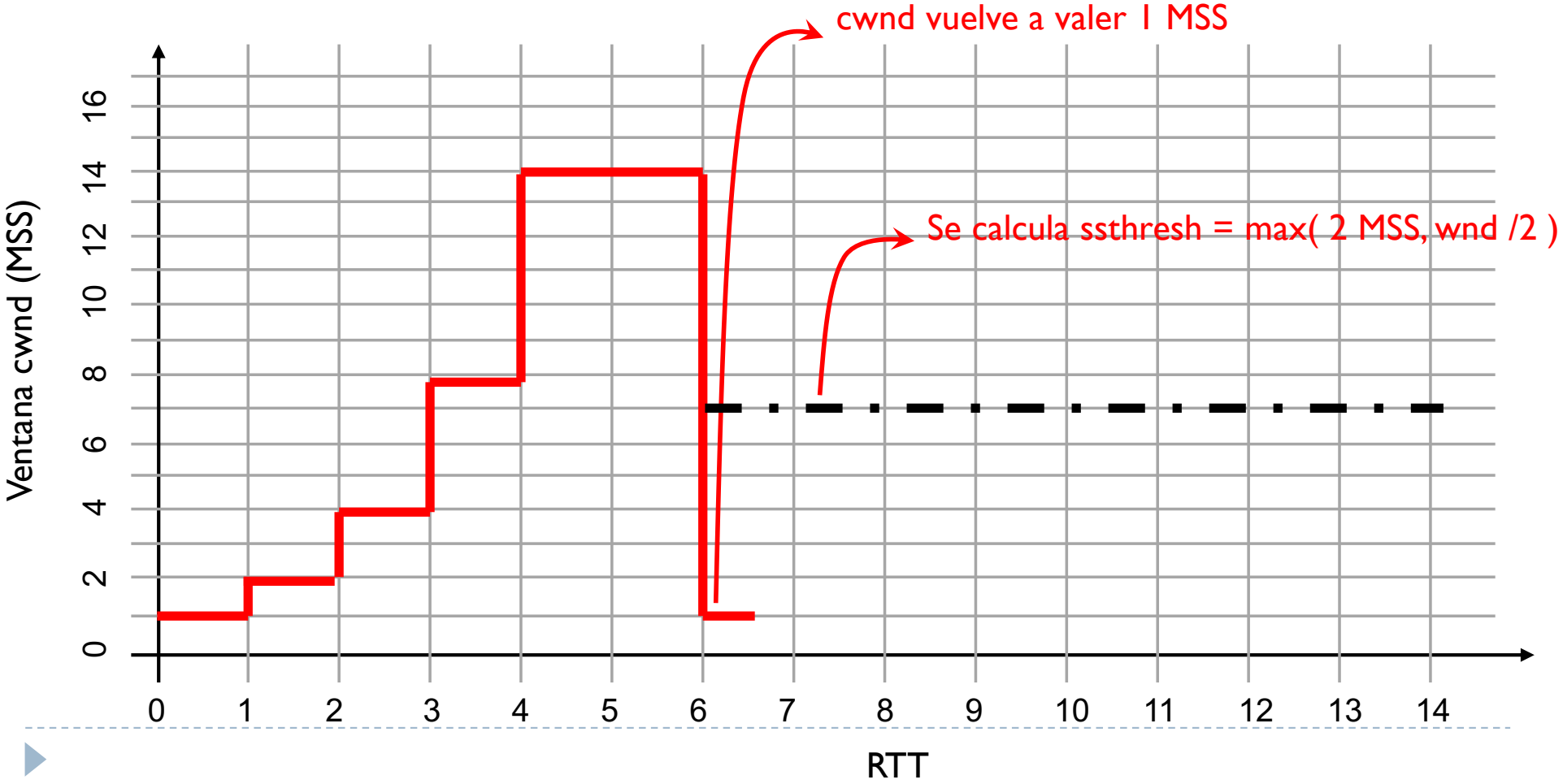
Tema 3 – Slow Start + Congestion Avoidance

- Si hay perdidas, se determina un nuevo valor por ssthresh
- Por debajo de este valor se aplica SS, por encima se aplica CA
- CA incrementa la ventana cwnd de 1 MSS por cada RTT



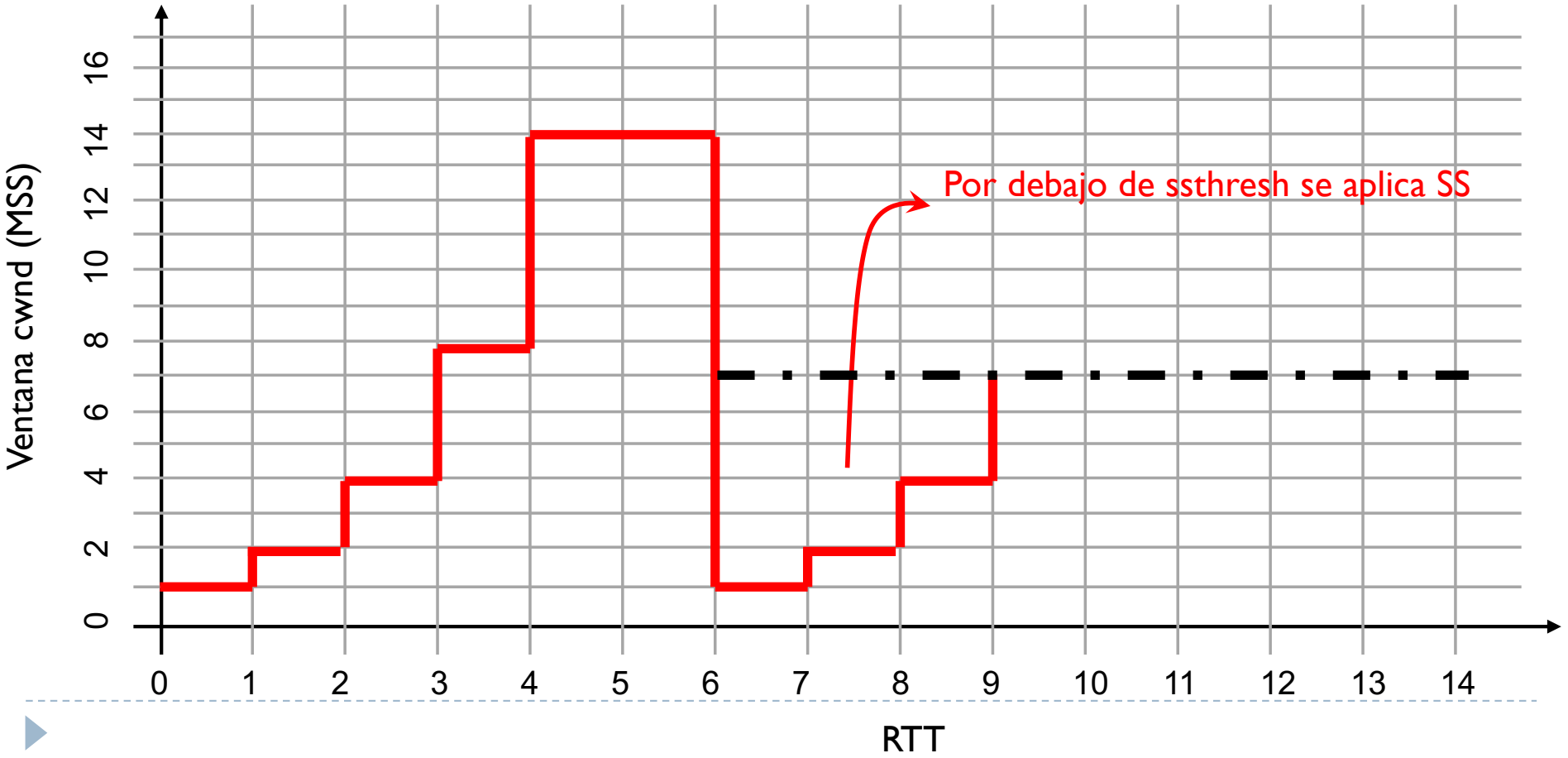
Tema 3 – Slow Start + Congestion Avoidance

- Si hay perdidas, se determina un nuevo valor por ssthresh
- Por debajo de este valor se aplica SS, por encima se aplica CA
- CA incrementa la ventana cwnd de 1 MSS por cada RTT



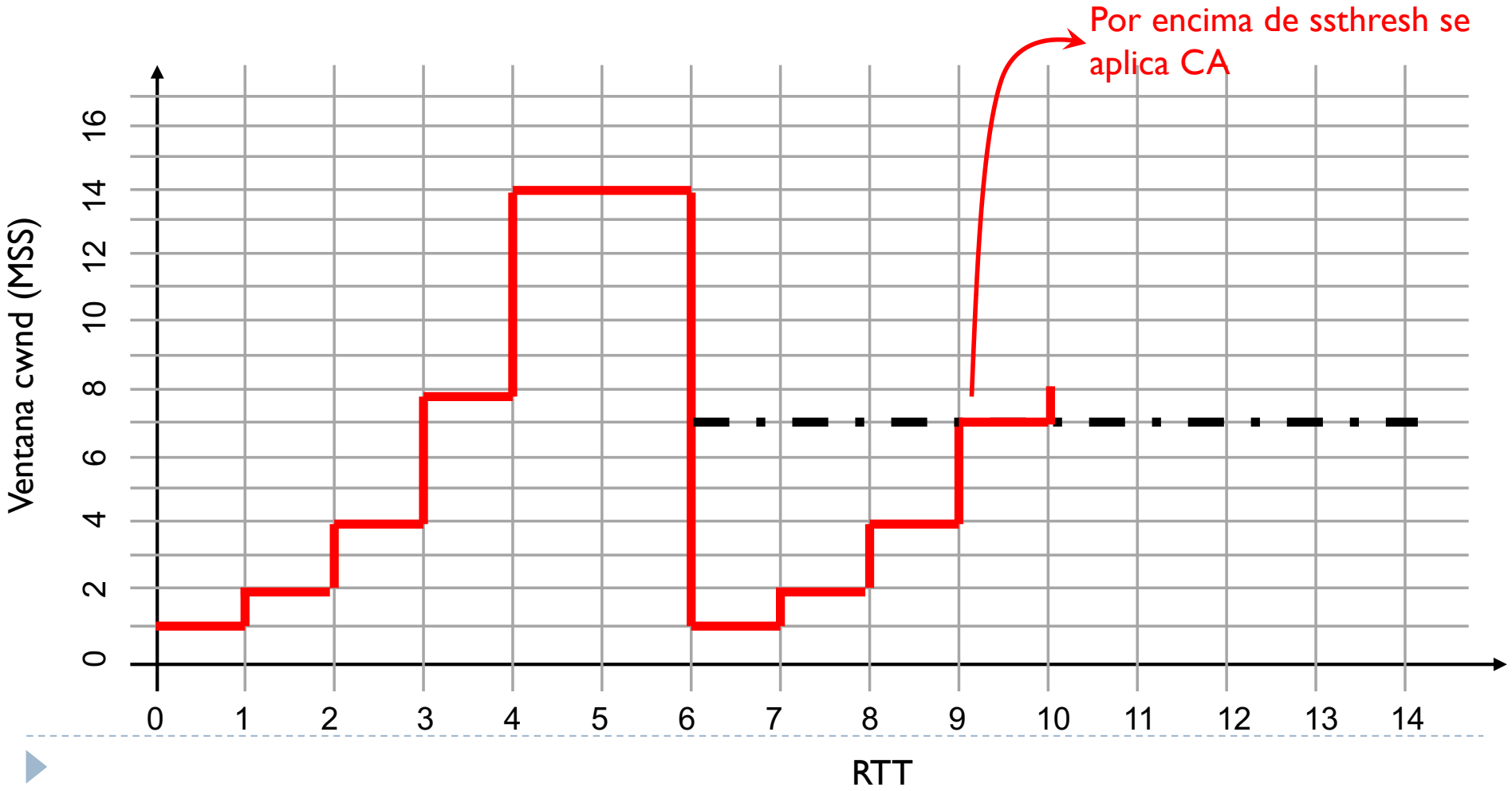
Tema 3 – Slow Start + Congestion Avoidance

- Si hay perdidas, se determina un nuevo valor por ssthresh
- Por debajo de este valor se aplica SS, por encima se aplica CA
- CA incrementa la ventana cwnd de 1 MSS por cada RTT



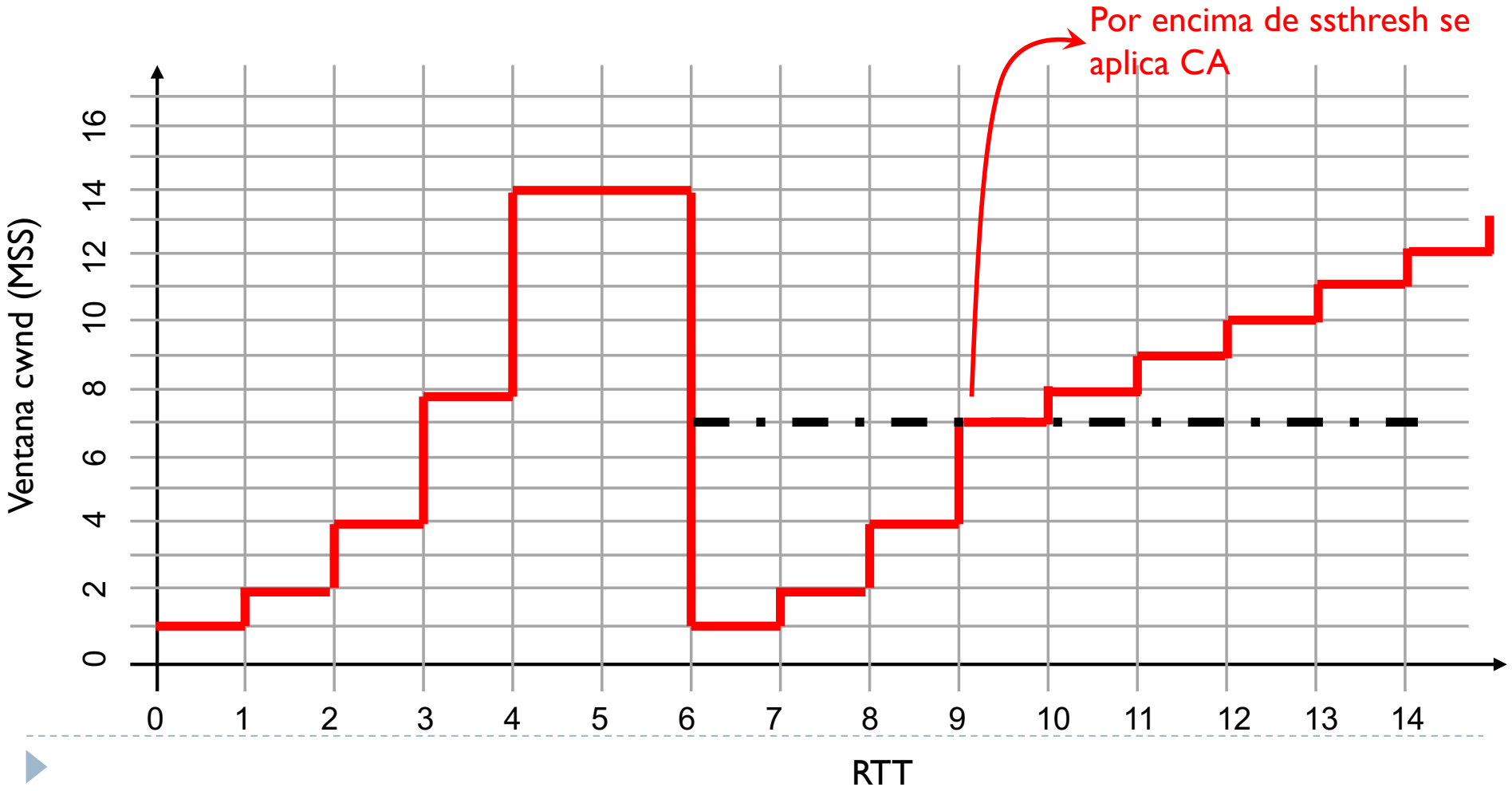
Tema 3 – Slow Start + Congestion Avoidance

- Si hay perdidas, se determina un nuevo valor por ssthresh
- Por debajo de este valor se aplica SS, por encima se aplica CA
- CA incrementa la ventana cwnd de 1 MSS por cada RTT



Tema 3 – Slow Start + Congestion Avoidance

- Si hay perdidas, se determina un nuevo valor por ssthresh
- Por debajo de este valor se aplica SS, por encima se aplica CA
- CA incrementa la ventana cwnd de 1 MSS por cada RTT



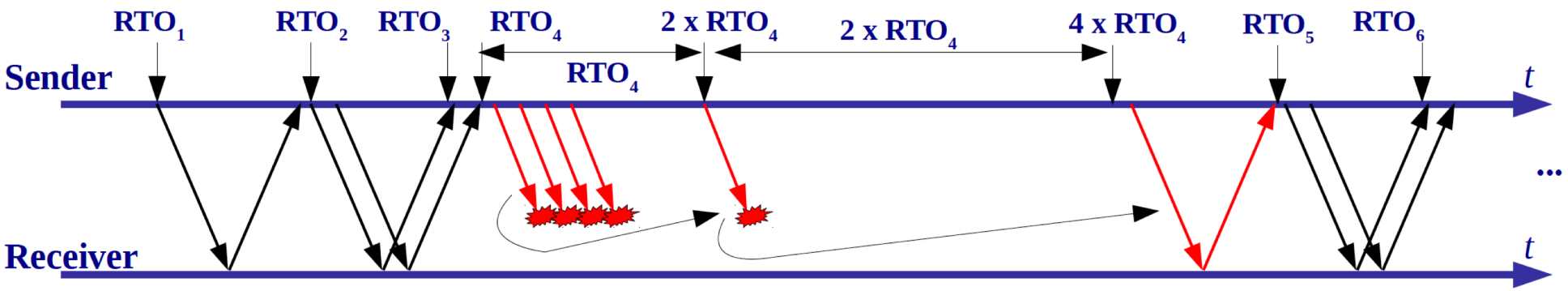
Tema 3 – Slow Start + Congestion Avoidance

- Si hay perdidas, el TCP vuelve a empezar con la menor tasa de envío posible (1 MSS) y va incrementando la ventana de doble por cada RTT
- Hasta pero la mitad de la ventana que había alcanzado cuando se ha producido una perdida siendo ssthresh = max(2 MSS, wnd / 2)
- A partir de este punto, TCP sigue incrementando la ventana pero más suavemente, es concreto de 1 MSS por cada RTT
- Esto porque la cantidad de información presente en una red cambia constantemente y por lo tanto también su punto de congestión
- Es muy probable por lo tanto que la próxima perdida (si la hay) será con un valor de ventana diferente

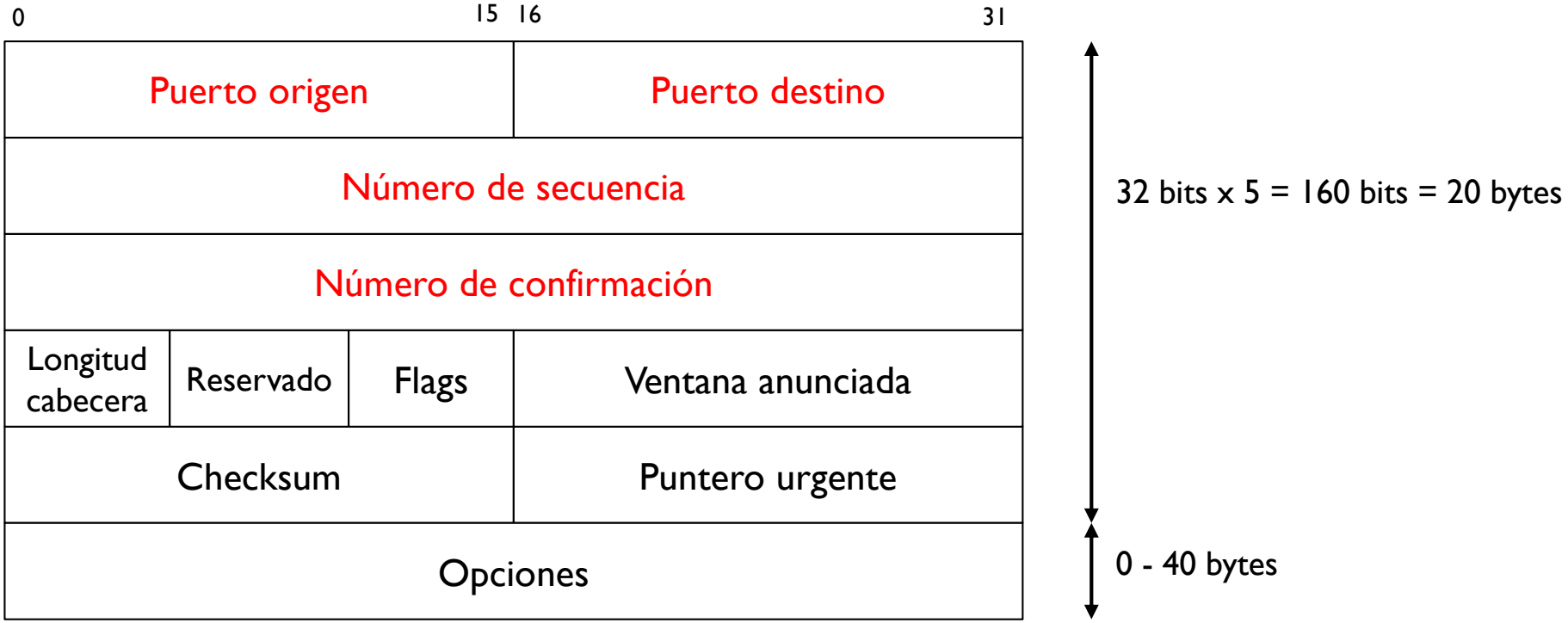
Tema 3 – Retransmission Time Out (RTO)

- El RTO se inicializa cuando se envía el primer segmento
  - Cada vez que se recibe un ack nuevo y hay segmentos pendientes de confirmación, el RTO está activo y va disminuyendo con el tiempo
  - Cuando llega a 0, se retransmite el primer segmento pendiente de confirmación (el primero del buffer TX)
  - Si no hay segmentos pendientes de confirmación, el RTO está desactivo
- RTO se computa con una formula
  - RTO = srtt + 4 rttvar
  - Donde srtt es la media de los RTT y rttvar es la varianza
- Cada vez que se pierde un segmento y salta el RTO, su valor se duplica

Tema 3 – Retransmission Time Out (RTO)

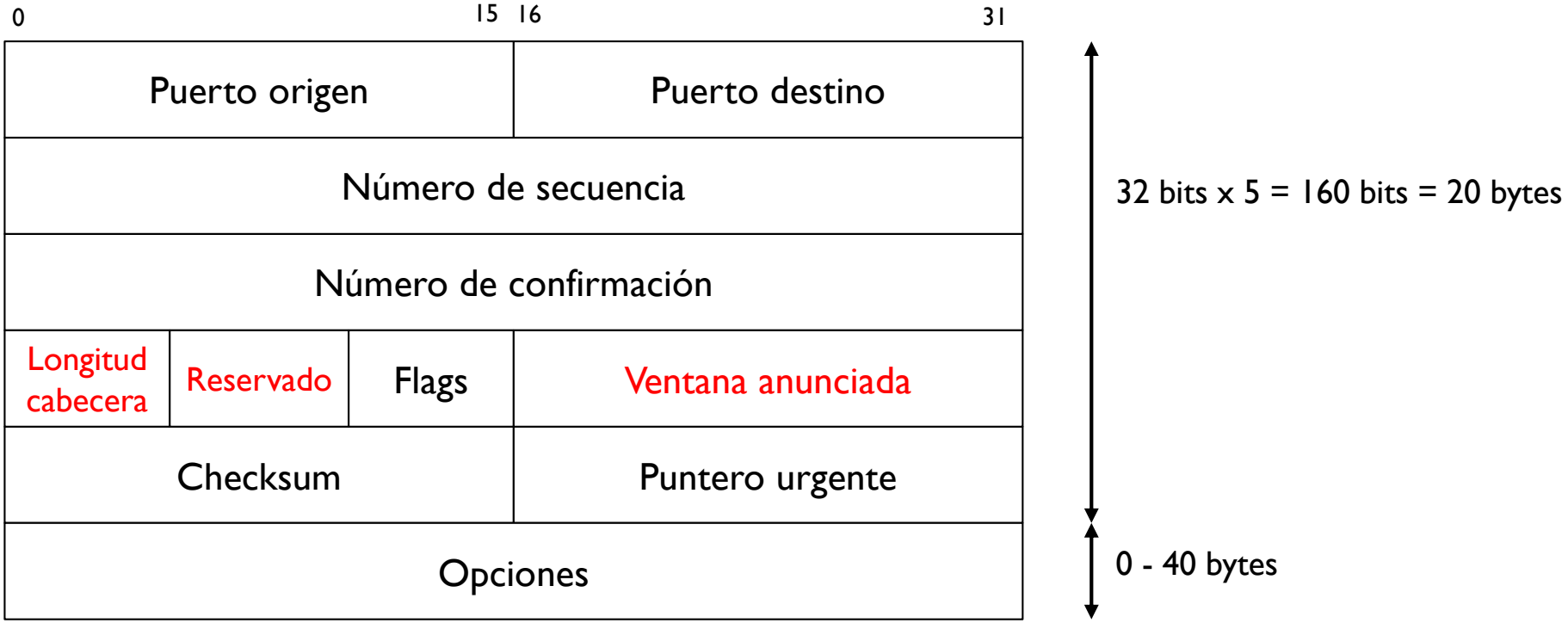


Tema 3 – Cabecera TCP



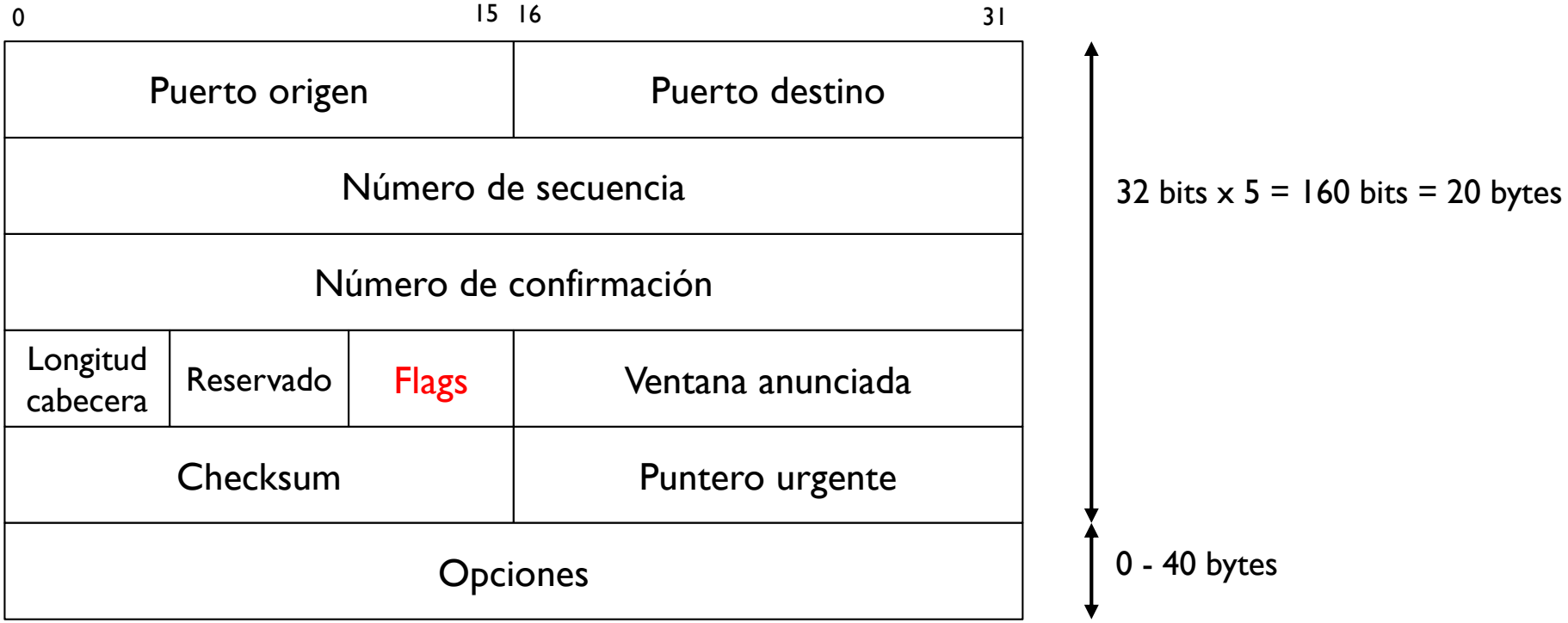
- Puerto origen: identifica la aplicación origen de los datos
- Puerto destino: identifica la aplicación destino de los datos
- Número de secuencia: indica la posición del primer byte de dato enviado respecto al total
- Número ack: confirmación de haber recibido todo correctamente hasta este número menos 1

Tema 3 – Cabecera TCP



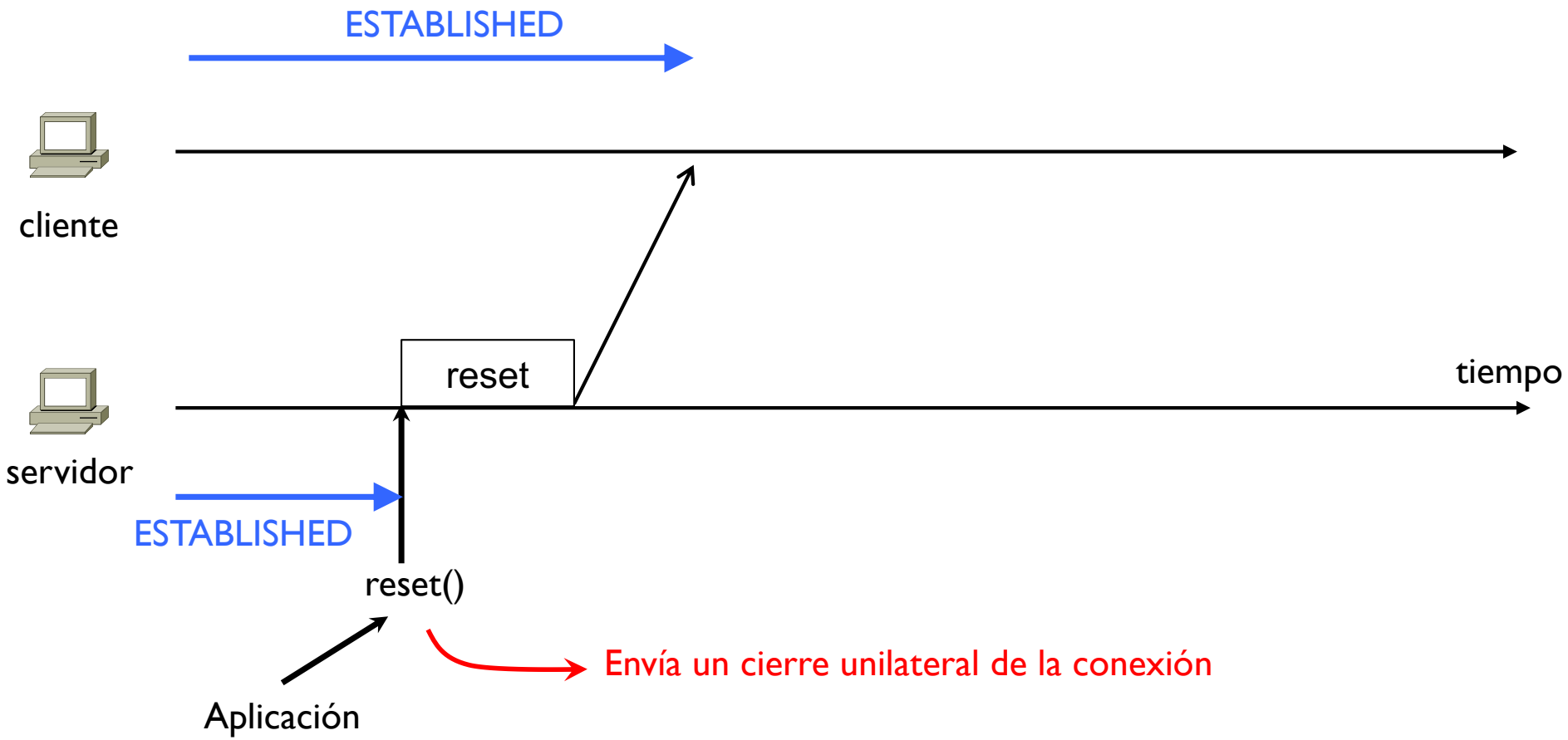
- Longitud cabecera: este campo indica la longitud de la cabecera ya que esta es variable. Por defecto la cabecera TCP es de 20 bytes, pero hay un campo opciones que puede ocupar de 0 a 40 bytes.
- Reservado: campo reservado para futuras mejoras del TCP
- Ventana anunciada: campo donde se pone el valor de la ventana awnd que indica el espacio libre en el buffer de RX

Tema 3 – Cabecera TCP

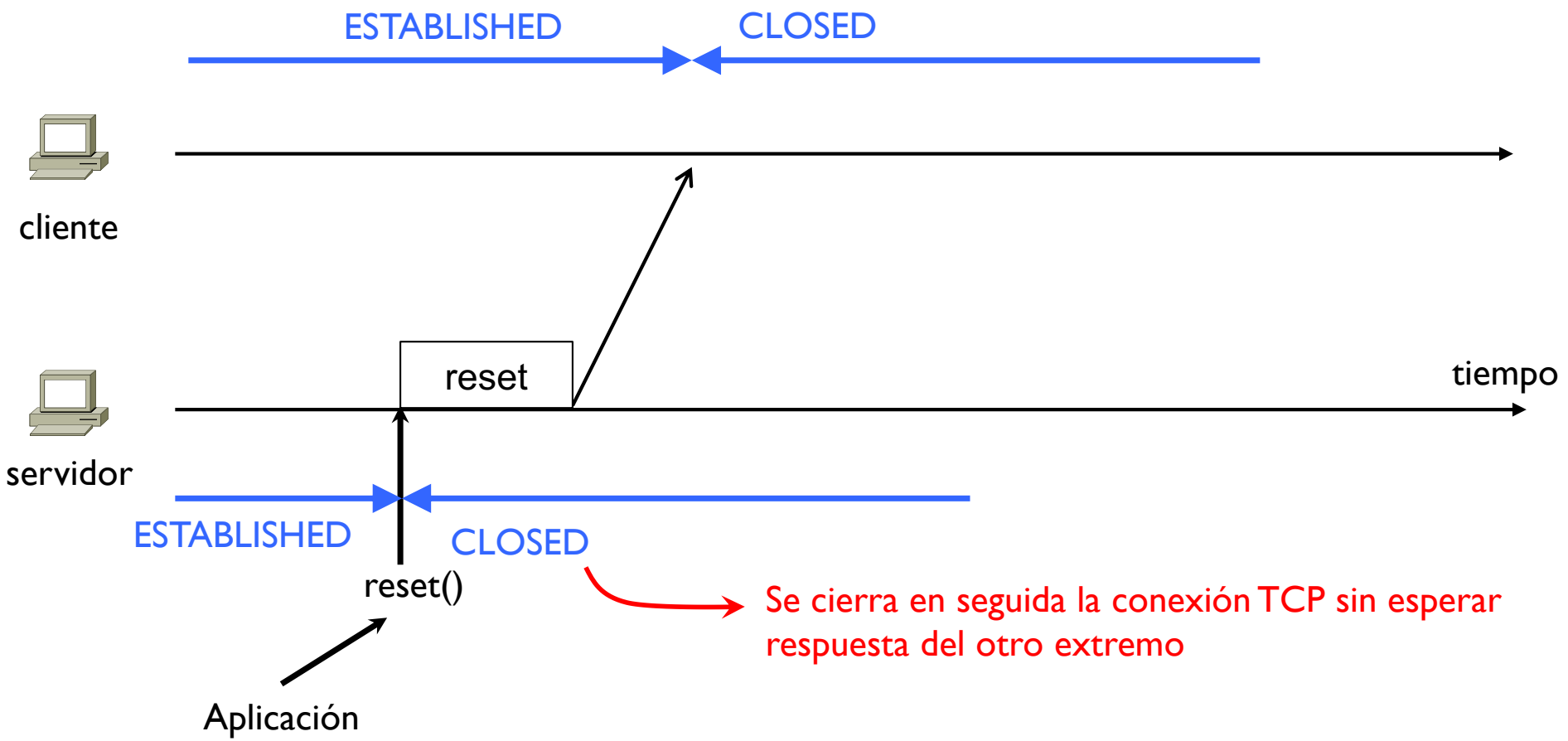


- Flags: indica que tipo de segmento se está transmitiendo. Los más usados son:
  - S: sincronismo, si activo quiere decir que se está enviando un syn (3WH)
  - A: ack, si activo significa que es una confirmación
  - F: finalizar, si activo significa que se quiere terminar la conexión TCP
  - R: reset, si activo significa que se quiere abortar una conexión TCP unilateralmente sin esperar un ack de vuelta (por ejemplo se ha intentado hacer una operación no permitida)

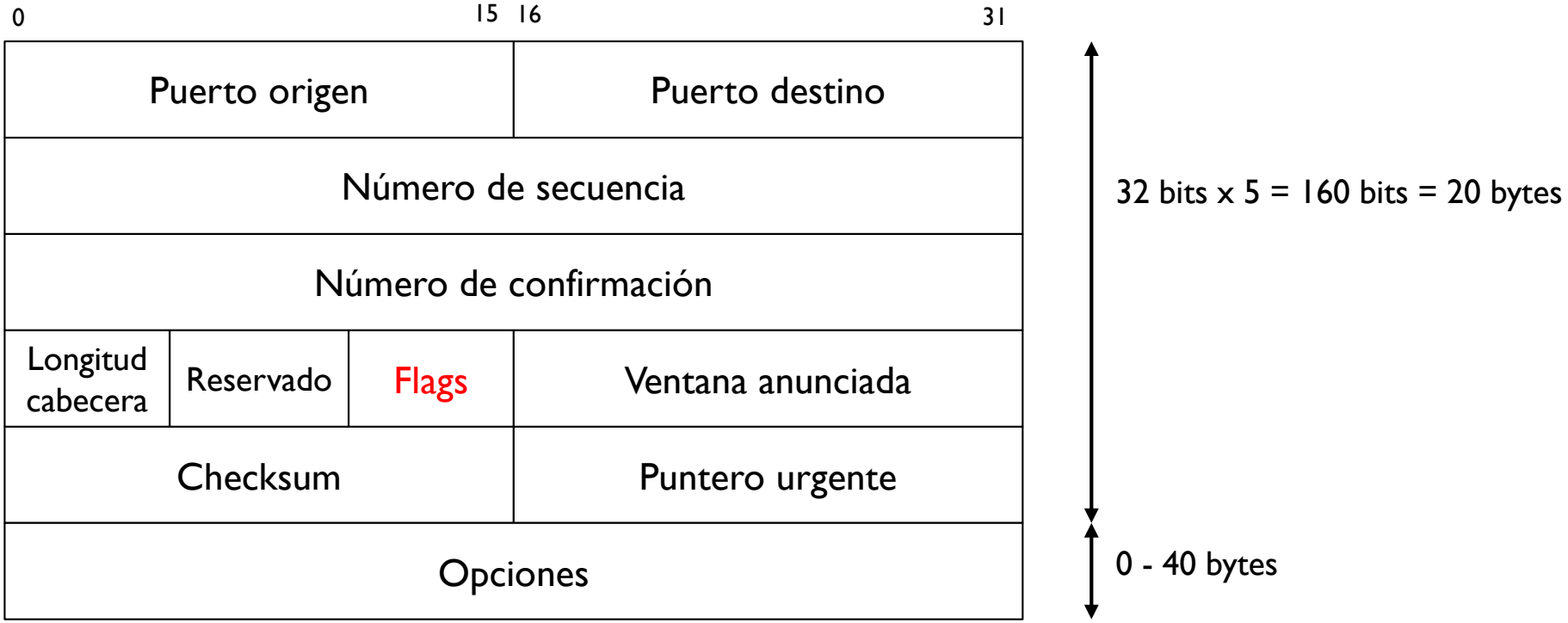
Tema 3 – Terminación de una conexión



Tema 3 – Terminación de una conexión

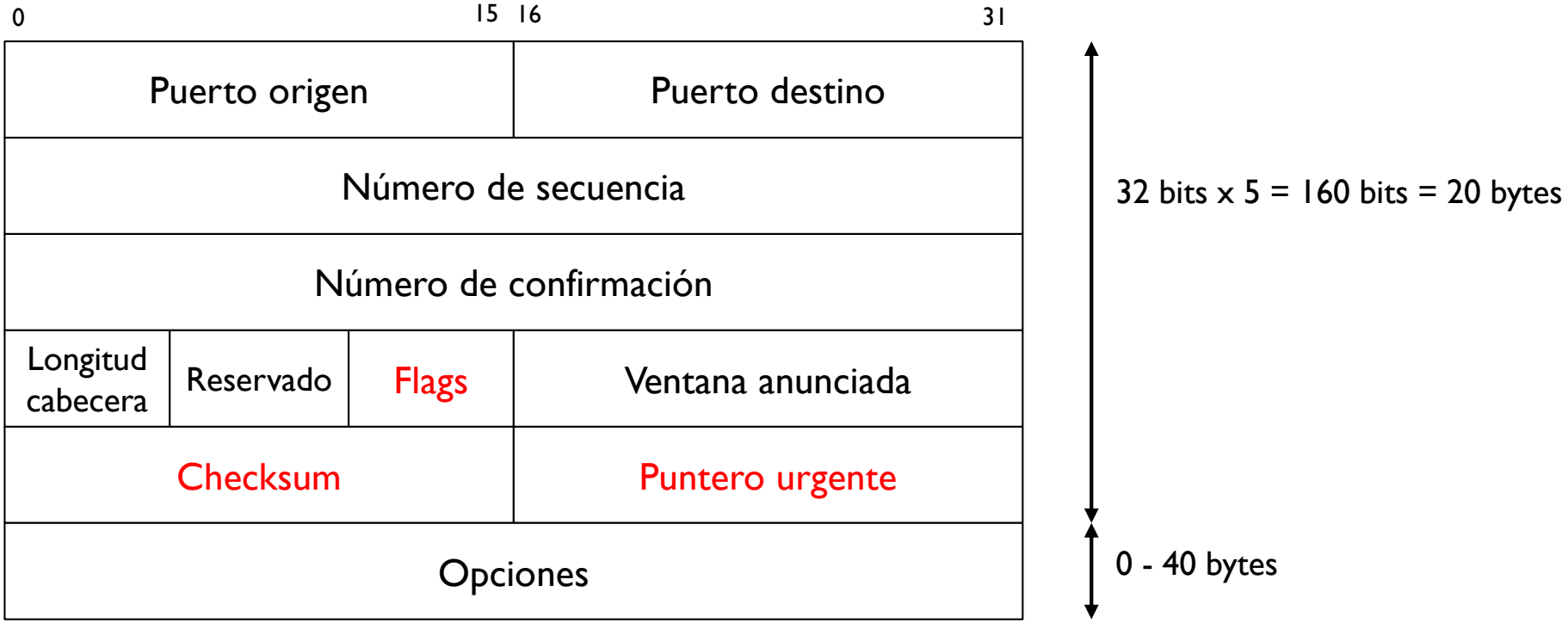


Tema 3 – Cabecera TCP



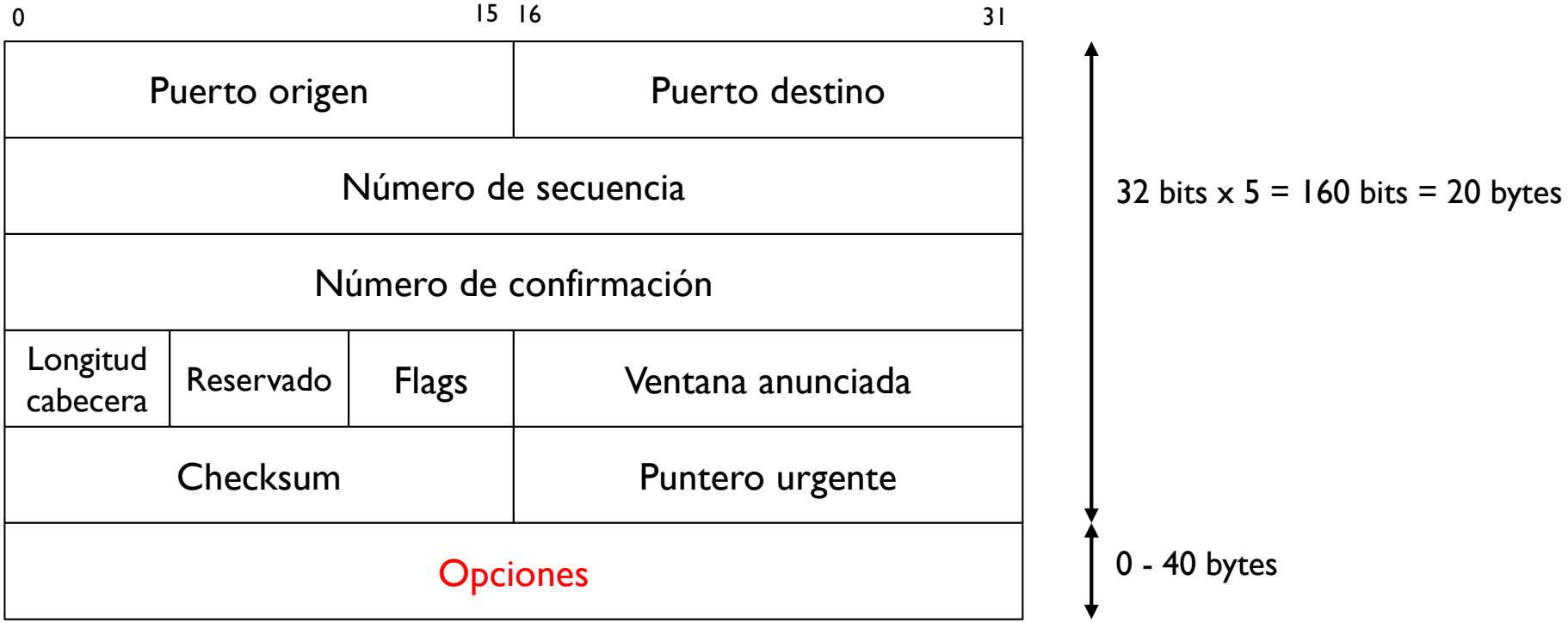
- Flags: indica que tipo de segmento se está transmitiendo. Los más usados son:
  - P: push, un extremo activa este flag cuando quiere que el otro lea rápido este mensaje

Tema 3 – Cabecera TCP



- Puntero urgente: se usa juntamente al flag U (urgente). Si el flag está activo, quiere decir que se quiere transmitir algo urgente y este puntero indica que parte de los datos son urgente (desde el número de secuencia hasta este puntero)
- Checksum: control de error en la lectura de los bits que componen el segmento TCP

Tema 3 – Cabecera TCP



- Opciones:
  - Se pueden usar este campo para añadir información y usar para implementar algún nuevo algoritmo

Tema 3 – Opciones

- Timestamp
  - Se puede enviar el reloj de un extremo y el otro extremo reenvía este mismo valor de vuelta al primero
  - Restando el reloj actual con el reloj contenido en este campo, se calcula el RTT y por lo tanto el RTO
- MSS
  - Durante el 3WH, los dos extremos se envían los respectivos MSS en este campo opciones para saber el máximo tamaño posible de los datos

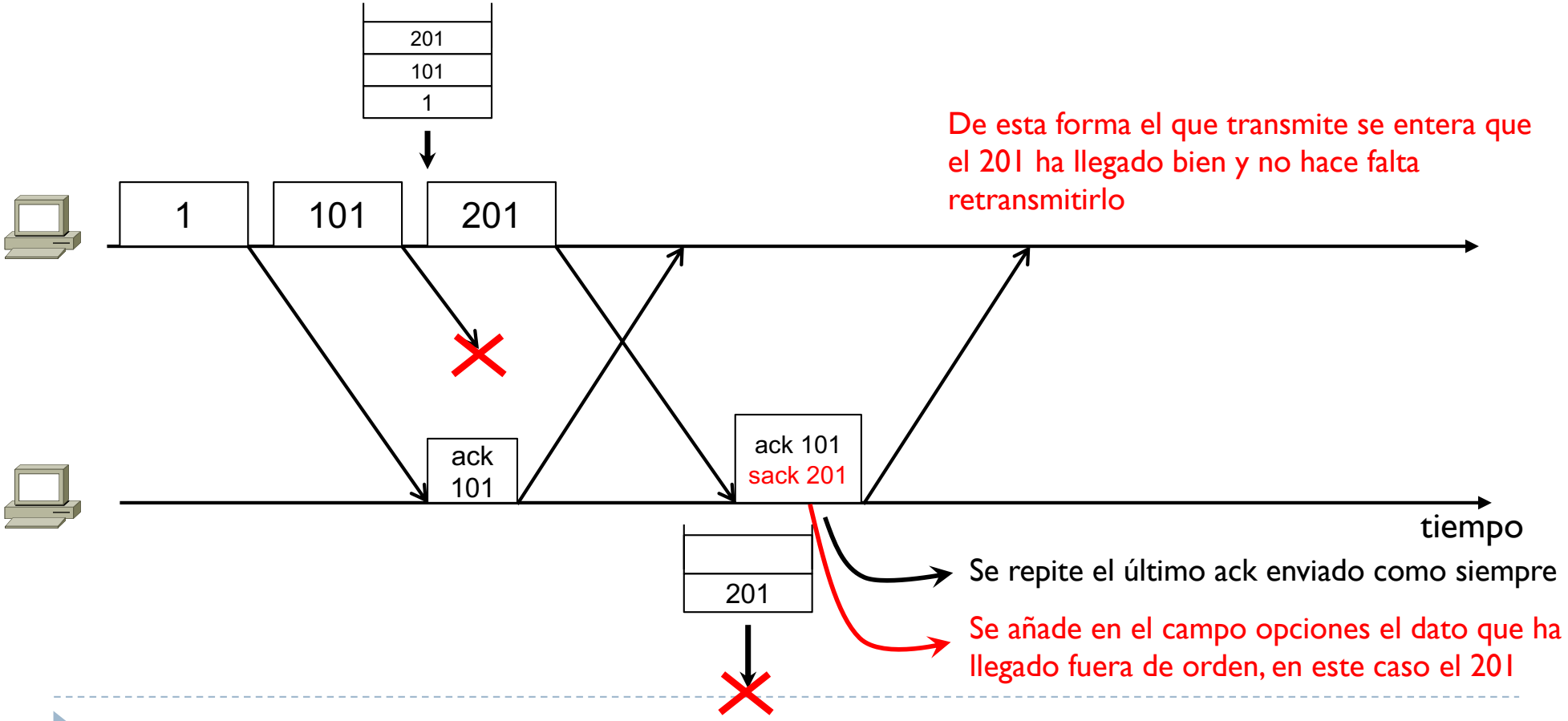
Tema 3 – Opciones

- Window Scale Factor (WSF)
  - Se envía en el 3WH. Esta valor sirve para anunciar ventanas awnd más grande de la representación máxima posible en la cabecera TCP
  - En la cabecera TCP hay 16 bits disponibles, quiere decir que el número máximo representable es 2^{16}-1 = 65535 bytes
  - Cuando se estandarizó TCP, este valor era considerado grande (recordar que este valor indica el espacio disponible en el buffer de RX)
  - Hoy en día es una gran limitación
  - El WSF permite alcanzar valores más grandes y en concreto el valor real de la ventana anunciada es el enviado en el campo de la cabecera TCP multiplicado por 2^{WSF}
    - Si por ejemplo el valor enviado es 10.000 y el WSF establecido en el 3WH es 3, la ventana anunciada awnd es

$$10.000 \times 2^3 = 80.000 \text{ bytes}$$

Tema 3 – Opciones

- Selective ACK (SACK)
  - Se usa para la versión TCP SACK
  - En concreto, permite usar el campo opciones para enviar los datos recibidos fuera de orden en caso de error.



Tema 3 – Protocolos UDP y TCP

- a) Introducción
- b) El protocolo UDP
- c) El protocolo TCP
  - Arquitectura
  - El MSS
  - Números de secuencia
  - Establecimiento y terminación de una conexión TCP
  - Funcionamiento durante la transmisión
    - Control de flujo
    - Control de congestión
  - Cabecera TCP

Xarxes de Computadors

Tema 3 – Protocolos UDP y TCP