Redes de computadoras

Capa de transporte – Servicio de Transferencia Fiable

Las diapositivas están basadas en en libro:

“Redes de Computadoras – Un enfoque descendente”
de James F. Kurose & Keith W. Ross

Transferencia de datos fiable

Implementar servicios de transferencia de datos fiables es necesario en varias capas

- Aplicación
- Transporte
- Enlace

Transferencia de datos fiable

Un canal fiable por el cual se puede transferir datos de modo que ninguno de los bits de datos esta corrompido

- Ningún bit pasa de 0 a 1 o de 1 a 0
- No se pierden bits
- Los bits llegan en el mismo orden en que se enviaron

Tarea compleja, la capa inferior puede se no fiable

(Ejemplo TCP corriendo sobre IP)

Transferencia de datos fiable

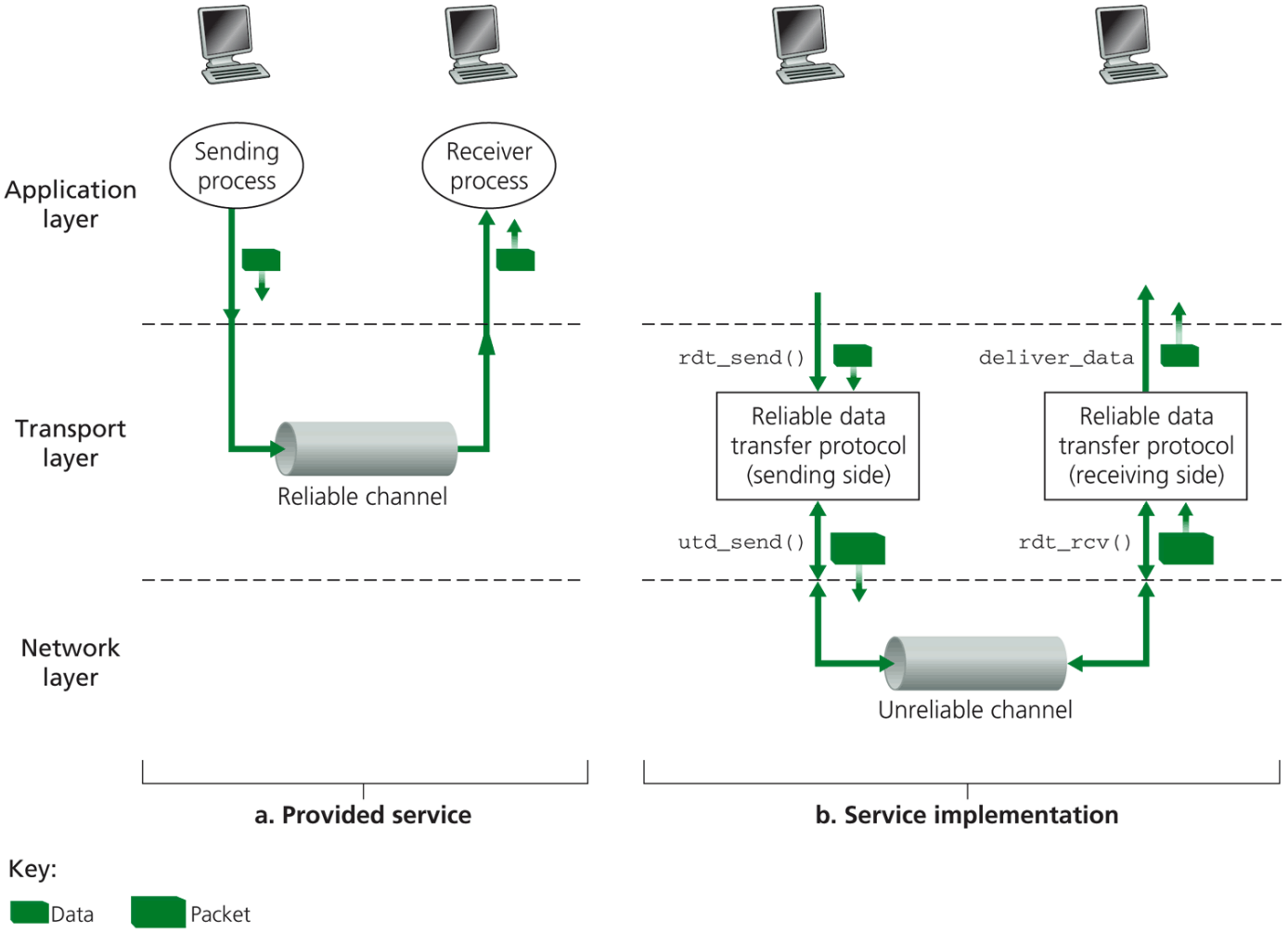


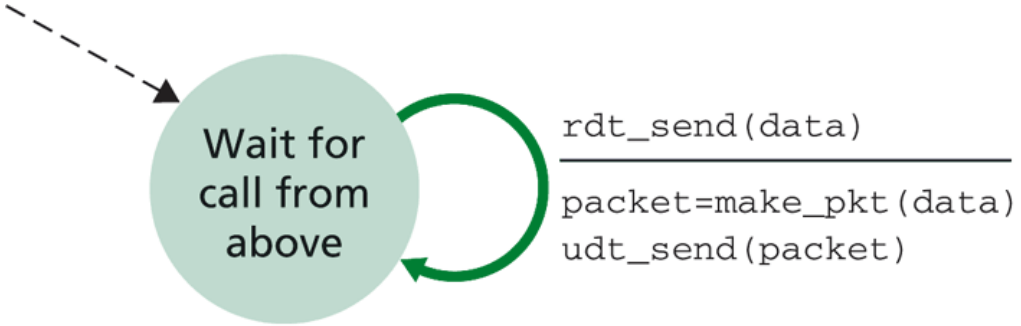
Figure 3.8 ♦ Reliable data transfer: Service model and service implementation

Protocolo RDT (reliable data transfer)

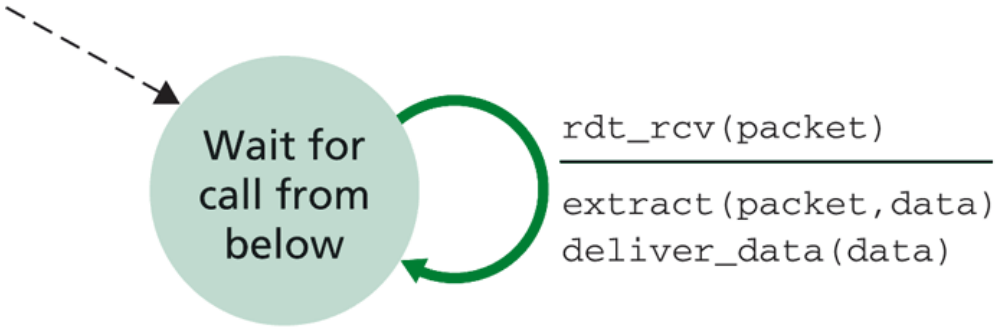
Transferencia de datos fiable sobre un canal totalmente fiable

- El lado emisor recibe datos de la capa superior
  - rdt_enviarDatos(datos)
- Crea un paquete que contiene los datos y envía el paquete
  - crear_paquete(datos)
- El receptor recibe el paquete mediante
  - rdt_recibir(paquete)
- Extrae los datos y los pasa a la capa superior.

RDT sobre canal fiable



a. rdt1.0: sending side



b. rdt1.0: receiving side

RDT sobre canal con errores de bit

Suposición:

Todos los paquetes llegan en el orden en que se enviaron pero sus bits pueden estar corrompidos.

El receptor debe comunicar si lo recibido es aceptado

- Reconocimientos afirmativos

  “De acuerdo”

- Reconocimientos negativos

  “Por favor repita”

Mensajes de control que permiten al emisor saber si se ha recibido el mensaje correctamente o se debe repetir.

Protocolos ARQ

Automatic Repeat reQuest

(Solicitud automática de repetición)

- Detección de errores

  Se requiere un mecanismo para que el receptor detecte errores

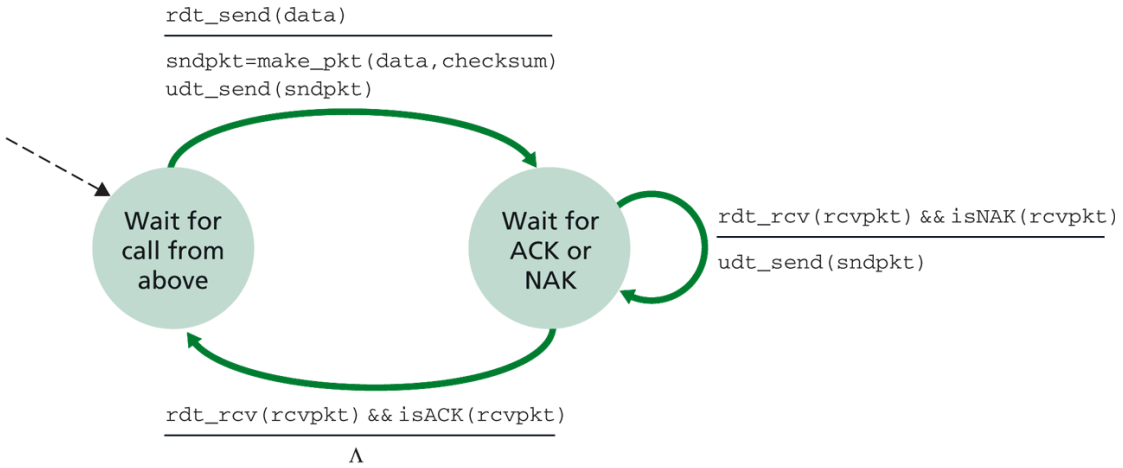
- Realimentación del receptor

  El receptor debe informar al emisor
  si recibió satisfactoriamente (ACK) o no (NAK)

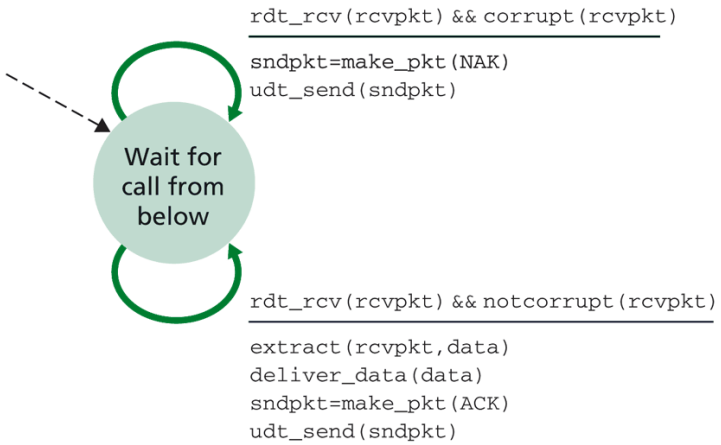
- Retransmisión

  Un paquete recibido con errores será retransmitido

RDT sobre canal con errores de bit



a. rdt2.0: sending side



b. rdt2.0: receiving side

RDT sobre canal con errores de bit

Mensajes ACK o NAK corruptos

- Otro tipo de paquete
- Añadir bits para no sólo detectar sino también recuperarse de un error
- Envíar nuevamente el paquete al recibir un ACK o NAK corrupto. (posibles paquetes duplicados)

Ejemplo sobre un protocolo de parada y espera simple

(stop-and-wait protocol)

RDT sobre canal con errores de bit Emisor

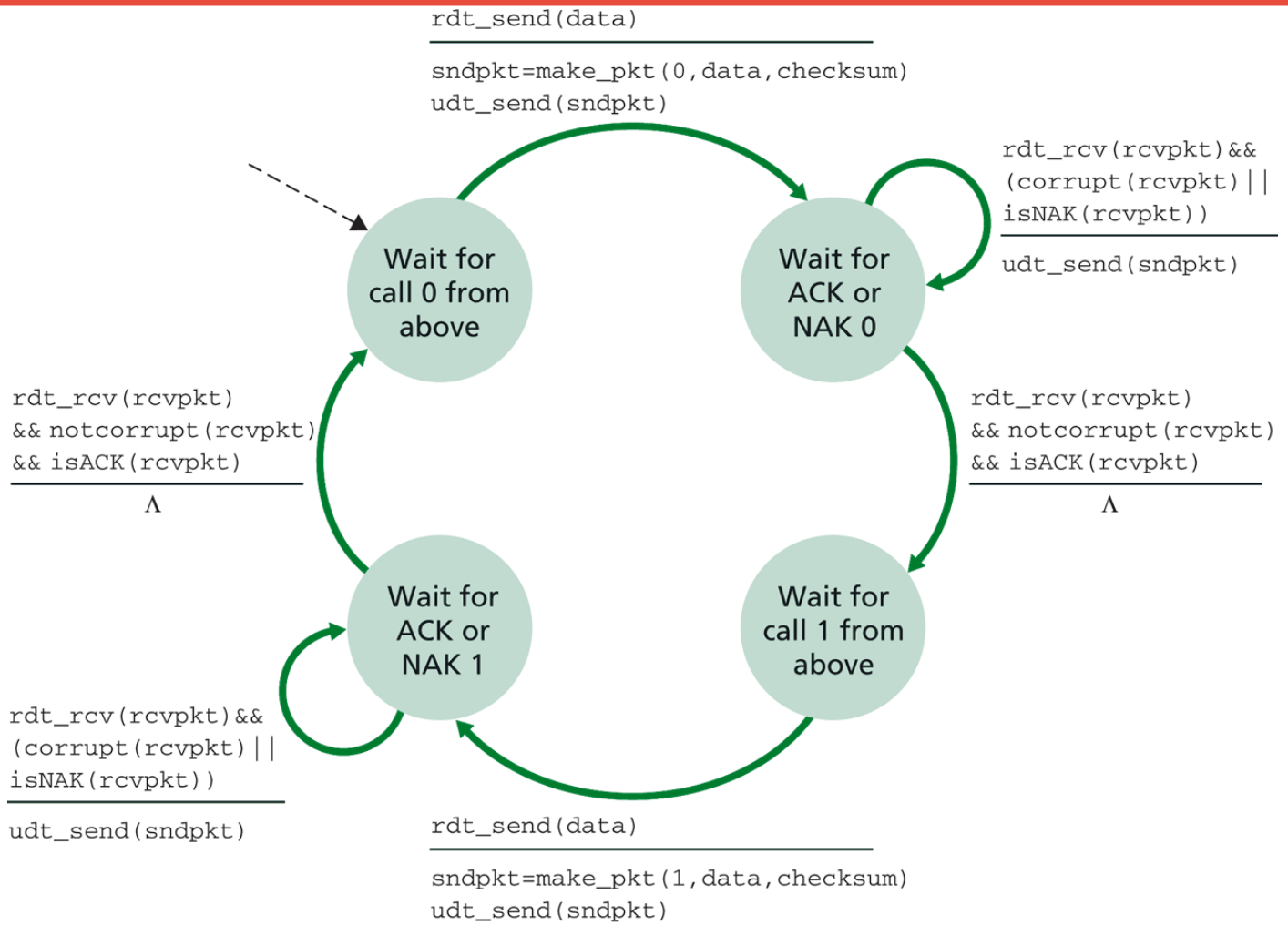


Figure 3.11 ♦ rdt2.1 sender

RDT sobre canal con errores de bit Receptor

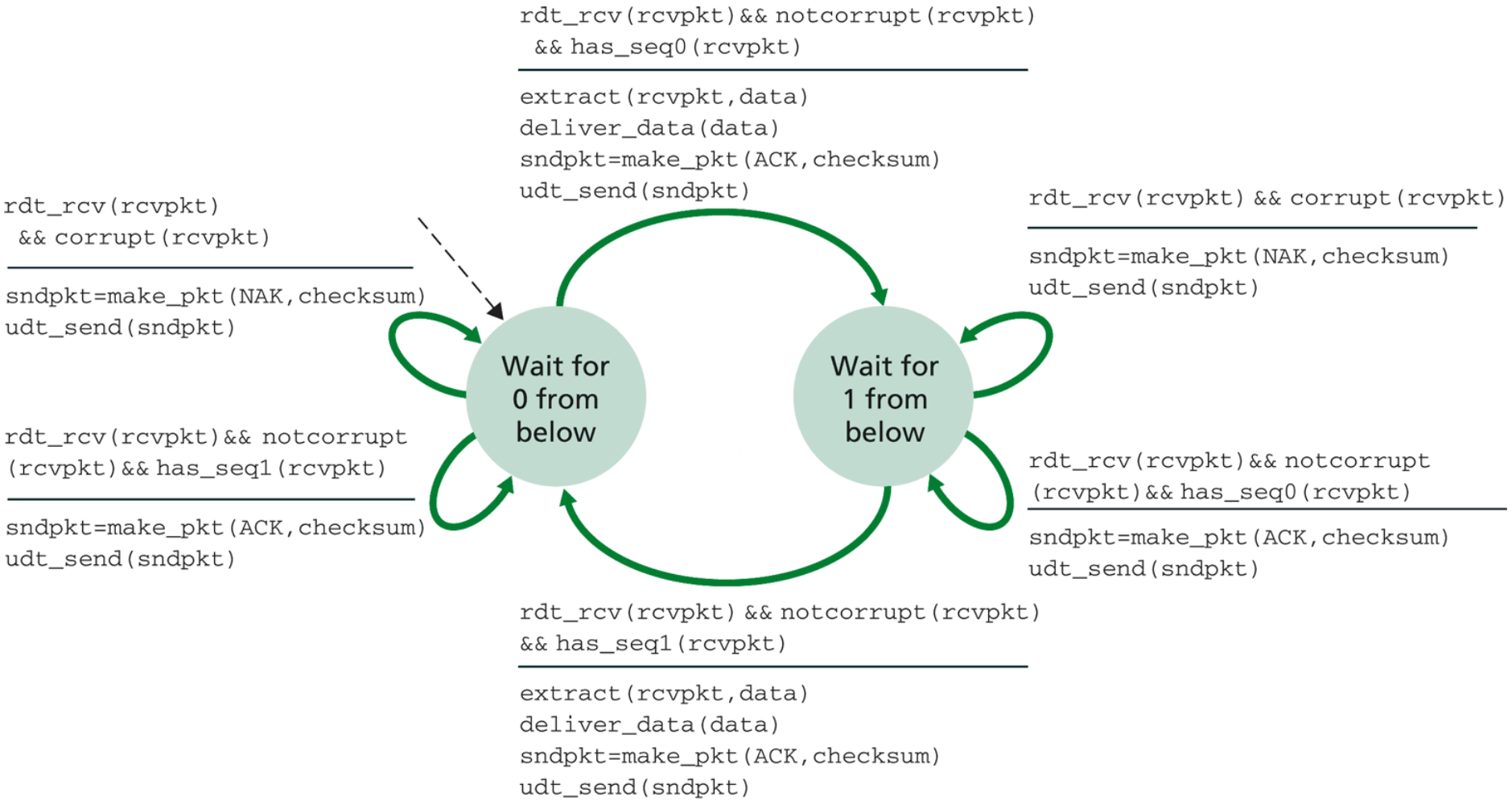


Figure 3.12 ♦ rdt2.1 receiver

RDT sobre canal con errores de bit y perdidas de paquetes

¿Cómo detectar si se pierde un paquete?

- Tiempo de espera de al menos el tiempo de retardo de ida y vuelta.

- Tiempo adicional en buffers y de procesamiento

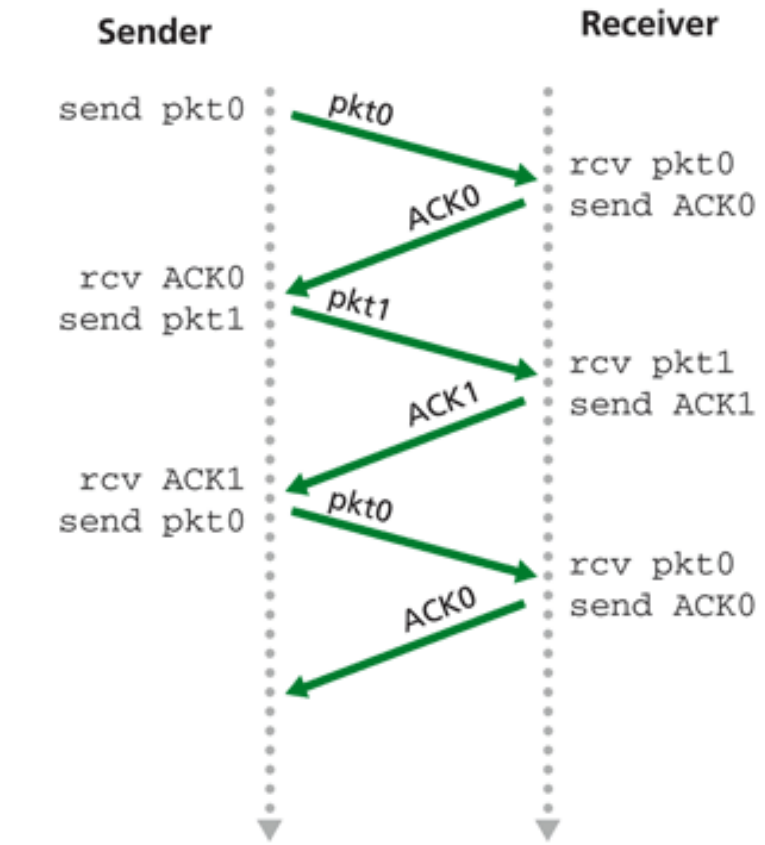
Difícil de estimar y casi imposible de saber con precisión

Se utiliza un tiempo en que es probable que el paquete se haya perdido, aunque no se tenga la certeza.

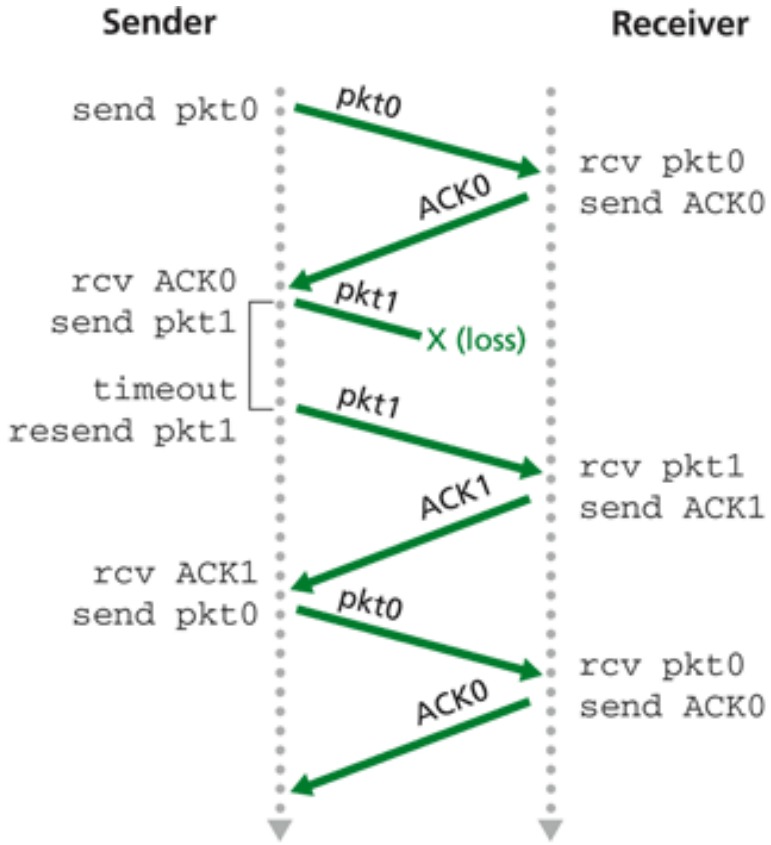
Posibles paquetes duplicados

Uso de número de secuencia

RDT sobre canal con errores de bit y perdidas de paquetes

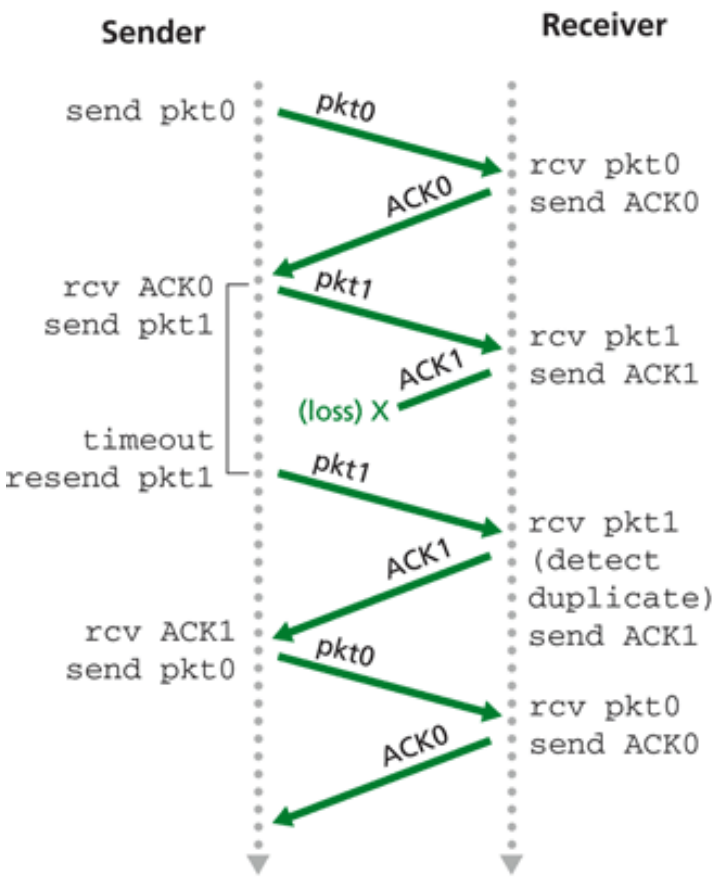


a. Operation with no loss

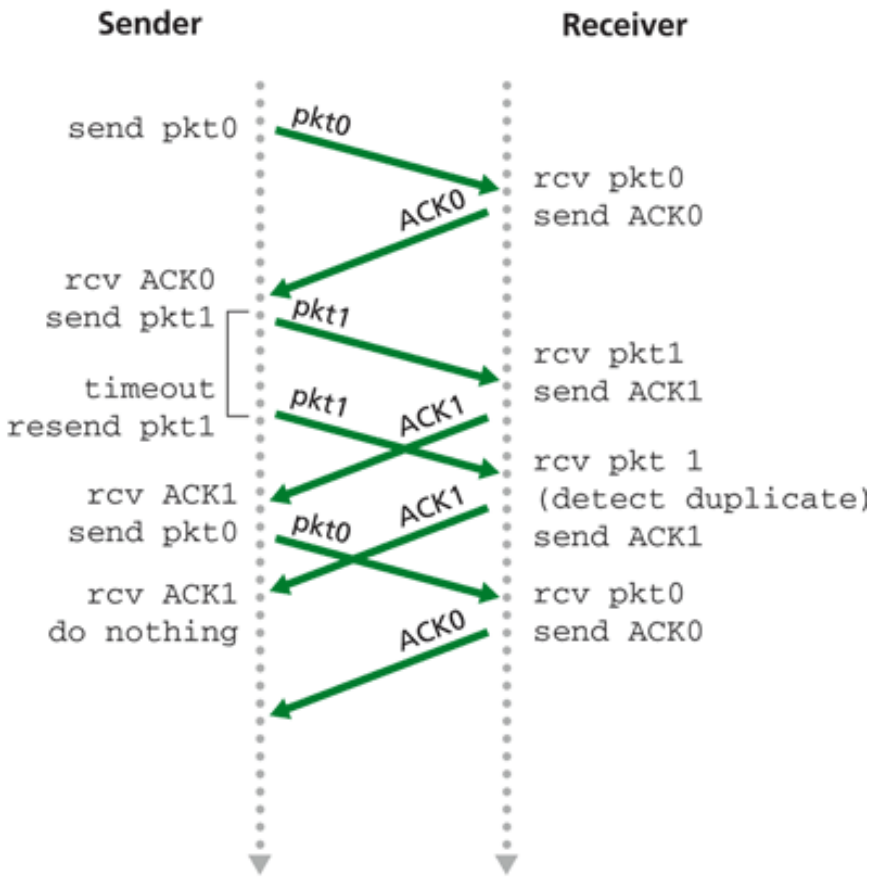


b. Lost packet

RDT sobre canal con errores de bit y perdidas de paquetes

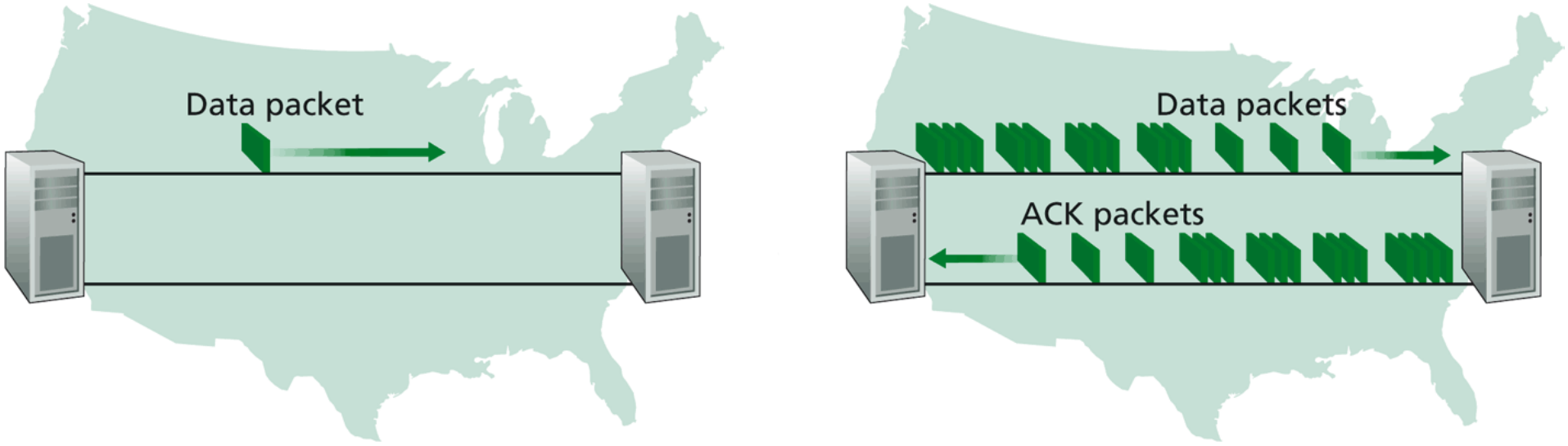


c. Lost ACK



d. Premature timeout

Protocolo de parada y espera vs procesamiento en cadena



a. A stop-and-wait protocol in operation

b. A pipelined protocol in operation

Figure 3.17 ♦ Stop-and-wait versus pipelined protocol

Protocolo de transferencia de datos fiable

El protocolo “stop-and-wait” puede ser sencillo de implementar y correcto en su proceder pero muy ineficiente.

A modo de ejemplo se consideran 2 hosts.

- Uno a cada lado de EEUU con un RTT entre ellos de 30ms.
- Los conecta un enlace con velocidad de transmisión R de 1Gbps (10^9 bits/s)
- Con un tamaño de paquete L de 1.000 bytes (8.000bits) por paquete.

$d_{trans} = L/R = 8000/10^9 = 8\mu s$ (microsegundos)

El emisor ha estado transmitiendo sólo durante 8 microsegundos, mientras que se demoró 30,008 ms en enviar el paquete al receptor.

Una tasa de transferencia de 267 kbps

Protocolo de transferencia de datos fiable

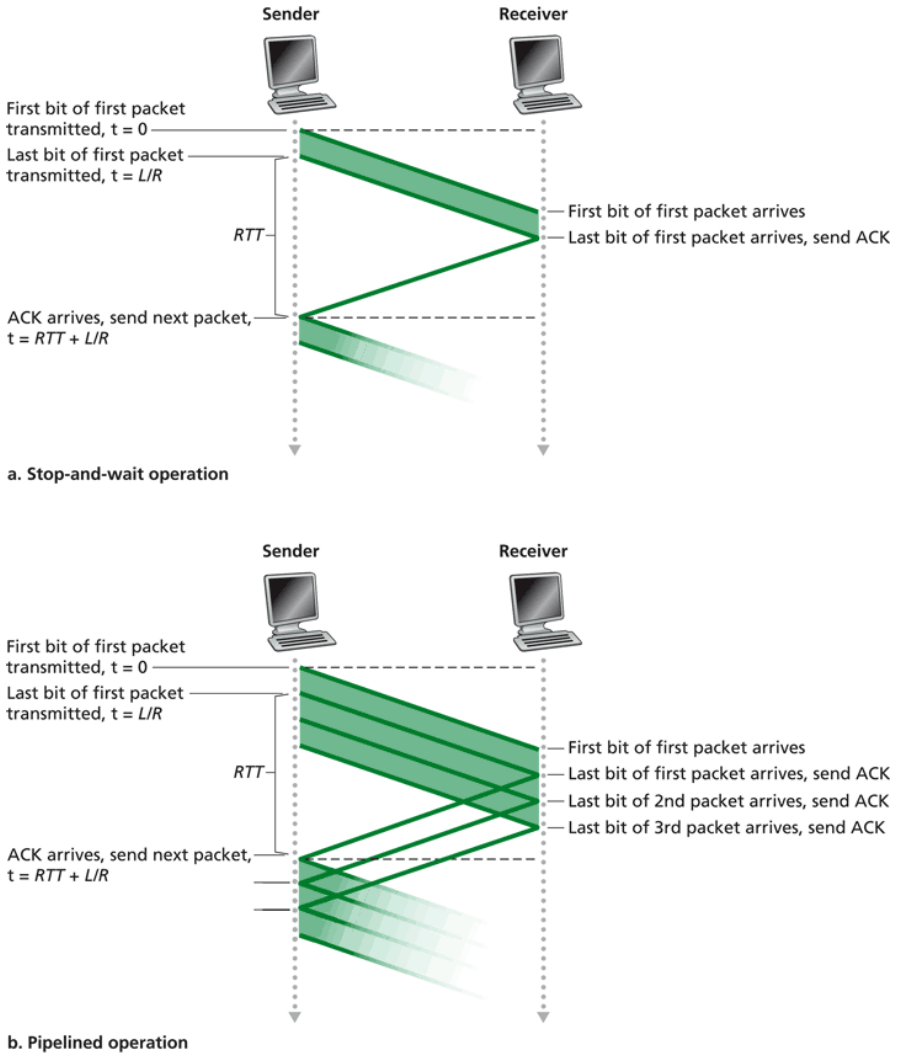


Figure 3.18 ♦ Stop-and-wait and pipelined sending

Protocolo de transferencia de datos fiable

Los protocolos de red pueden limitar las capacidades proporcionados por la infraestructura subyacente.

Posible solución:

Enviar varios paquetes sin esperar a los mensajes de

Reconocimiento.

Pipelining o procesamiento en cadena

Procesamiento en cadena

- El rango de los números de secuencia debe ser mayor

  Debe ser un número único y pueden haber muchos paquetes en tráfico sin confirmación

- Se debe contar con la posibilidad de almacenar en buffer más de un paquete.

  El emisor debe conservar los paquetes enviados y no confirmados
  El receptor puede necesitar almacenar paquetes recibidos correctamente.

- Contar con estrategias para la recuperación de pérdida de paquetes, corrupción o retardos

Retroceder N (GBN - Go-back N)

Un protocolo GBN utiliza “una ventana” deslizante la cual representa los números de secuencia disponibles a estar en tránsito

- base (número de secuencia del paquete más antiguo no reconocido)
- nextseqnum (número de secuencia más pequeño no utilizado)

  Se pueden utilizar números [signumsec, base+N-1]

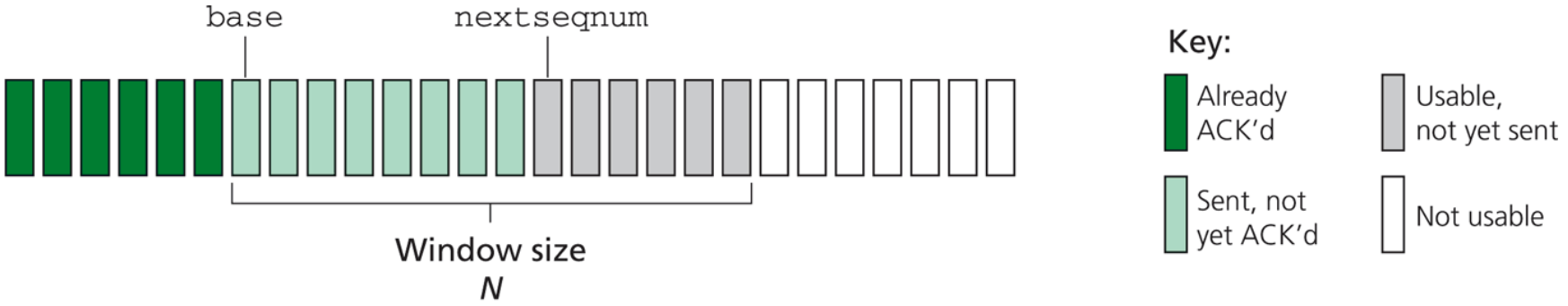


Figure 3.19 ♦ Sender's view of sequence numbers in Go-Back-N

Retroceder N (GBN - Go-back N) Emisor

Sucesos en el emisor:

Invocación desde la capa superior

Se debe saber si hay paquetes disponibles
(menos de N paquetes no reconocidos en circulación)

Se debe tener algo previsto para el caso en que la ventana esté llena.
Rechazo, buffer, semáforo, etc.

Recepción de ACK

Reconocimiento acumulativo, ACK x significa que también los paquetes menores a x han sido aceptados.

Fin de temporizador

En caso de completarse el tiempo limite del temporizador, se envían todos los paquetes transmitidos sin reconocimiento.

Retroceder N (GBN - Go-back N)

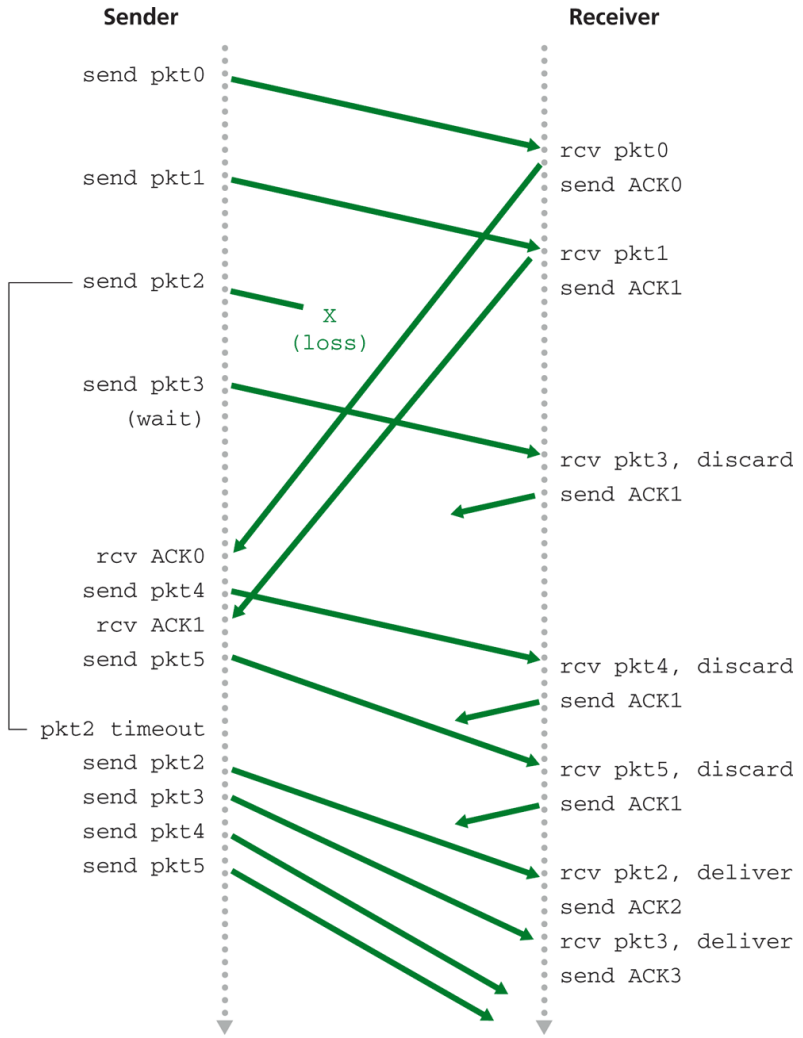


Figure 3.22 ♦ Go-Back-N in operation

Repetición Selectiva (SR)

GBN con tamaño de ventana, ancho de banda y retardo grande genera muchos paquetes en el canal.

Un paquete erroneo podría hacer retransmitir muchos datos de forma innecesaria.

El protocolo de repetición selectiva sólo retransmitirá aquellos paquetes que sospeche perdidos.

El receptor confirmará la recepción de un mensaje, tanto en orden correcto como no.

Los paquetes no recibidos en orden se almacenan en buffer

Repetición Selectiva (SR)

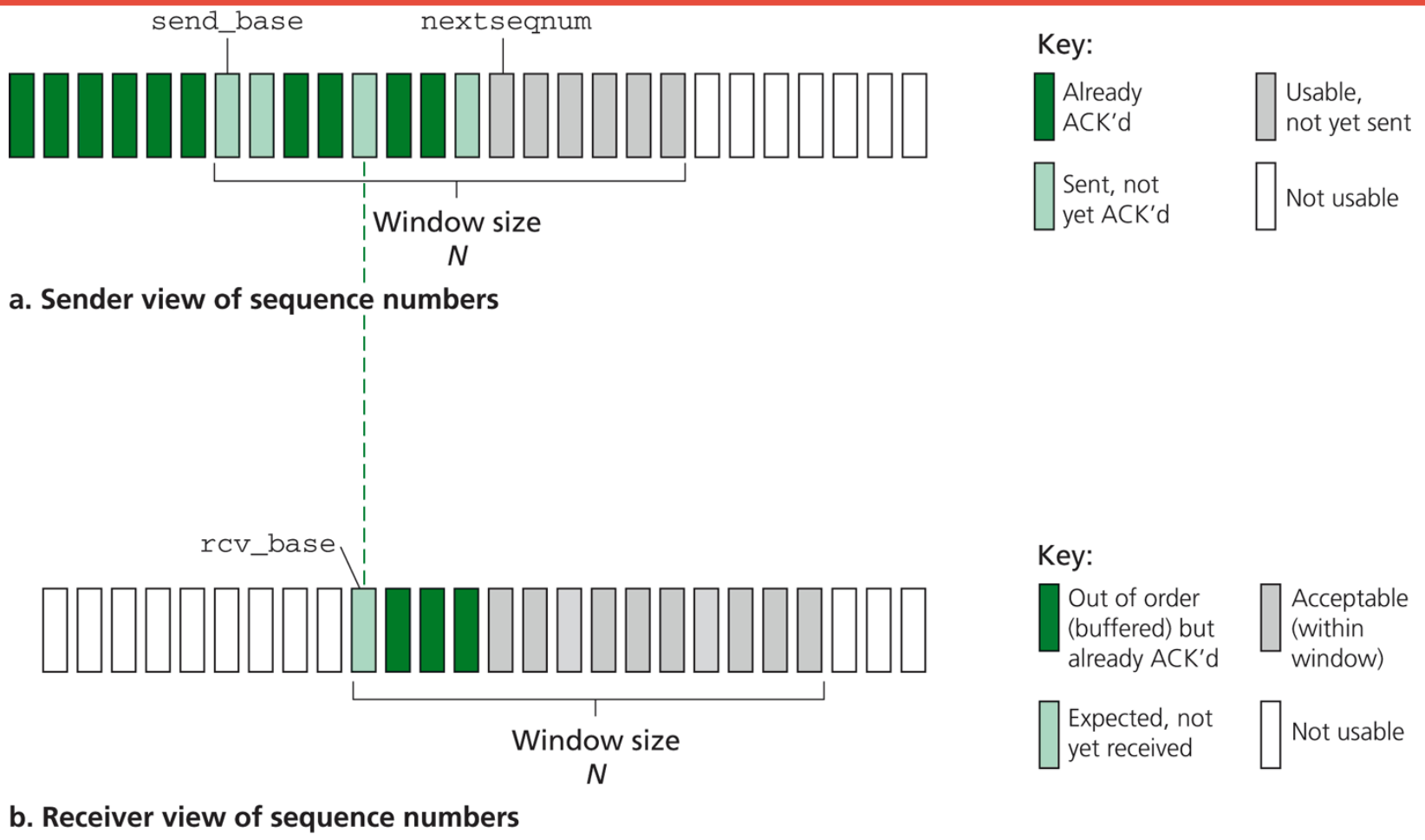
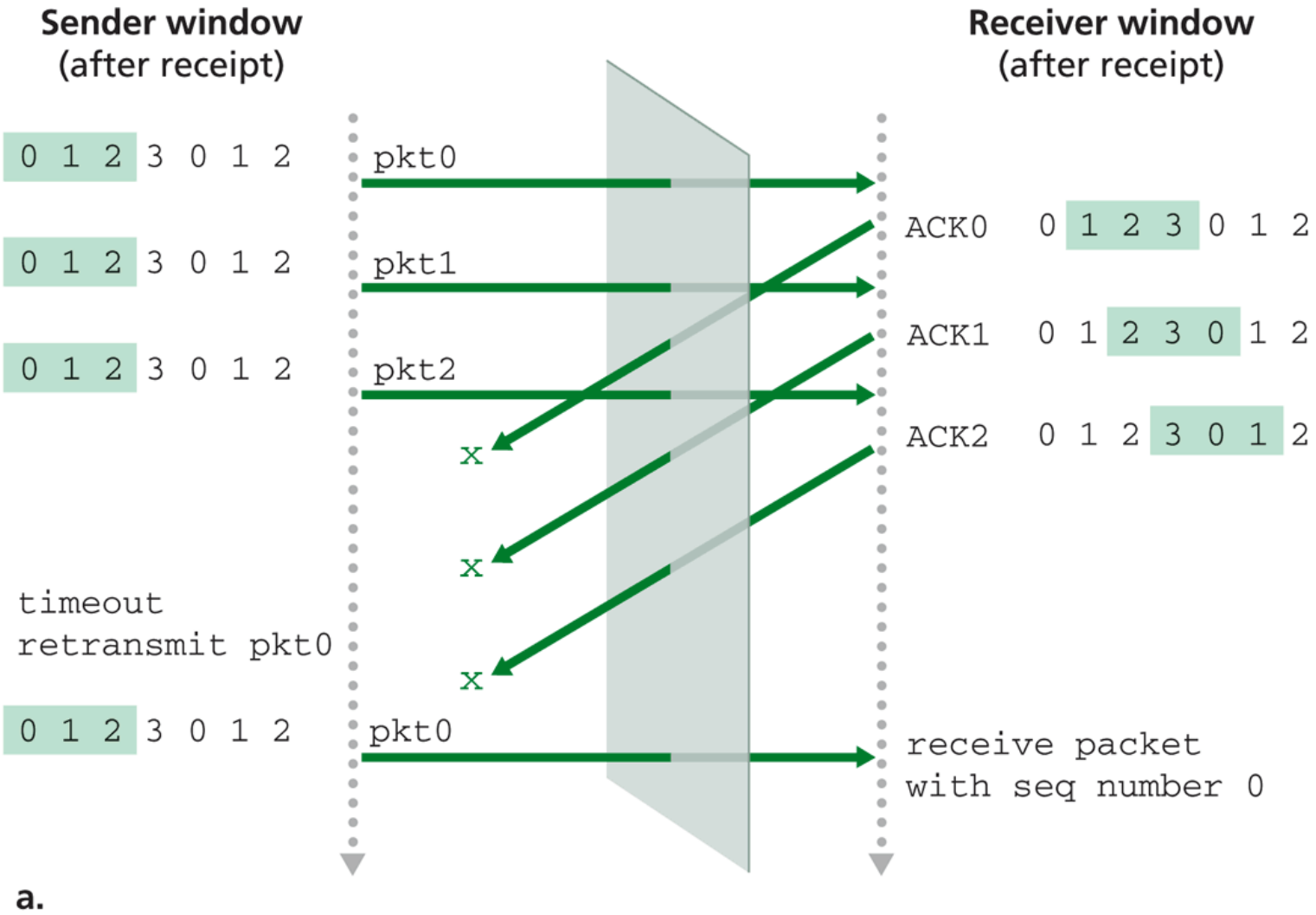
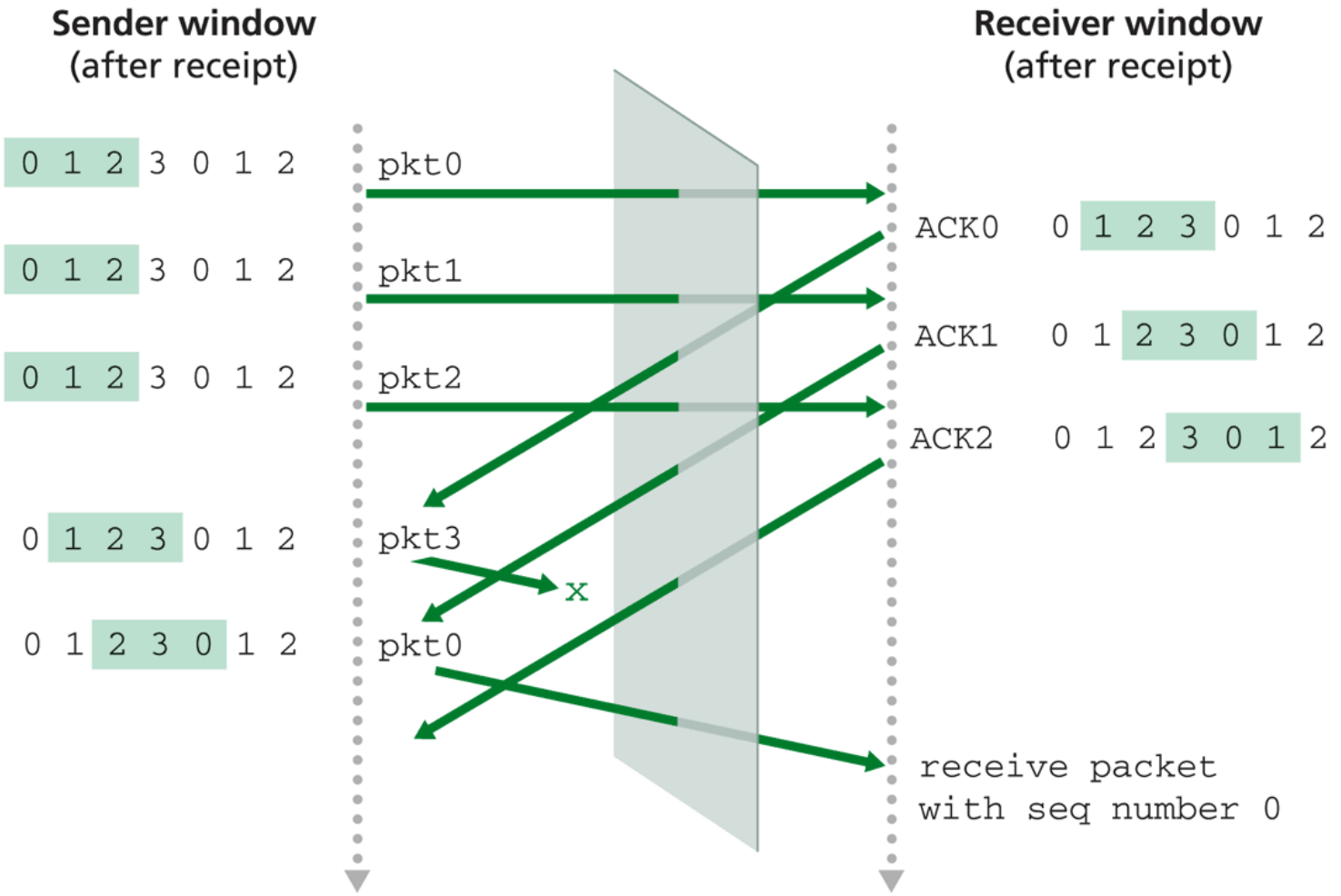


Figure 3.23 ♦ Selective-repeat (SR) sender and receiver views of sequence-number space

Repetición Selectiva (SR) tamaño de ventana



Repetición Selectiva (SR) tamaño de ventana



b.