Capa de Enlace de Datos

Redes de Computadores

FIEC04705

Sesión 06

Agenda

- Terminología
- Detección vs. Correción
- Corrección de errores
- Control de flujo

Terminología

Terminología

- **Redundancia**: El concepto central para detectar o corregir errores es la redundancia. Esto es, enviar bits adicionales con los datos. Estos bits redundantes son añadidos por el transmisor y removidos por el receptor. Su presencia permite al receptor detectar o corregir los bits corruptos.

Detección vs. corrección de errores

Detección vs. Corrrección de errores

- En **detección de errores** se busca únicamente si un error ha ocurrido
- En **corrección de errores** necesitamos saber el número exacto de bits que han sido corrompidos y lo que es más, su posición en el mensaje.

Corrección de errores

Hamming Codes

- Es un mecanismo de corrección de errores.
- Usa r bits de paridad para detectar y corregir errores de bits.
- Dado un dato con *m* bits: $m+r+1<=2^r$
  1. Los bits de paridad son ubicados en las posiciones *k*: $k = 2^i, i=0,1,2\ldots$
  2. Bit (dato/paridad) en la posición k contribuye a los bits de paridad de los en posiciones i donde:

     $k = \Sigma_x i, \; i = 2^x$
  3. Bits invertidos están en la posición i, donde:

     $i = \Sigma$ k, k = posición del bit de paridad con valor incorrecto
  4. Burst errors como paridad simple

Hamming Codes

Figure 6.8 Bit Stream Before Transmission

Data:	0	1	1	0	0	1	1	1				
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8				
Hamming code:	0	1	0	1	1	1	0	1	0	1	1	1
	p_1	p_2	m_1	p_3	m_2	m_3	m_4	p_4	m_5	m_6	m_7	m_8

Hamming Codes

FIGURE 4.9 Hamming Code for Single-Bit Error Correction

Data to send: m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8

Hamming code: p_1 p_2 m_1 p_3 m_2 m_3 m_4 p_4 m_5 m_6 m_7 m_8

Bit position 1 ↑ p_1 — Bit position 12 ↑ m_8

p_1 even parity for positions 1, 3, 5, 7, 9, 11

p_2 even parity for positions 2, 3, 6, 7, 10, 11

p_3 even parity for positions 4, 5, 6, 7, 12

p_4 even parity for positions 8, 9, 10, 11, 12

Hamming Codes

Table 6.2 Bit Position Errors and Associated Parity Errors

Erroneous Bit Position	Invalid Parity Checks	b_4, b_3, b_2, and b_1
No error	None	0000
1	p_1	0001
2	p_2	0010
3	p_1 and p_2	0011
4	p_3	0100
5	p_1 and p_3	0101
6	p_2 and p_3	0110
7	p_1, p_2, and p_3	0111
8	p_4	1000
9	p_1 and p_4	1001
10	p_2 and p_4	1010
11	p_1, p_2, and p_4	1011
12	p_3 and p_4	1100

Hamming Codes

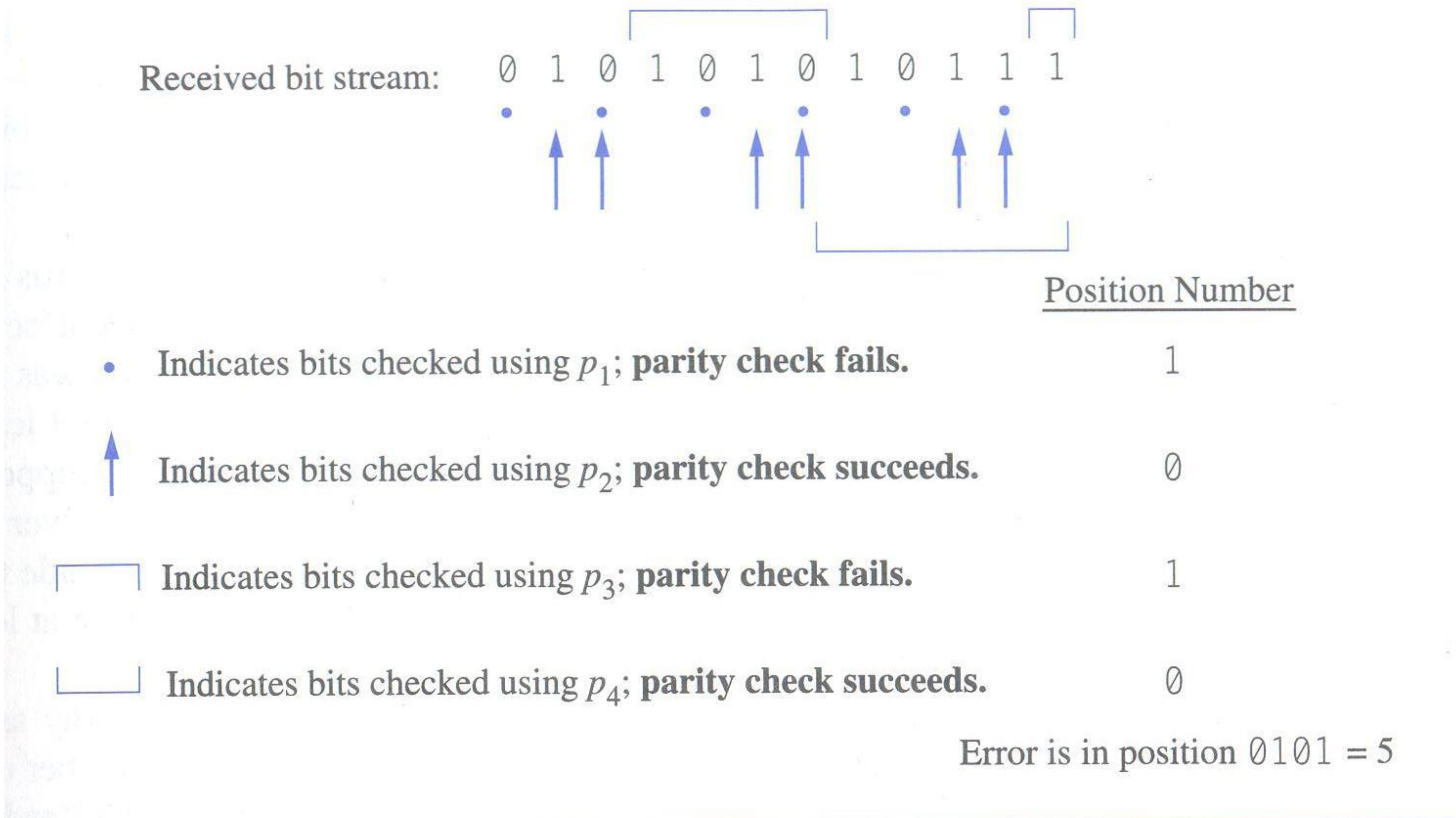


Figure 6.9 Parity Checks of Frame After Transmission

Control de flujo

Control de flujo

FIGURE 5.1 Flow Control Using Signaling

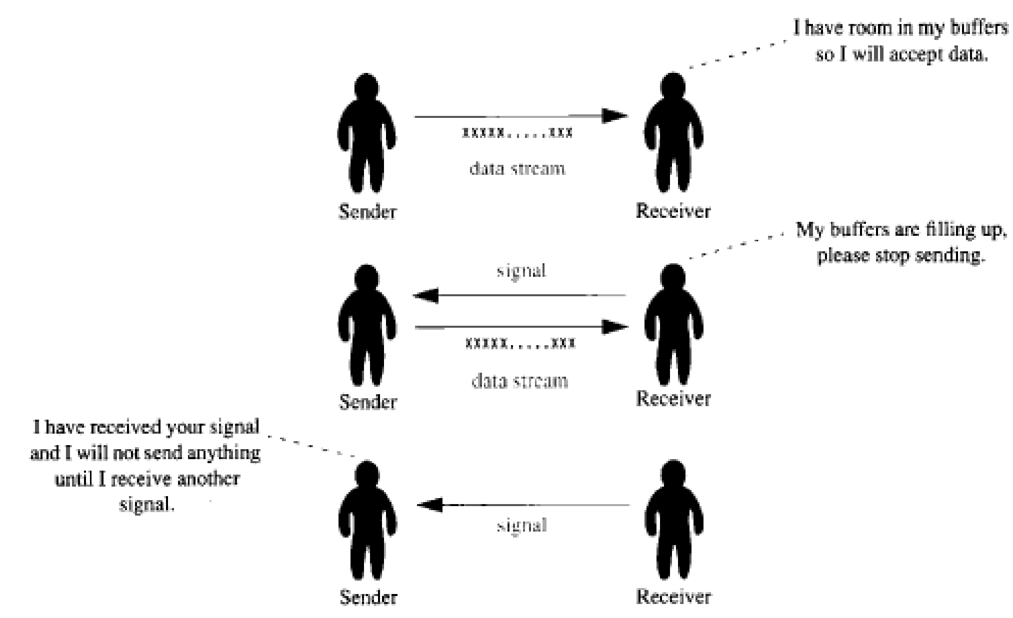
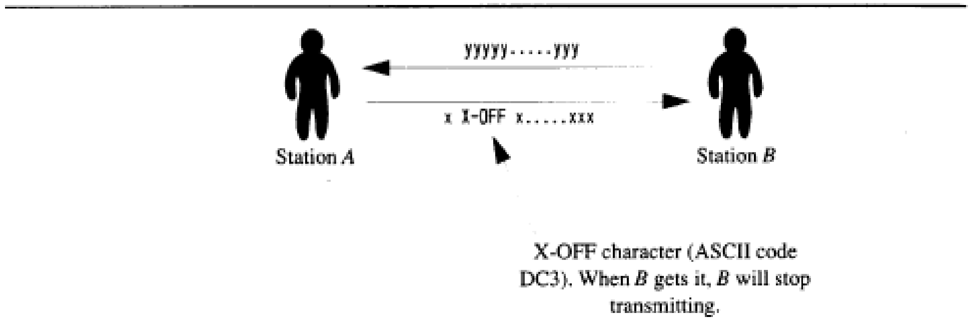


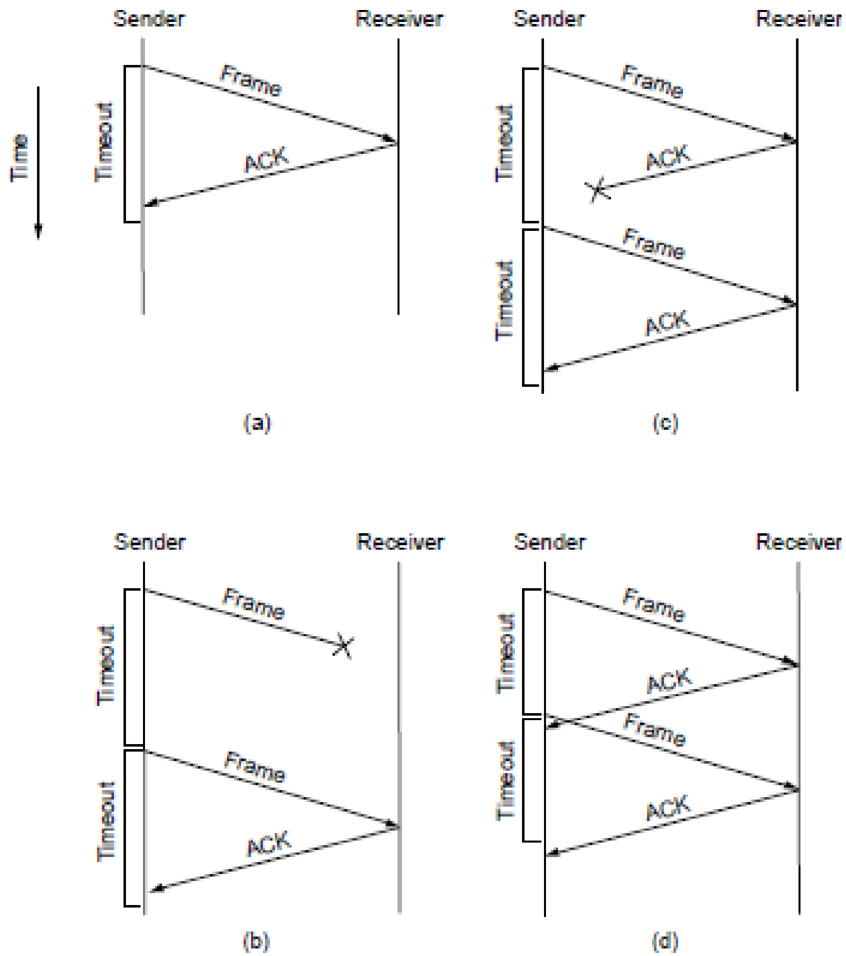
FIGURE 5.2 Flow Control Using Inband Signaling



Stop and wait

- Receptor confirma cada frame
- Transmisor espera por ack antes de enviar el siguiente frame
- Desventajas: Baja eficiencia
  - Delay*bandwidth
  - Utilización del canal : porcentaje de tiempo que el canal está transmitiendo frames
  - Data rate efectiva: el número real de bits enviados por unidad de tiempo

Stop and wait

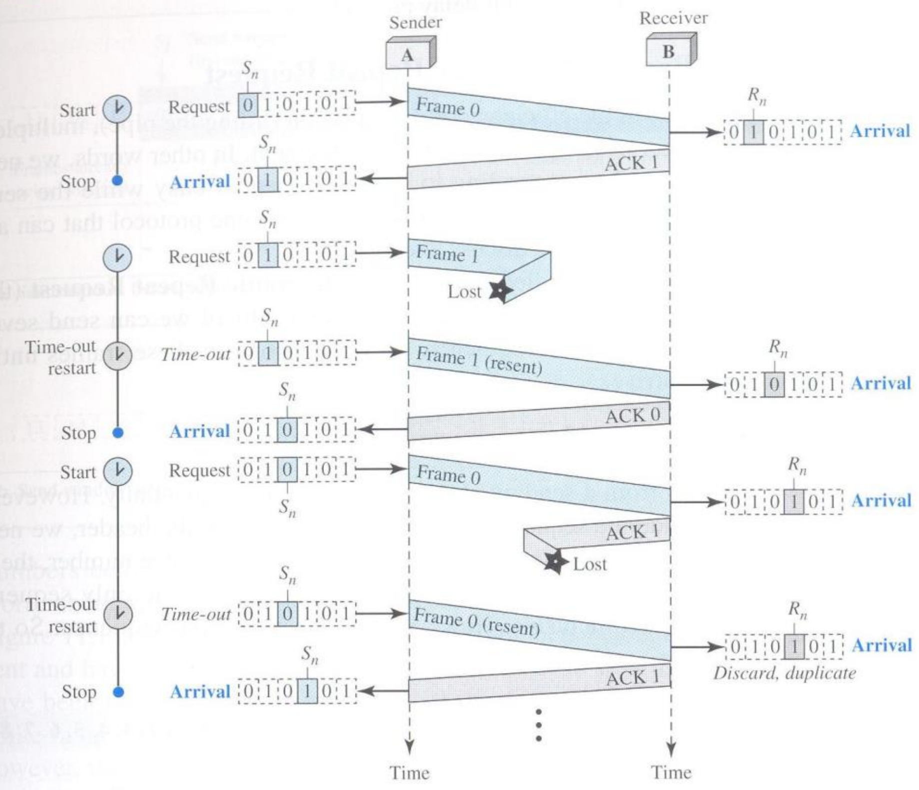


Stop and wait

Example 11.3

Figure 11.11 shows an example of Stop-and-Wait ARQ. Frame 0 is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame 0 is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

Figure 11.11 *Flow diagram for Example 11.3*



Sliding Window: go-back-n

- Transmisor asigna números de secuencia consecutiva a cada frame: 0 a $2^k - 1$
- Transmisor mantiene un window buffer de tamaño $2^k - 2$
  - Cada vez que un frame es enviado es almacenado en el buffer
  - Frames son removidos cuando son confirmados
  - Si el frame no es confirmado se retransmite junto con todos los frames pendientes
- Receptor espera recibir los frames en orden
  - Si el frame está dañado o fuera de orden es descartado
- Receptor no confirma la recepción de cada frame de forma explícita. Ack piggybacks data frames hacen referencia al frame más recientemente recibido. Ack timeout si no hay datos

Sliding Window: go-back-n

FIGURE 5.7 A Sliding Window Protocol

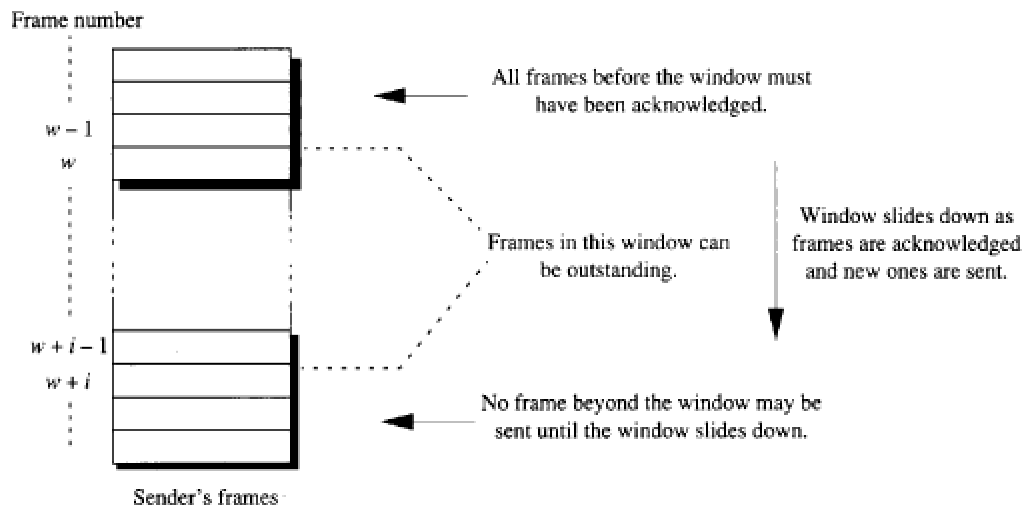


FIGURE 5.9 Typical Frame Format

Source	Destination	Number	Ack	Type	...Data...	CRC

Sliding Window: go-back-n

FIGURE 5.10 **Protocol Failure When Window Size Equals 2^K**

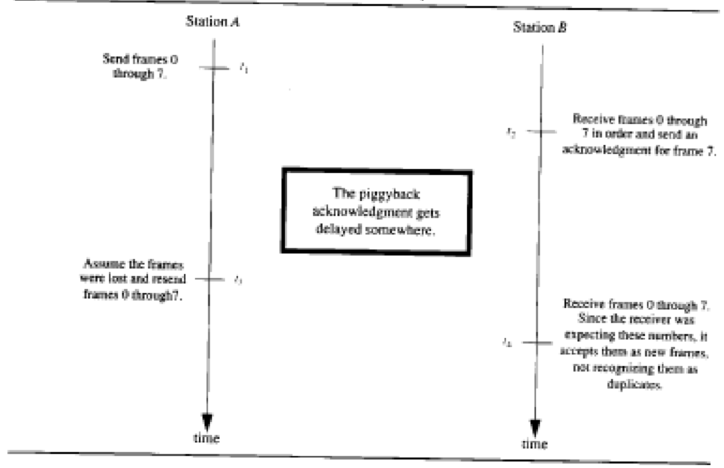
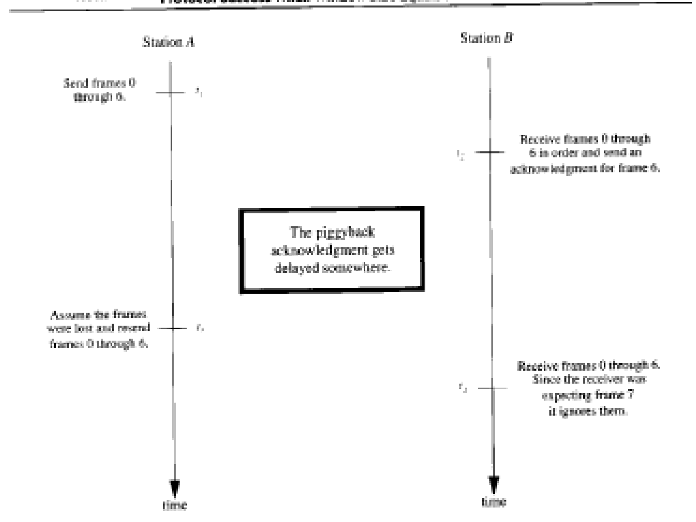
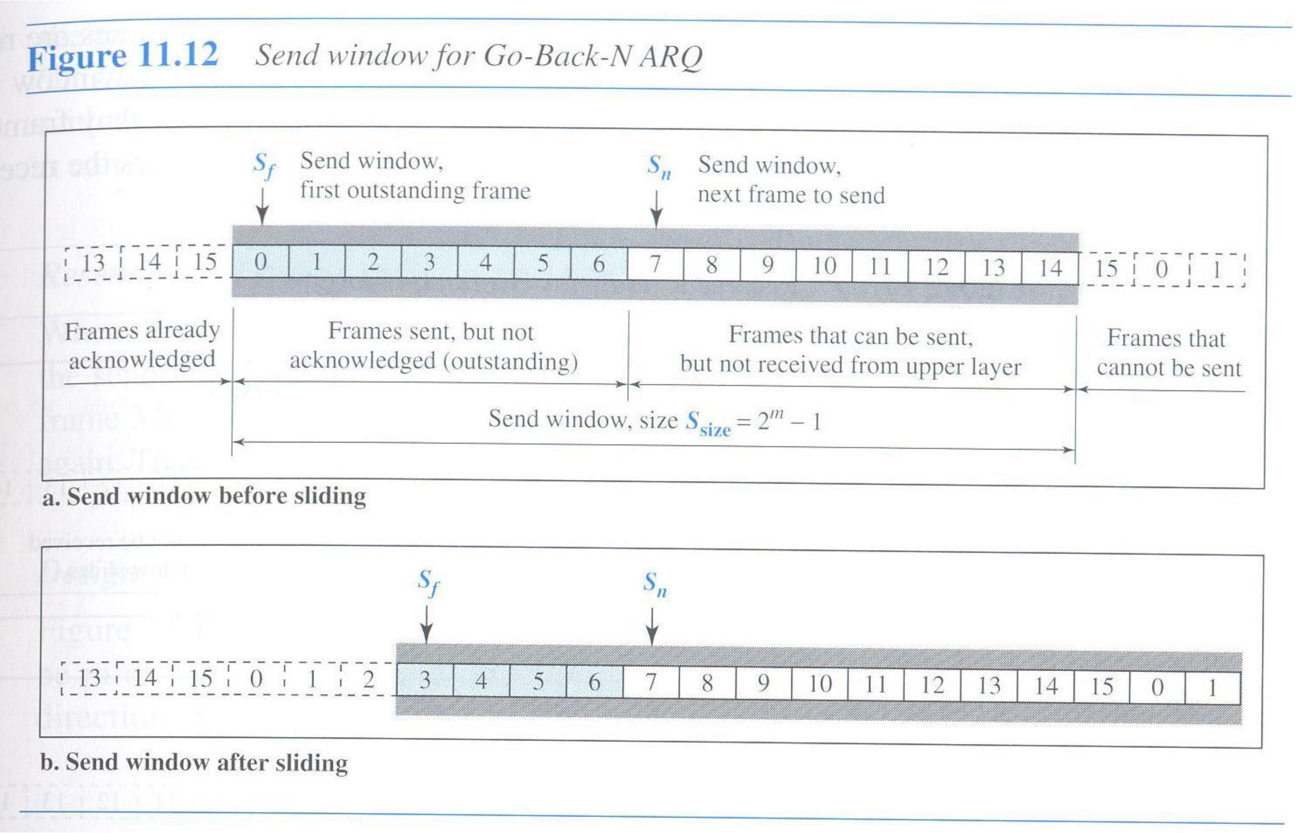


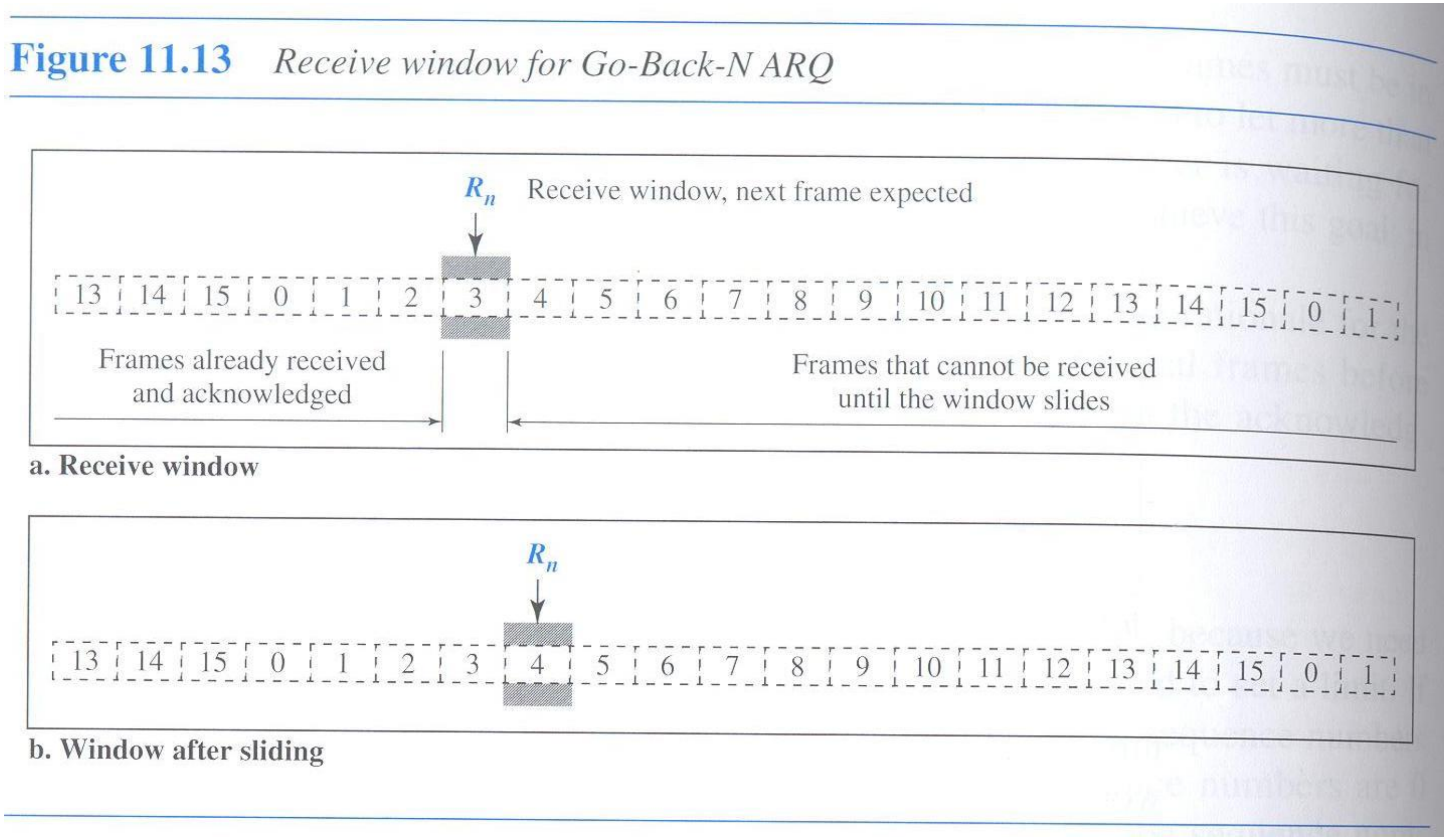
FIGURE 5.11 **Protocol Success When Window Size Equals $2^K - 1$**



Sliding Window: go-back-n



Sliding Window: go-back-n

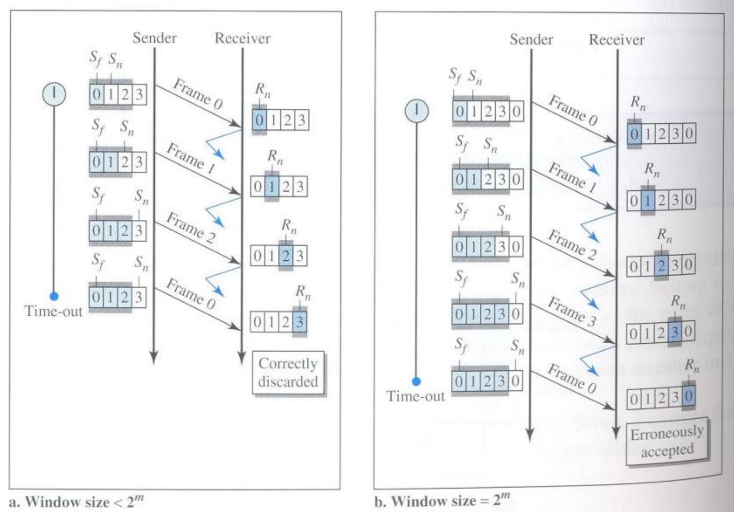


Sliding Window: go-back-n

Send Window Size

We can now show why the size of the send window must be less than 2^m. As an example, we choose $m = 2$, which means the size of the window can be $2^m - 1$, or 3. Figure 11.15 compares a window size of 3 against a window size of 4. If the size of the window is 3 (less than 2^2) and all three acknowledgments are lost, the frame 0 timer expires and all three frames are resent. The receiver is now expecting frame 3, not frame 0, so the duplicate frame is correctly discarded. On the other hand, if the size of the window is 4 (equal to 2^2) and all acknowledgments are lost, the sender will send a duplicate of frame 0. However, this time the window of the receiver expects to receive frame 0, so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is an error.

Figure 11.15 *Window size for Go-Back-N ARQ*



In Go-Back-*N* ARQ, the size of the send window must be less than 2^m; the size of the receiver window is always 1.

Sliding Window: go-back-n

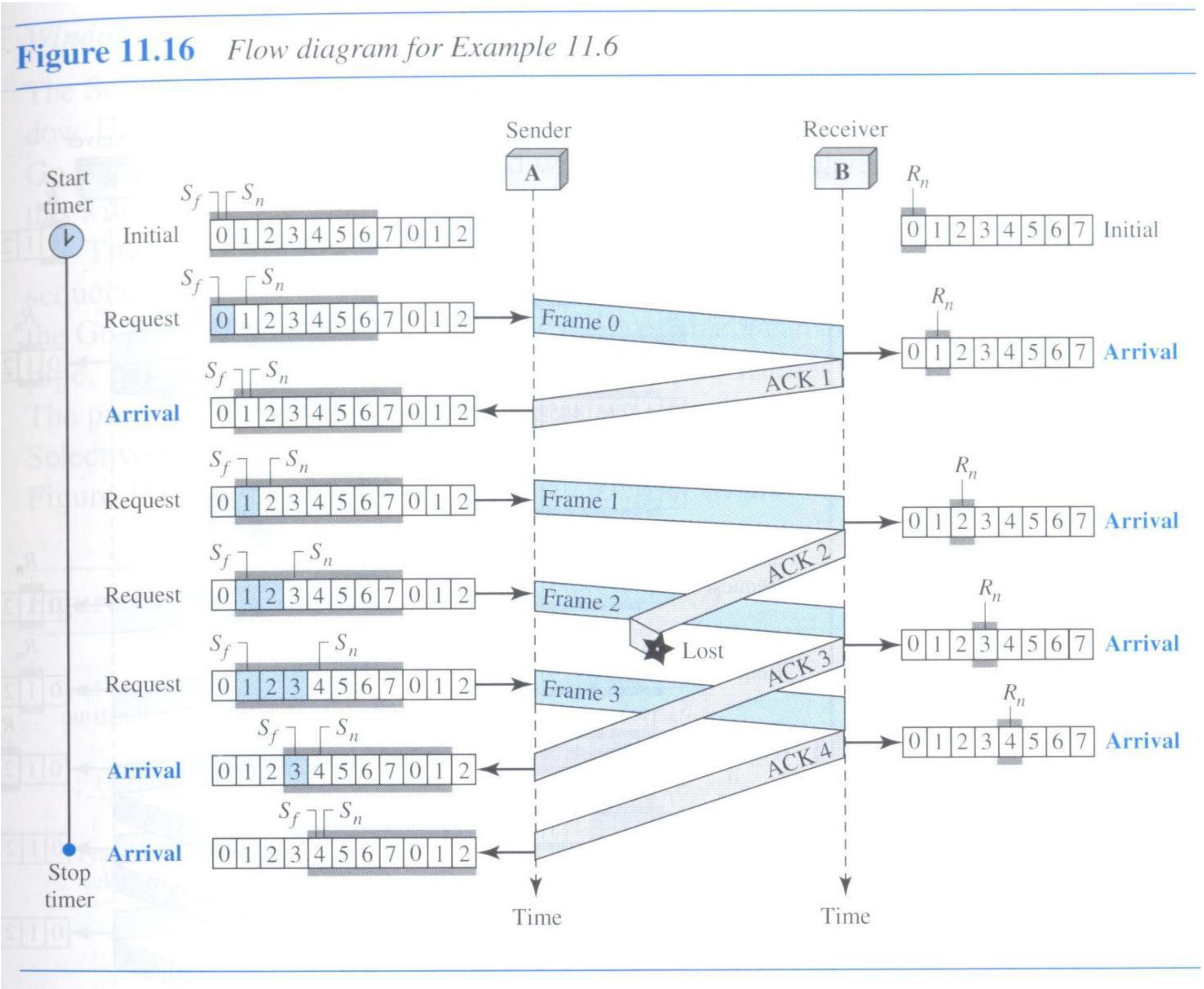
Example 11.6

Figure 11.16 shows an example of Go-Back-*N*. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost.

After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

There are four receiver events, all triggered by the arrival of frames from the physical layer.

Sliding Window: go-back-n



Sliding Window: go-back-n

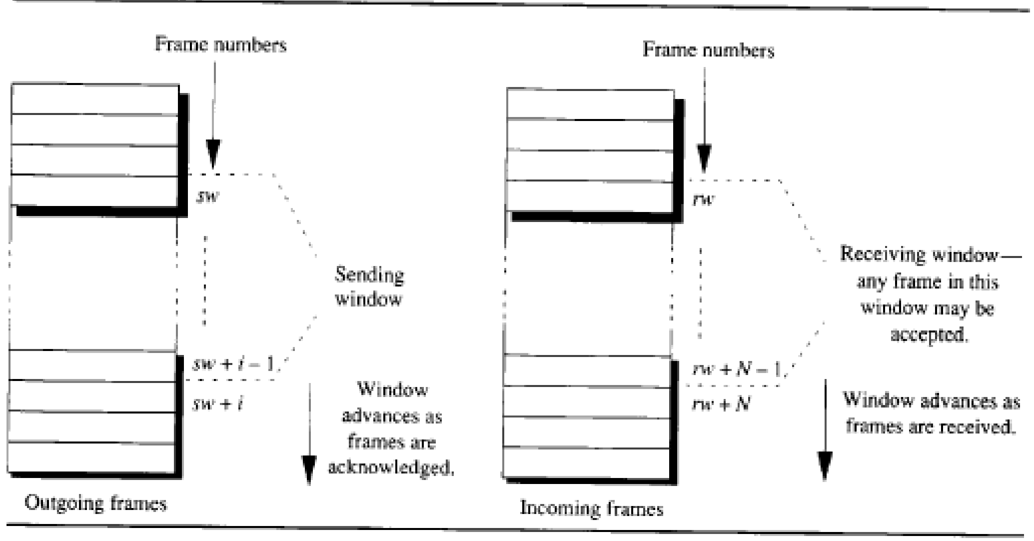
9. Grafique las ventanas para un transmisor y un receptor en un enlace punto a punto usando go-back-N con una ventana de tamaño 7. Dado lo siguiente: **[12%]**
   i. Frame 0 es enviado; frame 0 es acknowledged
   ii. Frame 1 y 2 son enviados; frames 1 y 2 son acknowledged
   iii. Frames 3, 4 y 5 son enviados; frame 4 es acknowledged pero el timer para el frame 5 expira
   iv. Frames 5, 6, y 7 son enviados; frames del 4 al 7 son acknowledged

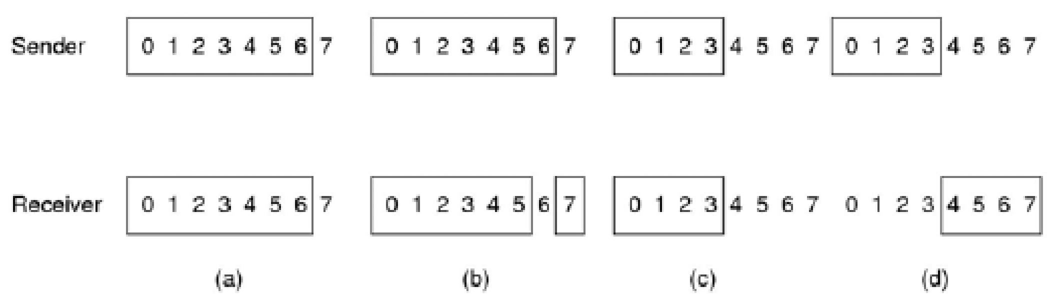
Sliding Window: selective repeat

- Similar a go-back-n pero:
  - Receptor:
    - Utiliza **ventana también**, la cual define los frames que pueden ser recibidos
    - Frames fuera de orden son almacenados en el buffer
    - Nack enviado para frames dañados o frames demorados
    - Ack enviados para frames f_i, donde i es el mayor número de la secuencia tales como todos los frames f_j, j<i que han sido recibidos
  - Transmisor:
    - Retransmite solo si timed out y Nack-ed frames
- Tamaño de la ventana w: $w_{transmisor} + w_{receptor} <= 2^k$

Sliding Window: selective repeat

FIGURE 5.13 Sending and Receiving Windows for Selective Repeat Protocol





Sliding Window: selective repeat

FIGURE 5.14 **Protocol Failure: Receiving Window Size is Greater Than 2^{K-1})**

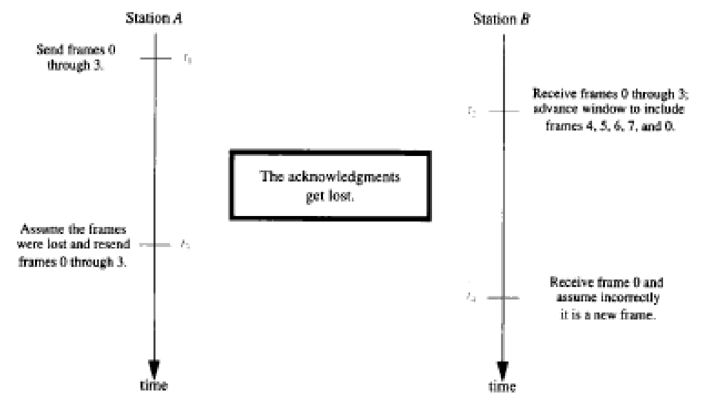
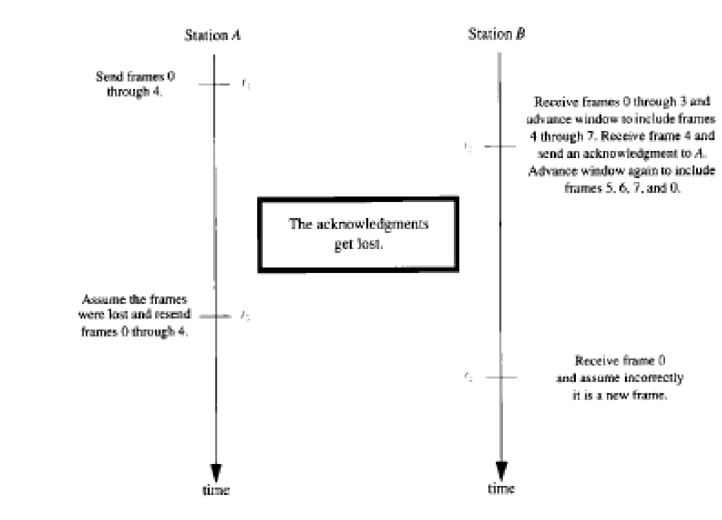
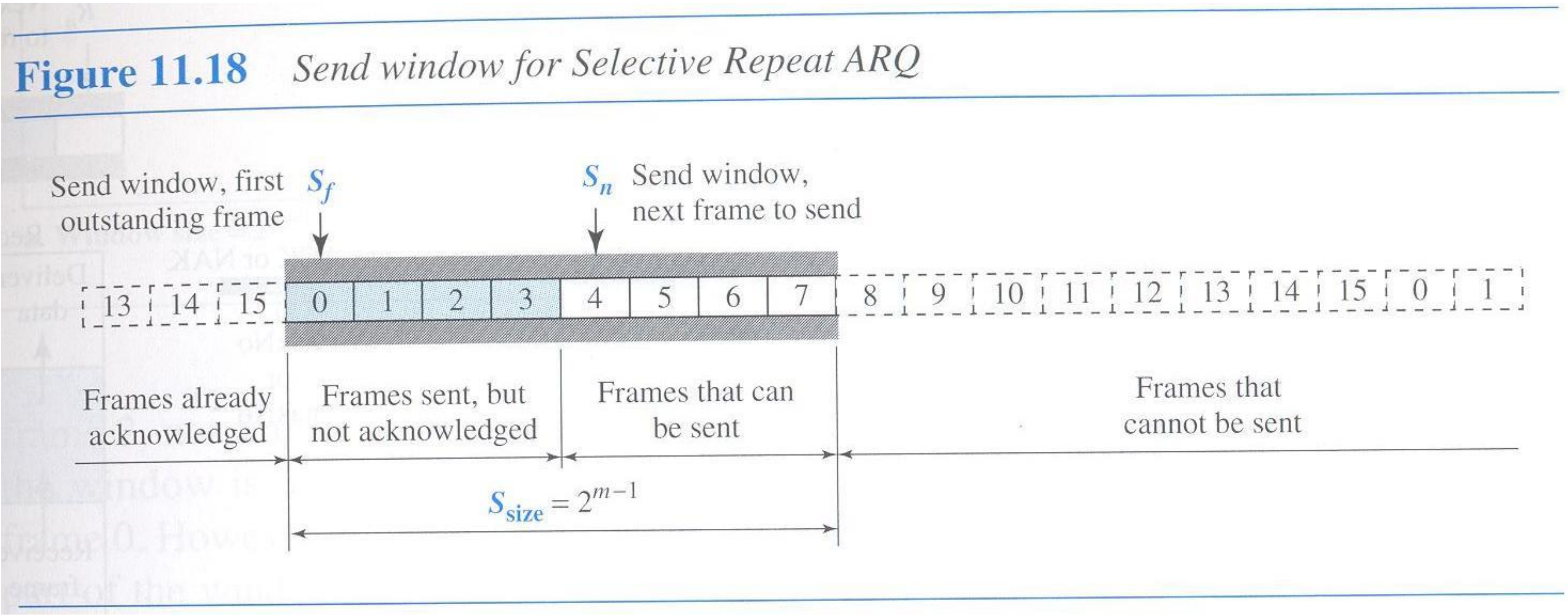


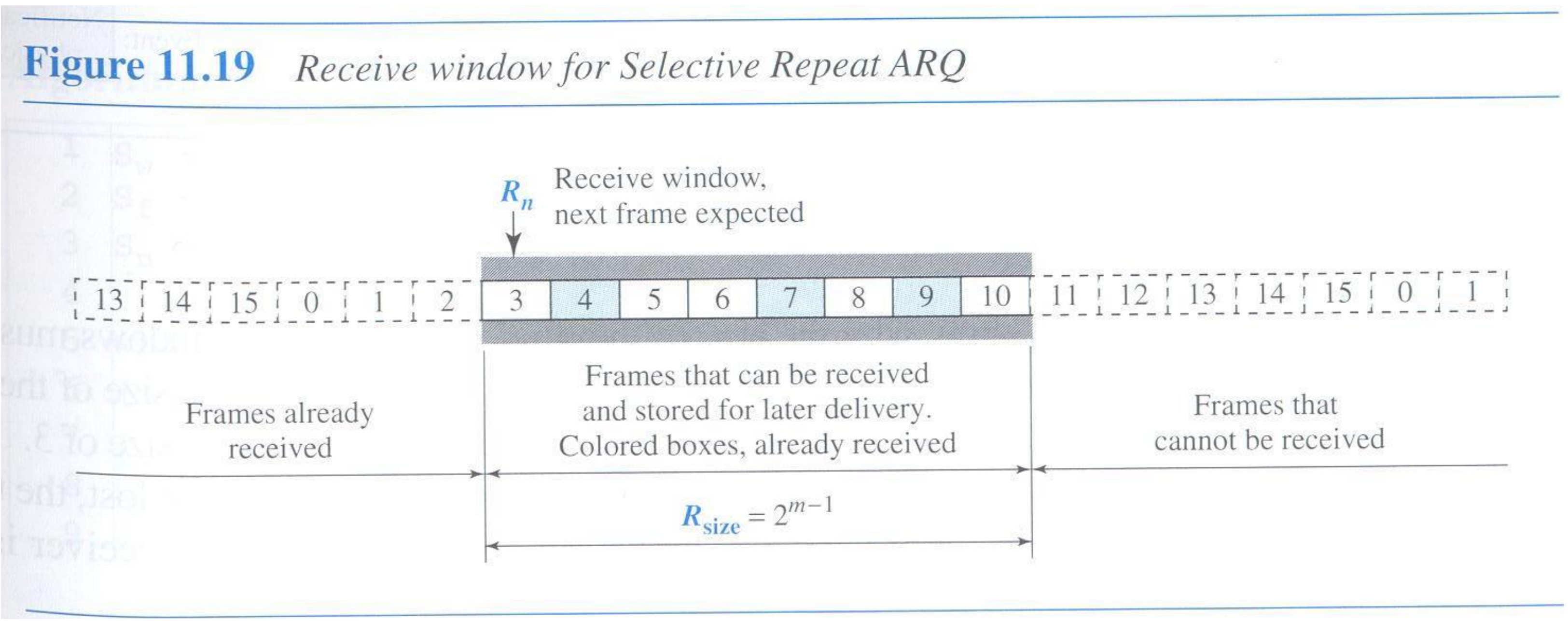
FIGURE 5.15 **Protocol Failure: Sending Window Size is Greater Than 2^{K-1}**



Sliding Window: selective repeat

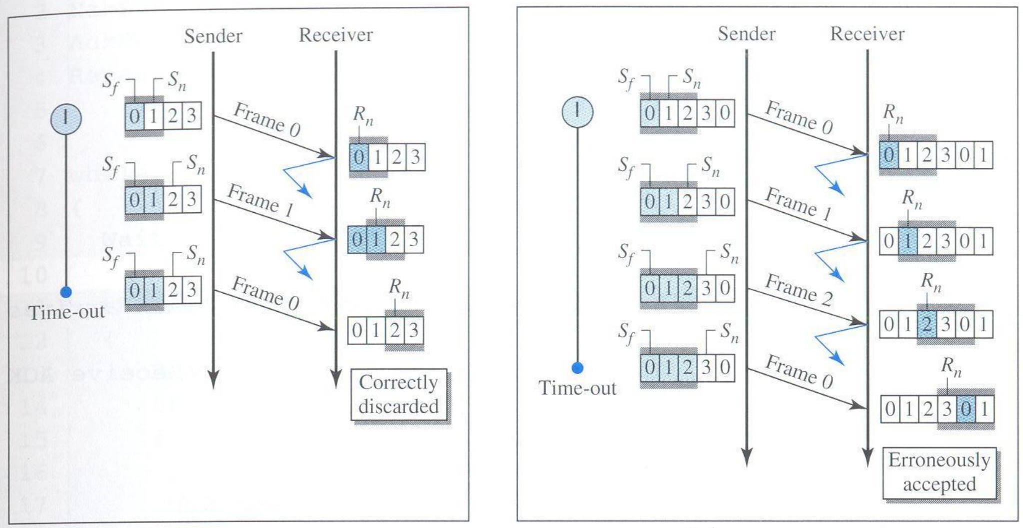


Sliding Window: selective repeat



Sliding Window: selective repeat

Figure 11.21 *Selective Repeat ARQ, window size*



a. Window size $= 2^{m-1}$ b. Window size $> 2^{m-1}$

frame 2, not frame 0, so this duplicate frame is correctly discarded. When the size of the window is 3 and all acknowledgments are lost, the sender sends a duplicate of frame 0. However, this time, the window of the receiver expects to receive frame 0 (0 is part of the window), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is clearly an error.

In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m.

Sliding Window: selective repeat

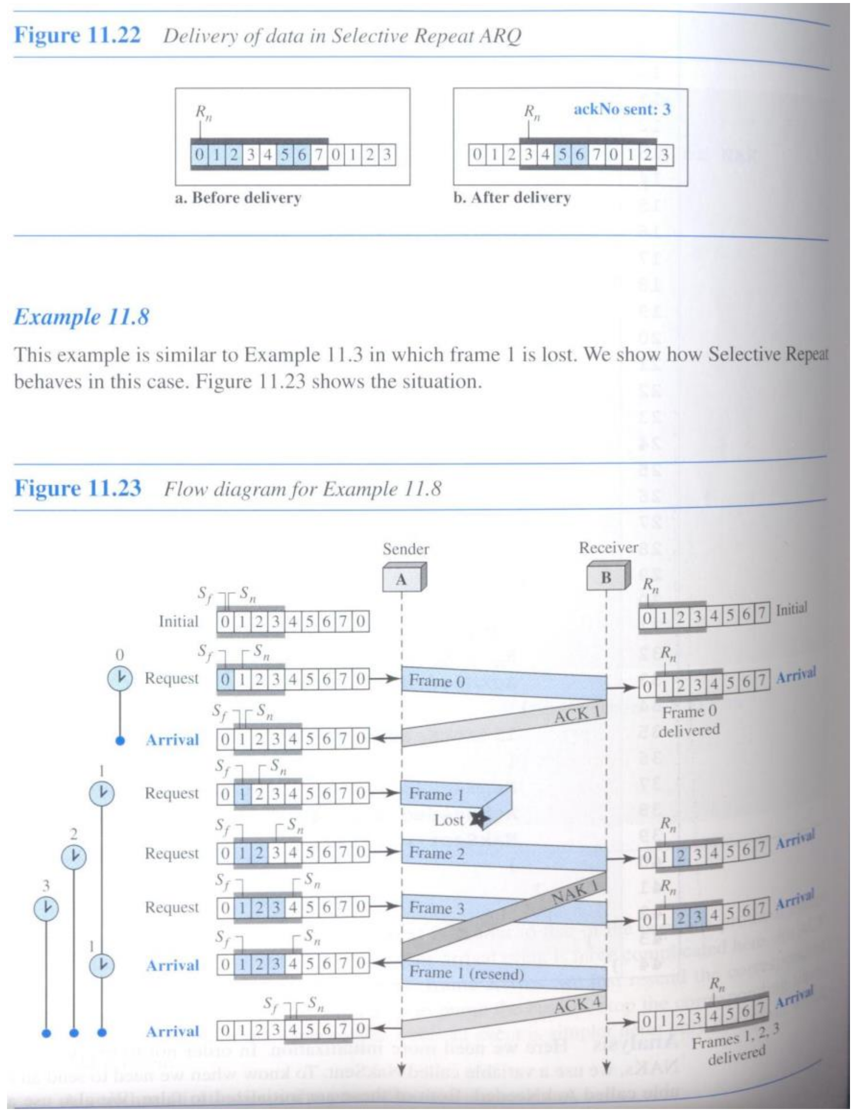


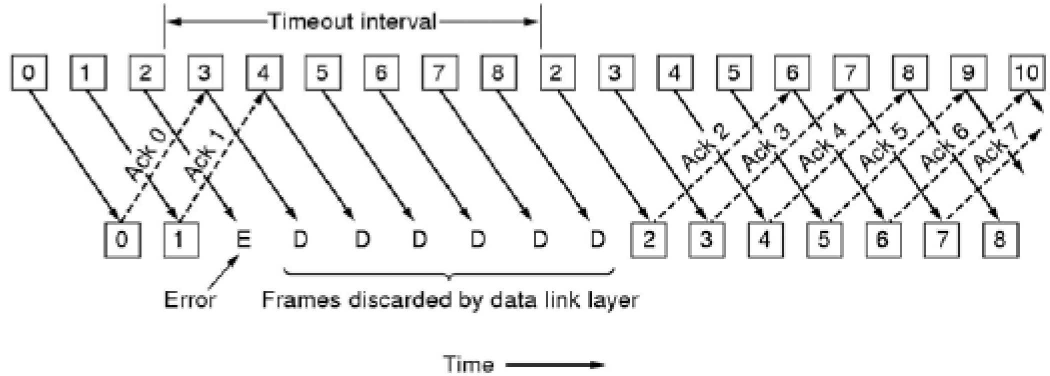
Figure 11.22 *Delivery of data in Selective Repeat ARQ*

Example 11.8

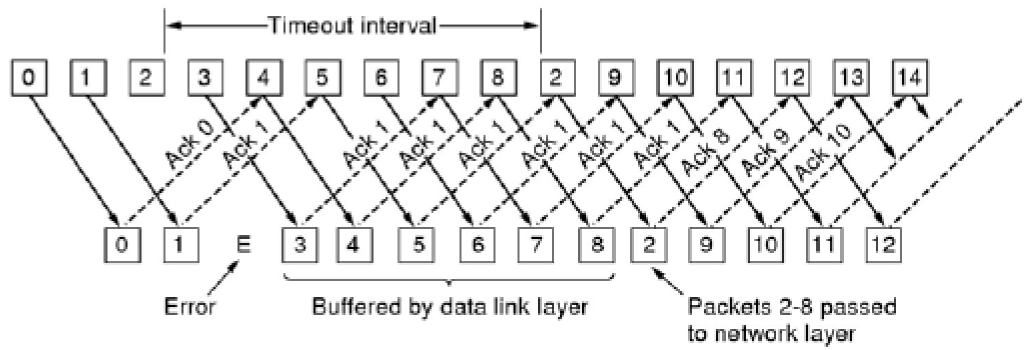
This example is similar to Example 11.3 in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure 11.23 shows the situation.

Figure 11.23 *Flow diagram for Example 11.8*

go-back-n vs. selective repeat



(a)



(b)

Puntos para recordar

- Diferencias entre detección y corrección de errores
- Hamming code
- Esquemas de graficación de cada mecanismo de control de flujo

Próxima Sesión

- El problema de la asignación de canal
- Cableado estructurado
- Direccionamiento: Direcciones MAC y ARP