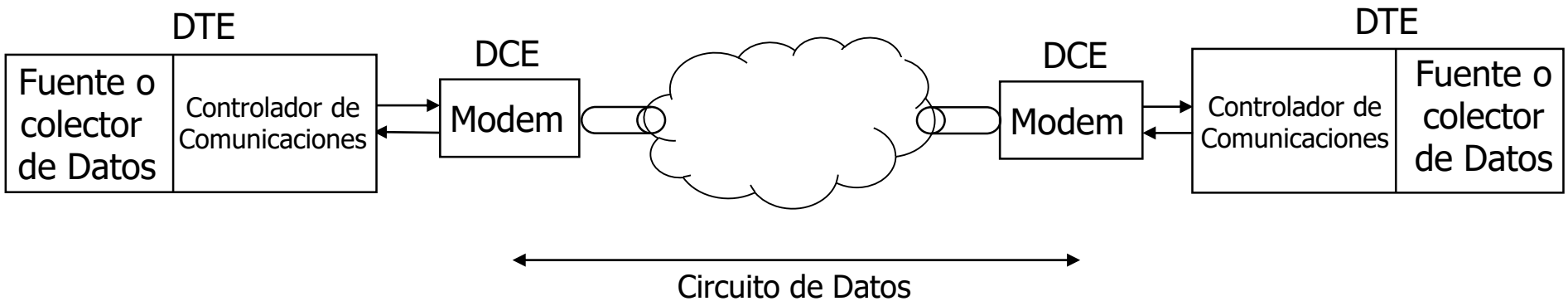
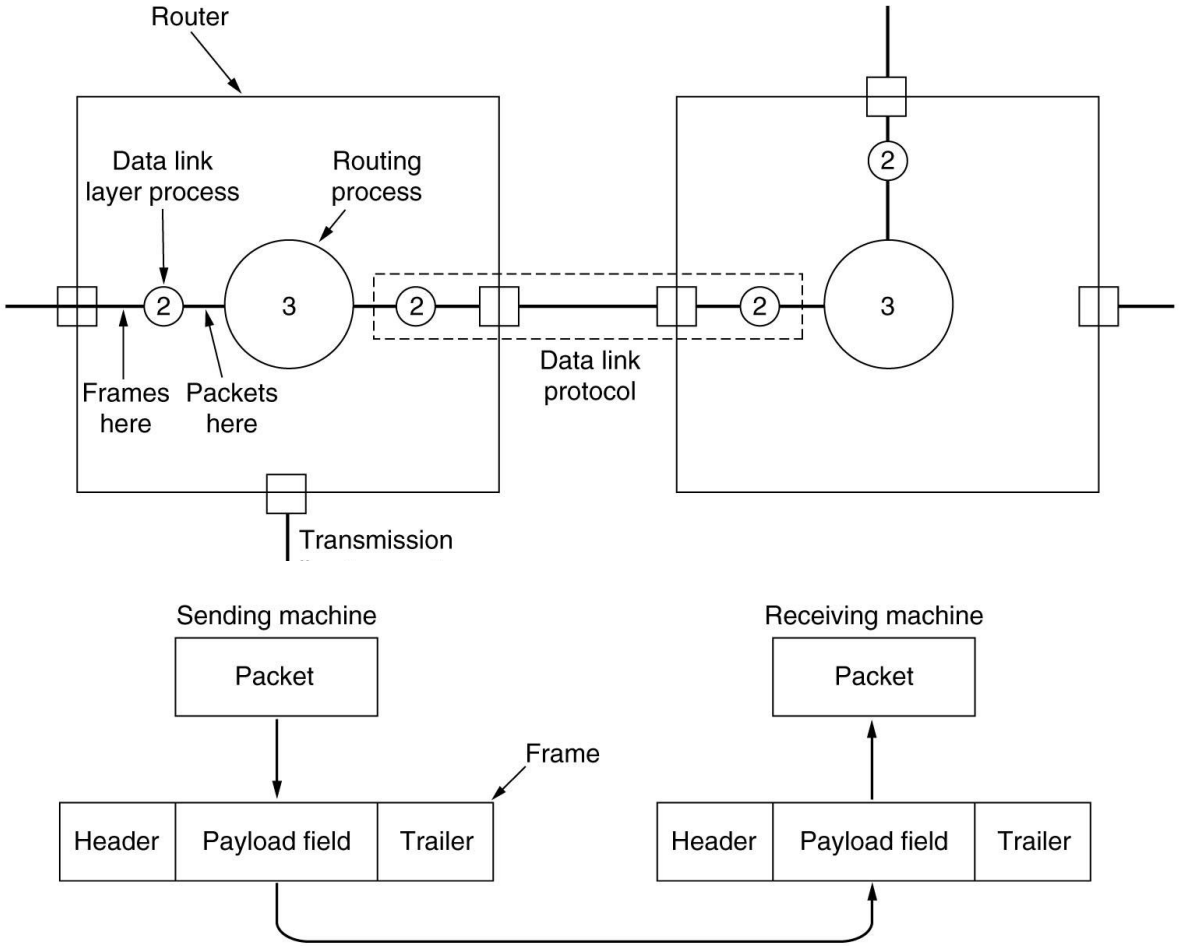
Capa de enlace de datos

Capa de enlace de datos

- Transmisión confiable de tramas entre equipos directamente conectados.



Capa de enlace de datos



Funciones principales

- Entramado
- Detección de errores
- Corrección de errores (posiblemente)
  - Enlaces actuales extremadamente confiables, excepto en enlaces inalámbricos
- Control de flujo

Control de errores

- Errores de transmisión
  - Detección
  - Retransmisión (Acuse de recibo positivo, PAR)
- Tramas duplicadas
  - Números de secuencia
- Tramas perdidas
  - Petición de retransmisión

Detección de errores

- Códigos de bloques: se agregan bits de redundancia a cada bloque de información transmitida.
- VRC y LRC (paridad)
- Checksum
- CRC
  - $x^{16} + x^{12} + x^5 + 1$
  - 10000100110000010001110110110111

Detección de paridad

- Populares para la transmisión de caracteres (ASCII)
- Se agrega un bit de paridad con un valor tal que todos los bits en la palabra sean pares (paridad par) o nones (paridad non)
- Al recibir los datos, se hace la misma operación. Si los bits coinciden, se considera que no hubo error.
  - Sin embargo, no es capaz de detectar errores en parejas de bits

Detección por paridad

	H	o	l	a	LRC
	0	0	0	0	1
	1	1	1	1	1
	0	1	1	1	0
	0	0	0	0	1
	1	1	1	0	0
	0	1	1	0	1
	0	1	0	0	0
	0	1	0	1	1
VRC	1	1	1	0	0

Checksum

- Calcula la suma de los datos a enviar
  - Muy sencillo, pero poco confiable: varios errores pueden producir la misma suma

Valor	ChkSum	Valor	ChkSum
001100	12	001101	13
100101	37	100101	37
000111	7	000011	3
010001	17	010100	20
Total:	73		73

Cyclic redundancy check CRC

- Cadenas de bits, D(x), como polinomios con coeficientes “0” y “1”
  - $100101 = 1*x^5+0*x^4+0*x^3+1*x^2+0*x^1+1$
- Se calcula el remanente de dividir la secuencia entre un polinomio generador G(x)
  - Aritmética en módulo 2. No hay acarreo ni préstamo. Suma y resta son idénticas, y equivalen a XOR

+/-	0	1
0	0	1
1	1	0

CRC

- Sea r el grado de G(x). Agregar *r* ceros a la derecha de D(x).
  - $D'(x) = D(x) * 2^r$
- Dividir D'(x) por G(x) en aritmética módulo 2
- Sustraer (o agregar) el remanente (el cual tiene r o menos bits) a D'(x). El resultado, T(x) es la trama con el verificador de integridad
- Transmitir T(x) y repetir la operación.
- G(x) debe ser elegido de forma tal que la probabilidad de que D(x) se divisible por G(x) sea extremadamente baja

CRC

- Muy fácil de implementar con registro de corrimientos y compuerta XOR. Se calcula conforme se va transmitiendo/recibiendo la secuencia
- Detecta
  - Todos los errores de un bit
  - Casi todos los errores de dos bits
  - Cualquier número impar de errores
  - Todas las ráfagas =< n, la longitud de G(x)

Ejemplo de un circuito CRC

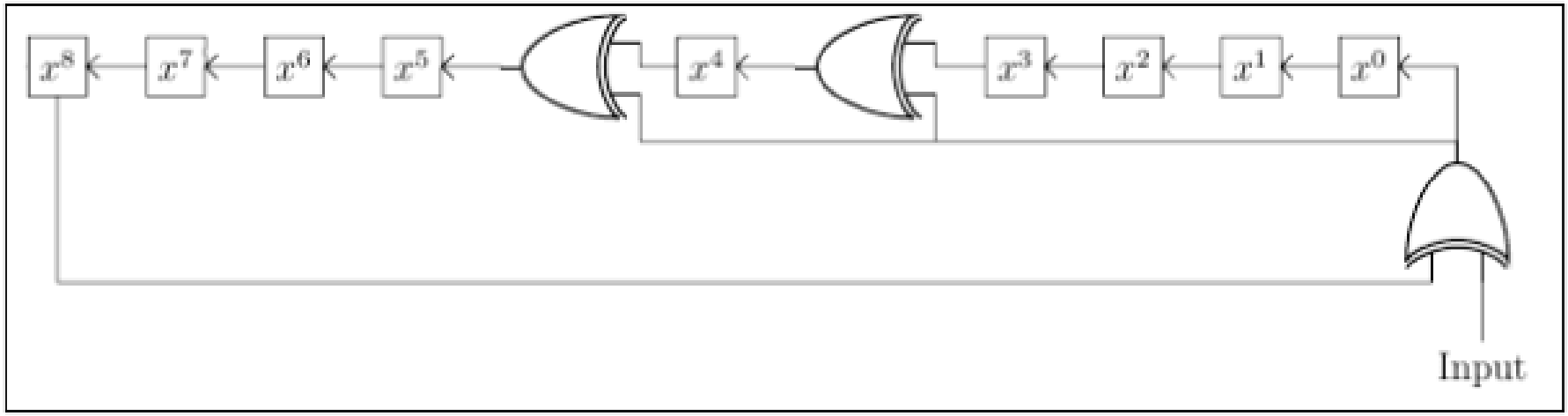
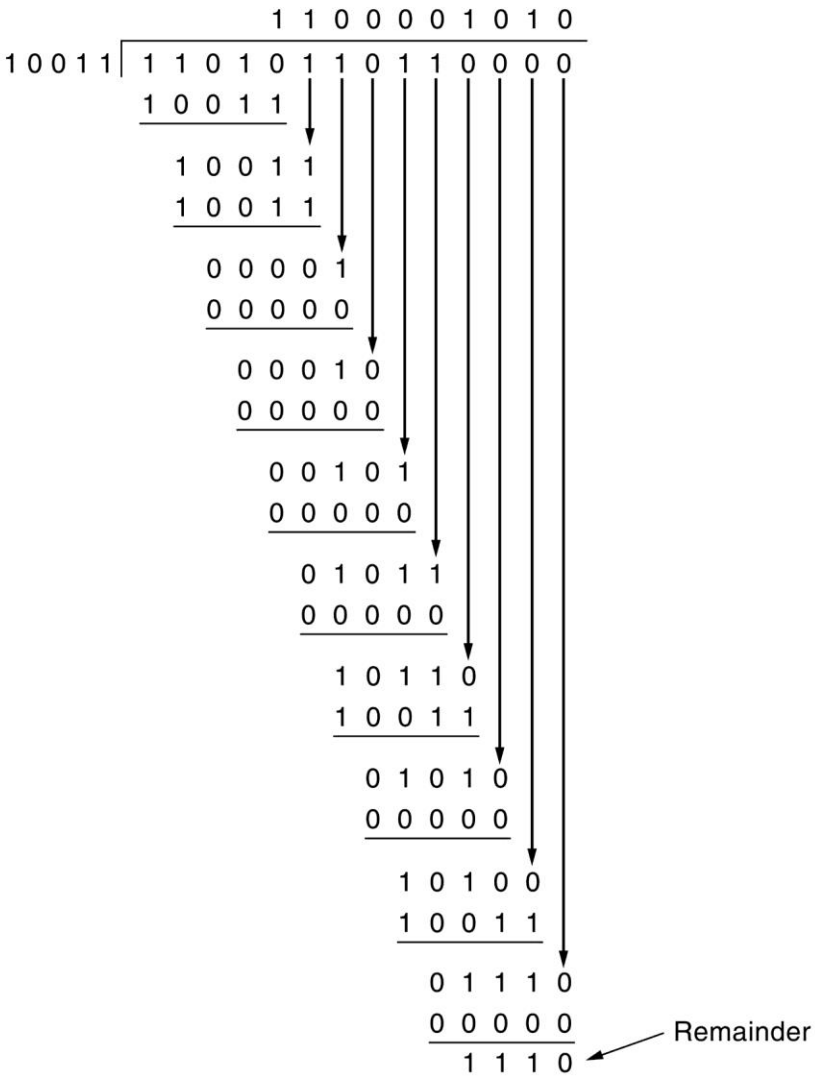


Figure 1. Architectural representation of a CRC-8 with a polynomial of x8+x5+x4+x0

Frame : 1 1 0 1 0 1 1 0 1 1

Generator: 1 0 0 1 1

Message after 4 zero bits are appended: 1 1 0 1 0 1 1 0 1 1 0 0 0 0



Transmitted frame: 1 1 0 1 0 1 1 0 1 1 1 1 0

Algunos polinomios populares

CRC	G(x)
CRC-8	$X^8+X^2+X^1+1$
CRC-10	$X^{10}+X^9+X^5+X^4+X^1+1$
CRC-12	$X^{12}+X^{11}+X^3+X^2+X^1+1$
CRC-16	$X^{16}+X^{15}+X^2+1$
CRC-CCITT	$X^{16}+X^{12}+X^5+1$
CRC-32	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$

Códigos correctores de errores

La intención es agregar suficiente redundancia para que además de detectar el error, indique cuál es o son los bits erróneos en la trama recibida

- **Distancia Hamming:** Cantidad de bits en las que difieren la trama emitida y la recibida. Simplemente se hace el XOR de ambas y se cuenta la cantidad de 1's.
- Para corregir *d* errores se necesita un código de distancia 2*d* +1. Un ejemplo sencillo podría ser con un código de sólo cuatro palabras válidas:

  0000000000, 0000011111, 1111100000 y 1111111111

  lo que significa que con distancia 5 se pueden corregir errores dobles o detectar errores cuadruples (no ambos al mismo tiempo) …

- Si llega 0000000111 y se esperan a lo sumo 2 errores, la palabra correcta es 0000011111. Con 3 errores hay ambigüedad …

Código Hamming

- Para una trama de m bits se pueden construir 2^m mensajes posibles y habrá m mensajes con un error cercano. Los r bits de redundancia deben cumplir la siguiente condición:

$$m + r + 1 \leq 2^r$$

- En 1950 Richard Hamming ideó un método posicional donde en una secuencia de bits, las posiciones potencias de 2 son los r bits de verificación. El resto se rellena con los m bits de los datos
- Cada m-esimo bit de datos puede estar incluido en varios cálculos de bits de verificación
- Para determinar a cuáles bits de verificación contribuye cada bit de dato en la posición k, se reescribe k como una suma de potencia de 2. Por ejemplo, 11 = 1 +2 + 8 …
- En cada bit de verificación se calcula el bit de paridad de los bits de datos que le corresponden

Ejemplo Código de Hamming

Palabras a transmitir: 100110, 011010, 001001

1) Colocarlas en la tabla, en su posición correspondiente

1	2	3	4	5	6	7	8	9	10
		1		0	0	1		1	0
		0		1	1	0		1	0
		0		0	1	0		0	1

2,3) Calcular qué bits de datos afectan a los de verificación

3=2+1
5=4+1
6=4+2
7=4+2+1
9=8+1
10=8+2

P1=f(3, 5, 7, 9)
P2=f(3, 6, 7, 10)
P4=f(5, 6, 7)
P8=f(9,10)

4) Calcula los bits de verificación (Paridad non)

1	2	3	4	5	6	7	8	9	10
0	1	1	0	0	0	1	0	1	0
1	0	0	1	1	1	0	0	1	0
1	1	0	0	0	1	0	0	0	1

5) Varios bits se afectaron.
En el receptor se calcula la nueva paridad

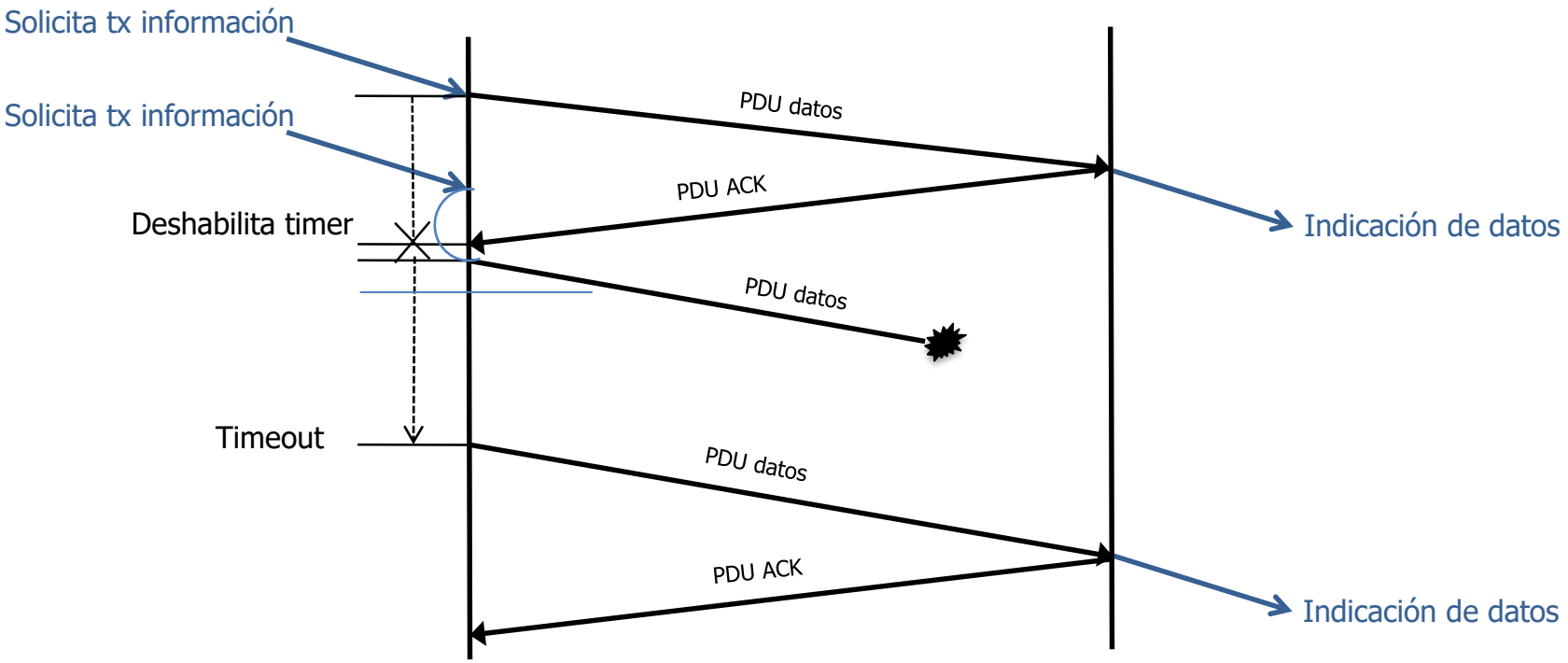
1	2	3	4	5	6	7	8	9	10
1	**0**	**0**	**0**	0	0	1	0	1	0
1	**1**	0	**0**	1	**0**	0	0	1	0
0	1	0	0	0	1	0	**1**	**1**	1

Paridad original	0	1	0	0		1	0	1	0		1	1	0	0
Nueva paridad	1	0	0	0		1	1	0	0		0	1	0	1
Diferencia	1	1	0	0		0	1	1	0		1	0	0	1
Bit afectado	3					6					9			

Control de errores – Protocolo send and wait

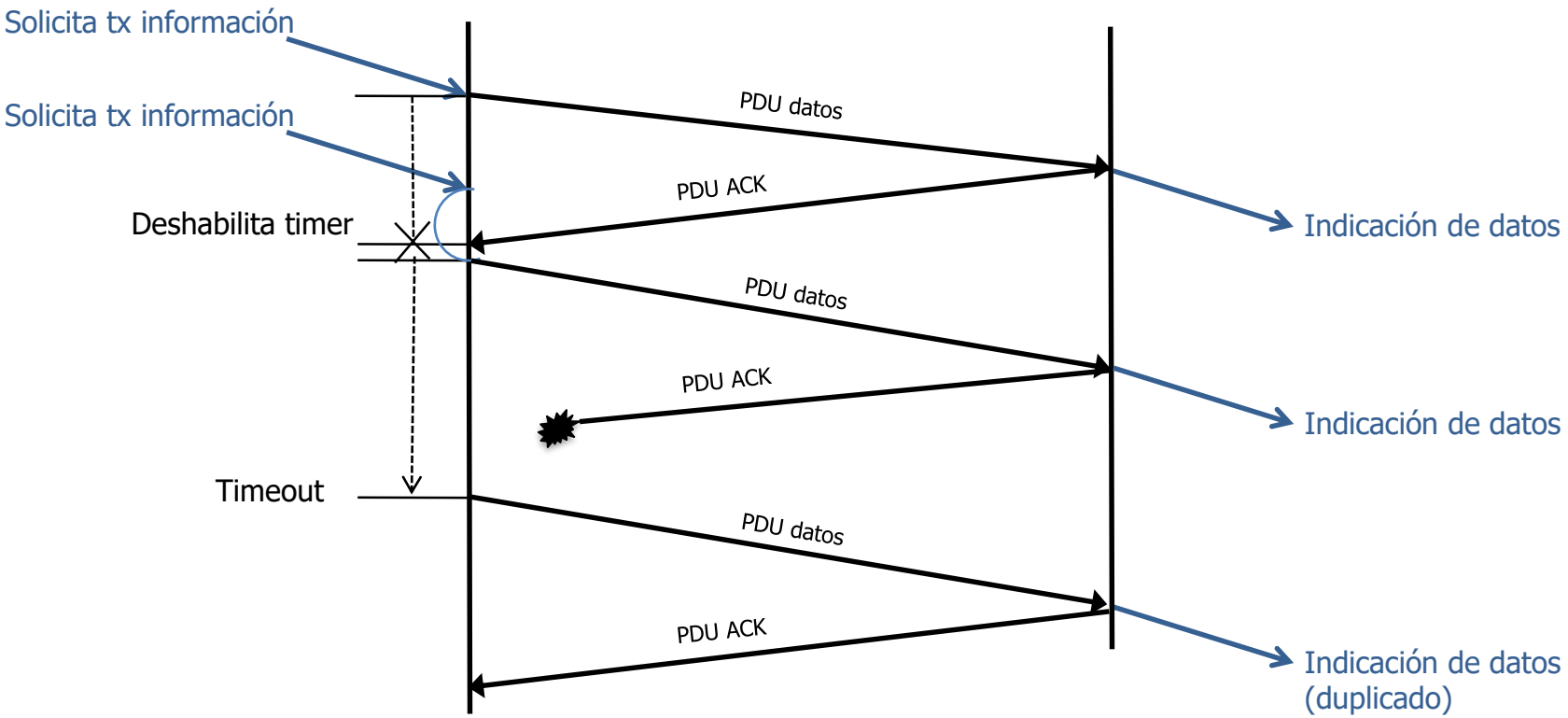
- Emisor:
  1. Toma la secuencia a transmitir y calcula el CRC
  2. Envía el paquete y arma temporizador
  3. Espera acuse de recibo
  4. Si recibe el acuse, regresa a uno, de lo contrario va a 2
- Receptor
  1. Espera paquete
  2. Verifica CRC
  3. Si el paquete es correcto, envía ACK y entrega a la capa superior. De lo contrario, descártalo
  4. Regresa a 1

Protocolo send and wait



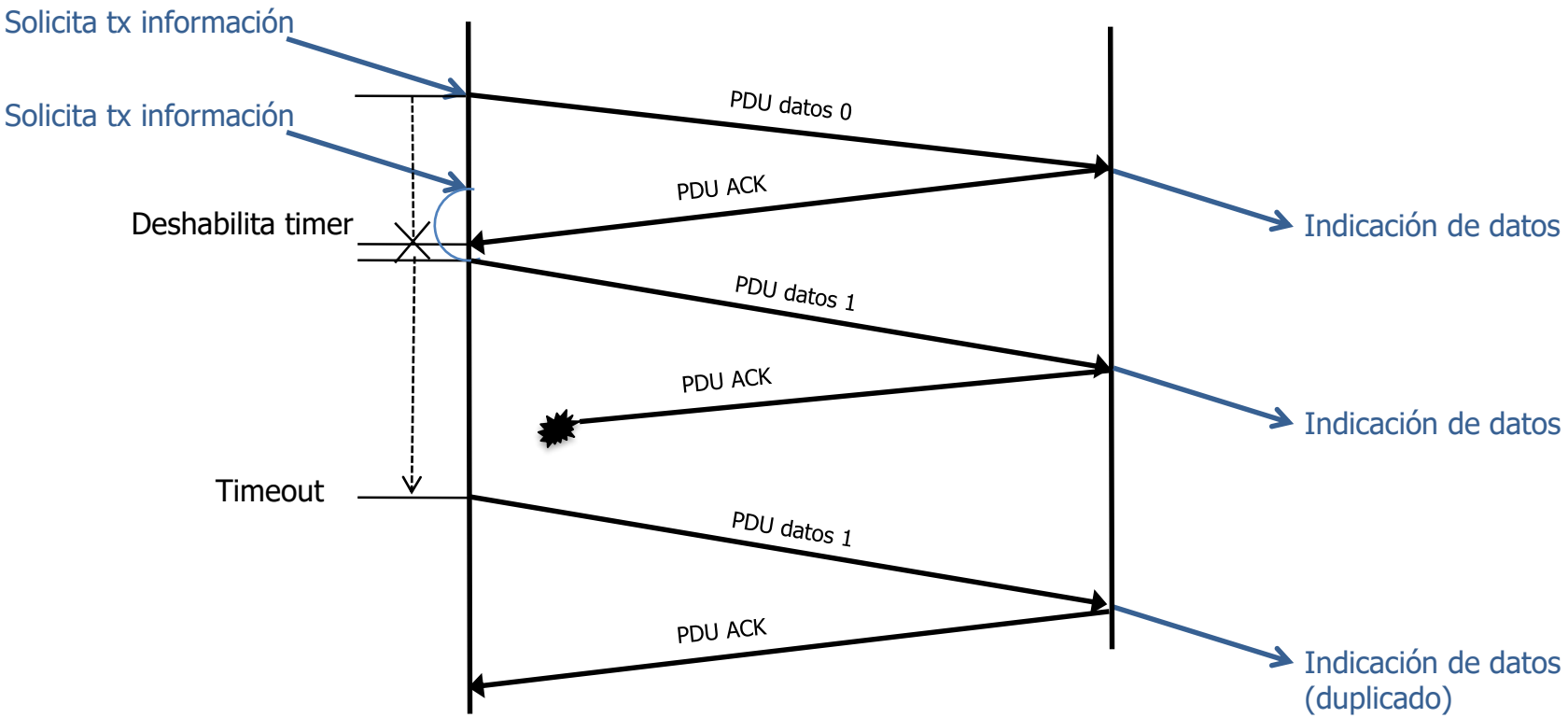
Retransmisión por PDU perdido.
¿Qué pasa si ACK se pierde?

Protocolo send and wait



Retransmisión PDU duplicado.
¿Cómo corregir este problema?

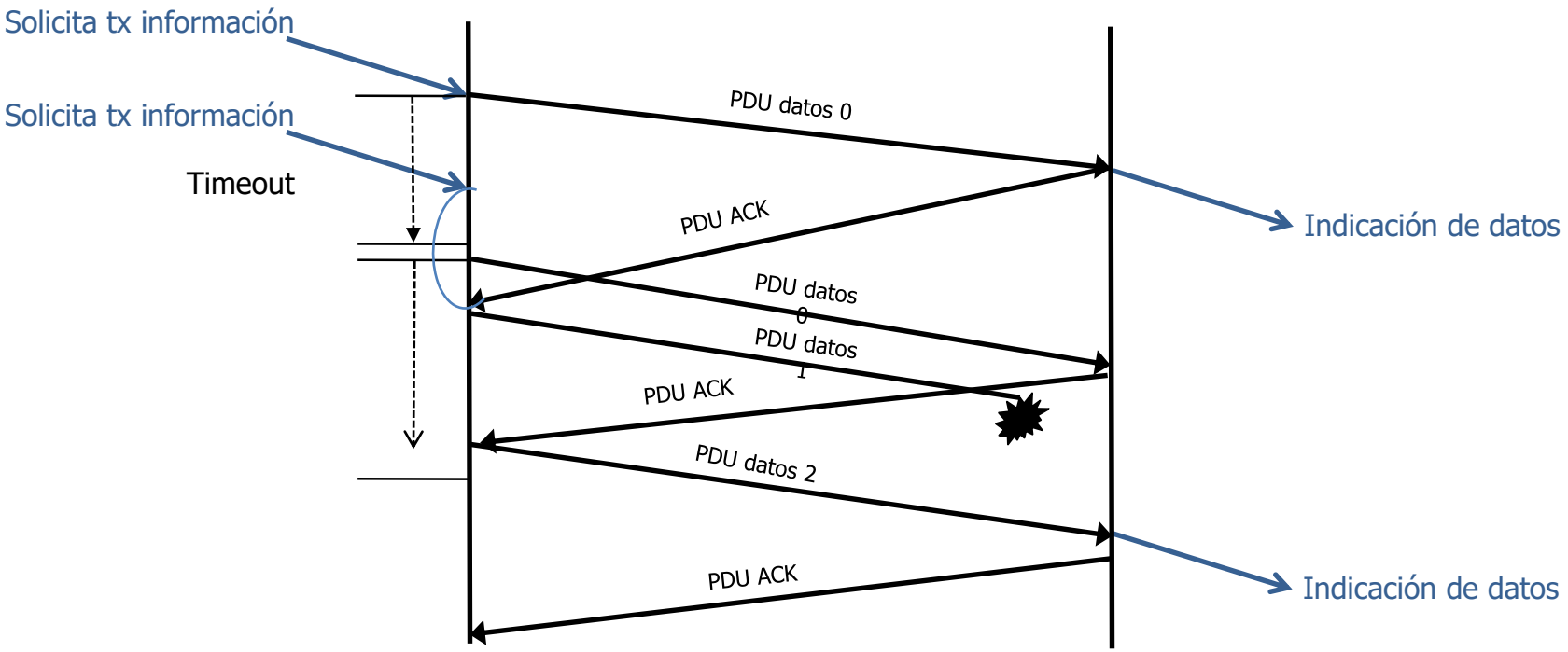
Protocolo send and wait



Enumeración PDU

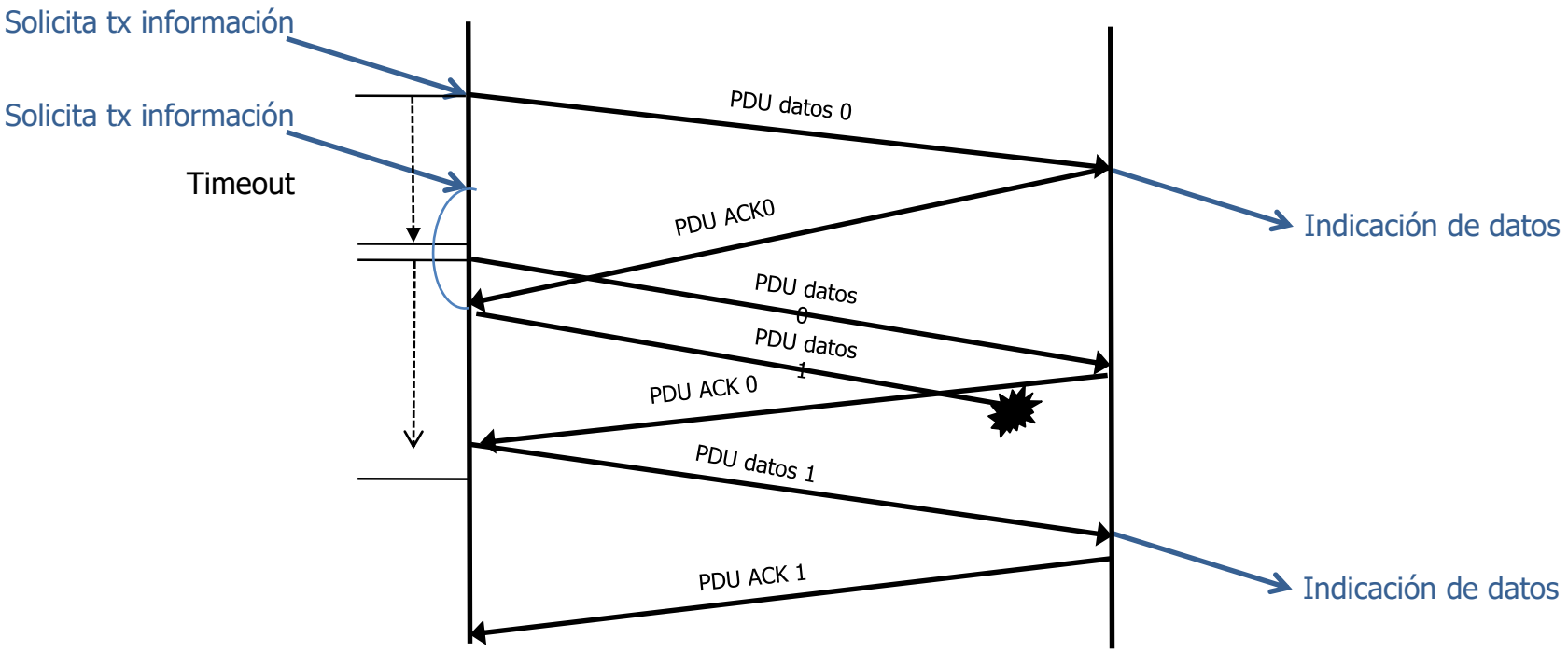
¿Qué pasa si ACK se pierde?

Protocolo send and wait



Temporizador mal ajustado y
Pérdida de tramas inadvertida.
¿Cómo corregir este error?

Protocolo send and wait

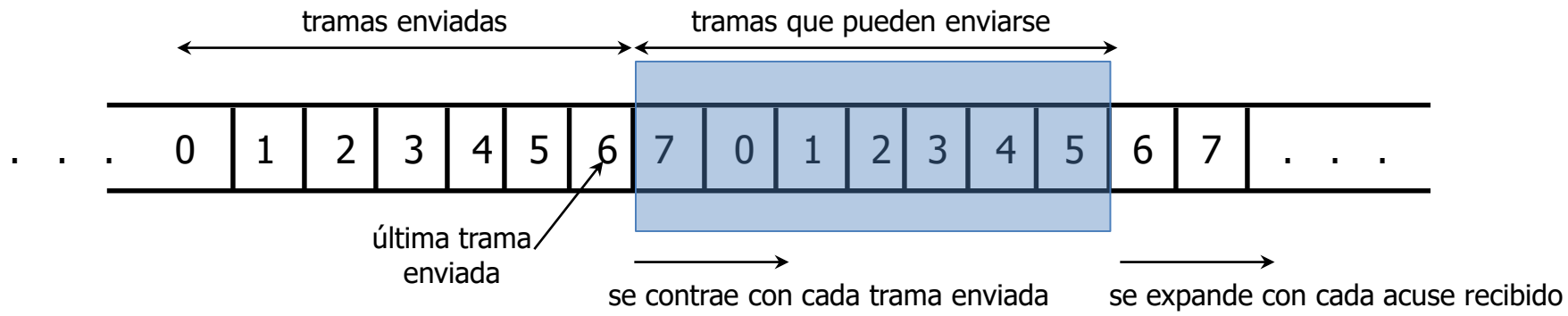


Enumerar PDU y ACK

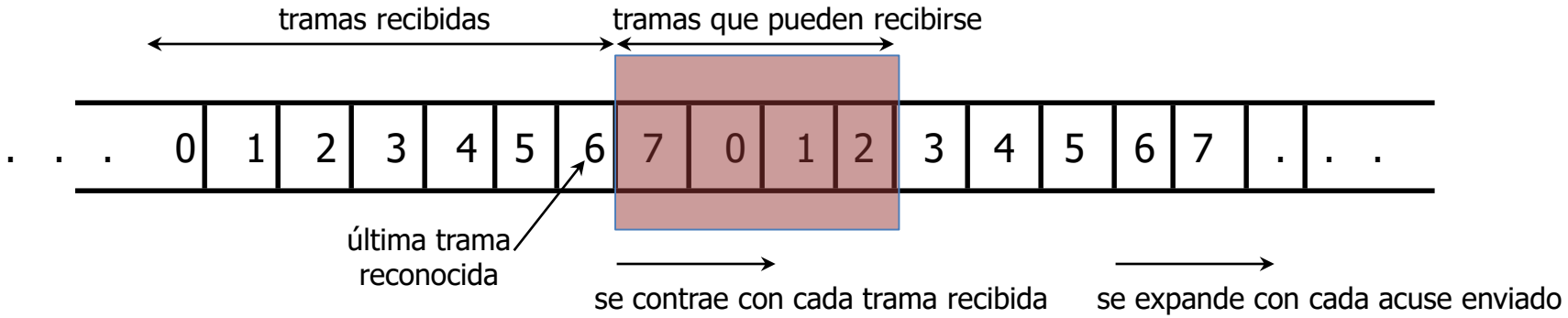
Protocolo de bit alternado

- Familia de protocolos Automatic Response Request (ARQ)
- Muy sencillo, pero sumamente ineficiente.
  - Si el retardo de propagación es grande, el medio queda subutilizado mucho tiempo
- Acuses de recibo negativos parecerían una buena idea, pero son difíciles de implementar (NACK también puede perderse)

Protocolos de ventanas deslizantes

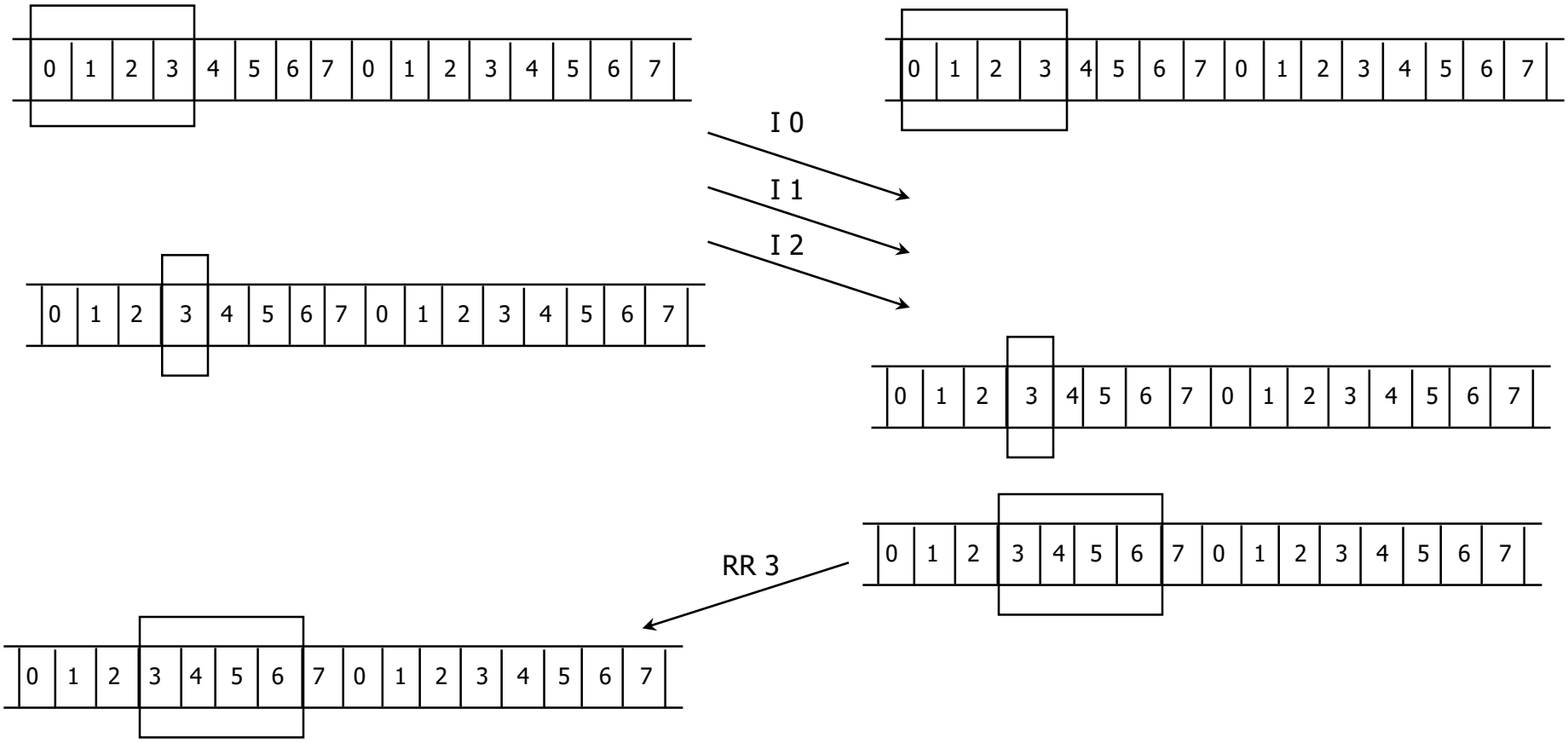


a) Ventana del transmisor

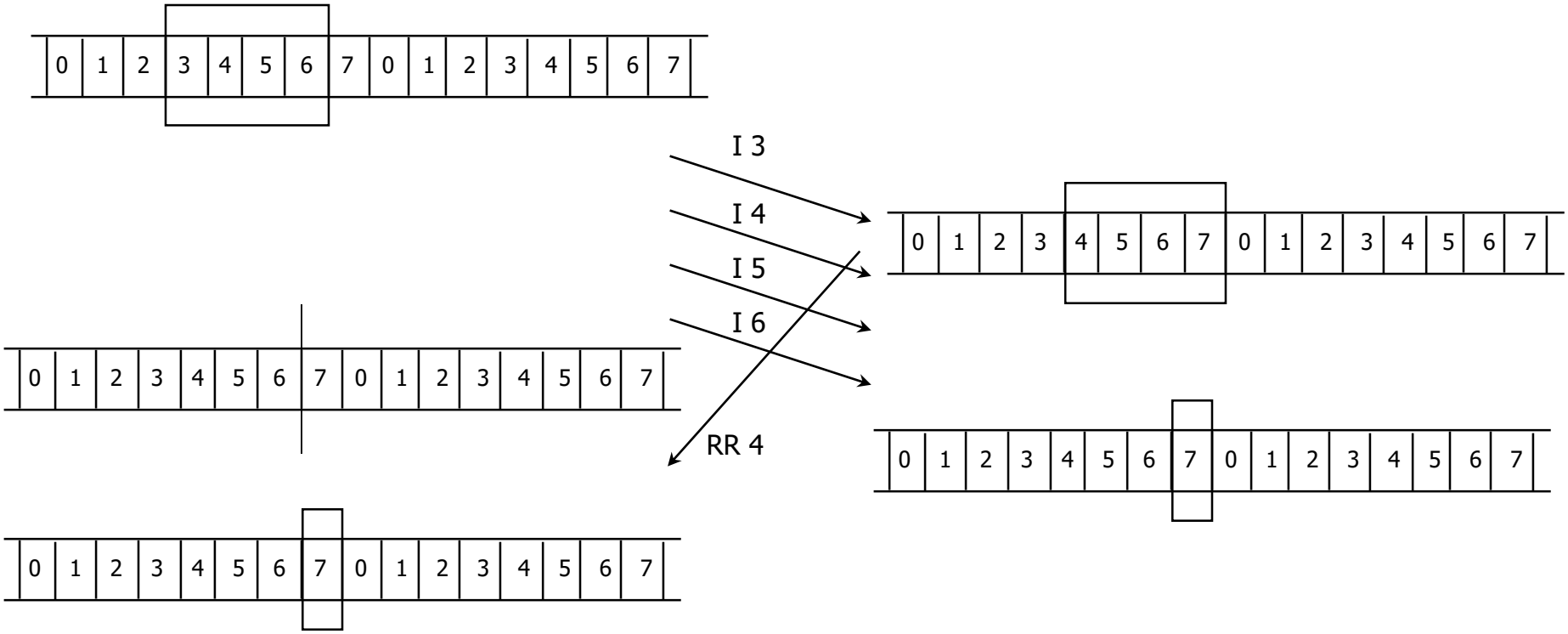


b) Ventana del receptor

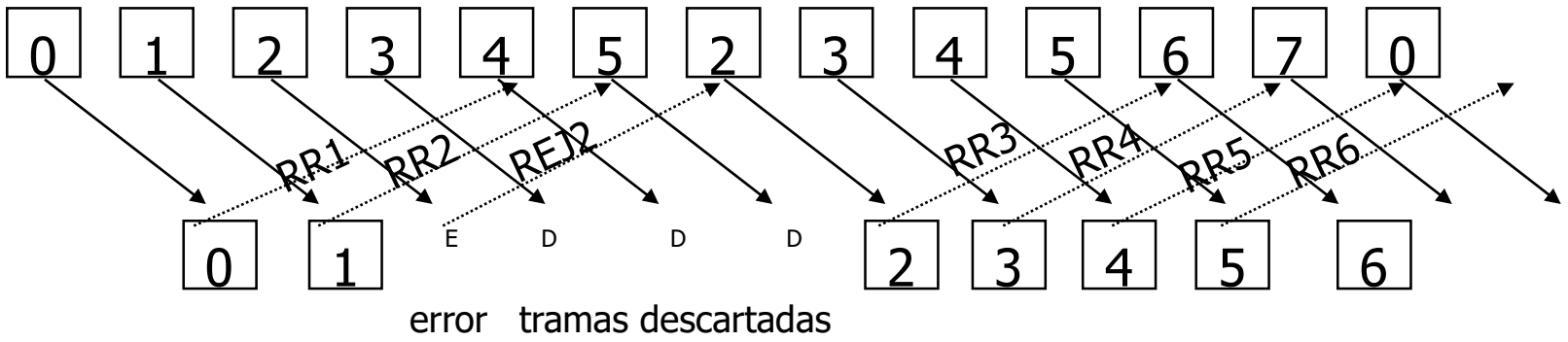
Ventanas deslizantes



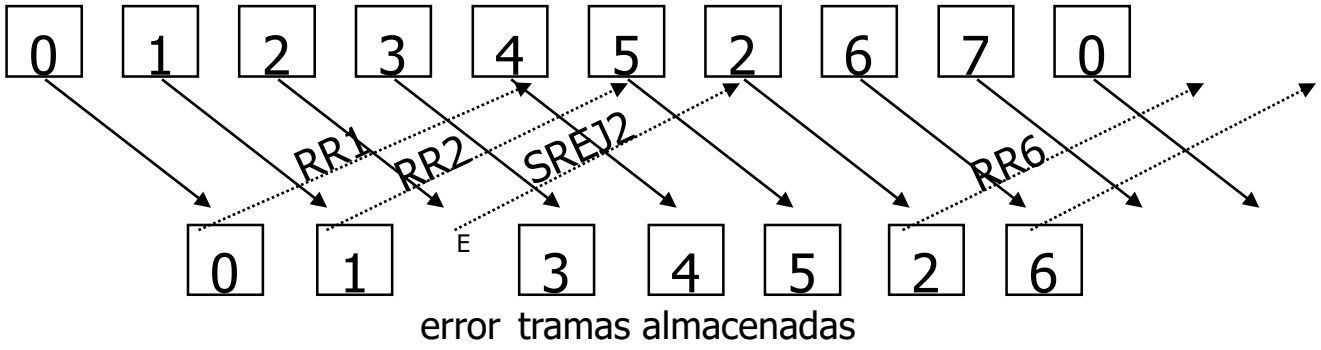
Ventanas deslizantes



Rechazos



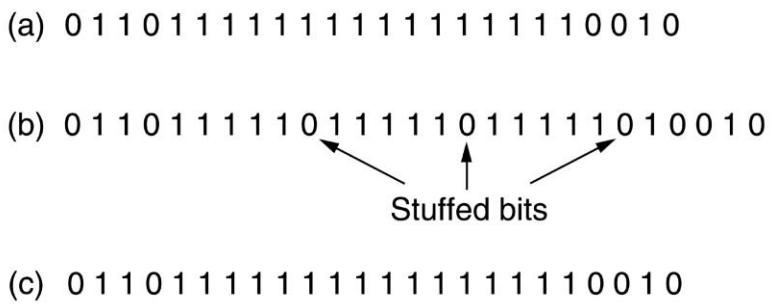
a) Go back N



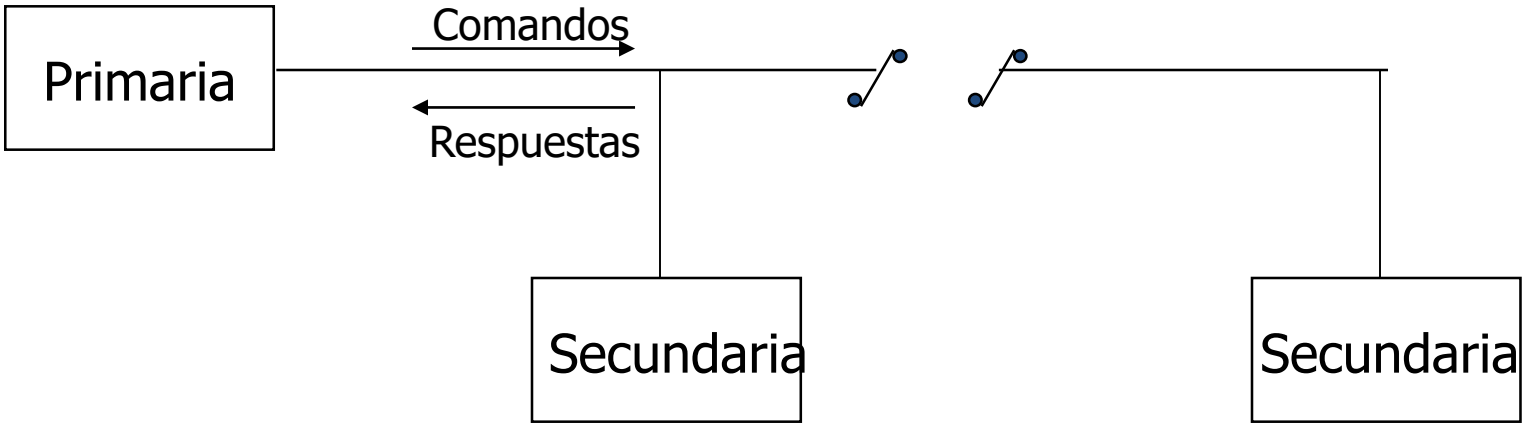
b) Rechazo selectivo

HDLC

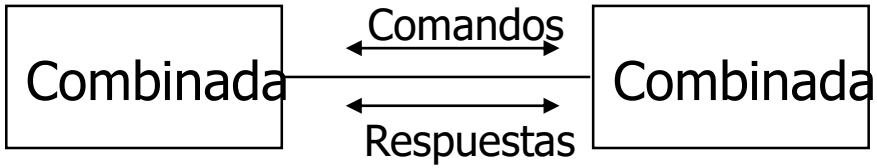
- Protocolo orientado a bits
  - Utiliza un campo de las tramas para implementar funciones de control
  - Transmisión bidireccional simultánea
  - Usa una ventana deslizante
  - En modo transparente
    - inserción de bits



HDLC

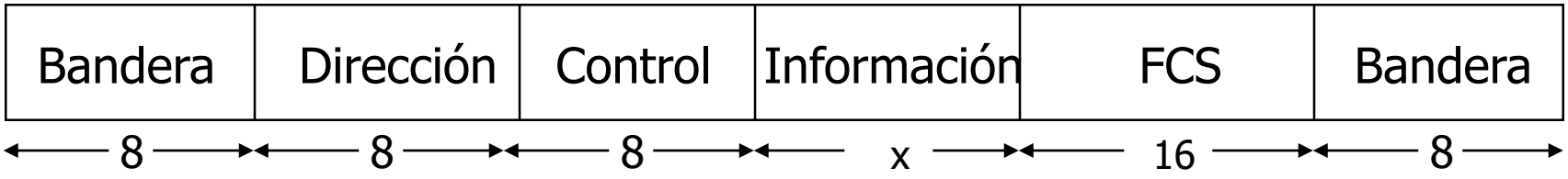


a) Configuación no balanceada



b) Configuración balanceada

HDLC



a) Formato de las tramas

	1	2	3	4	5	6	7	8
I: Información	0	N(S)			P/F	N(R)		
S: Supervisión	1	0	S		P/F	N(R)		
U: No numerada	1	1	M		P/F	M		

b) Formato del campo de control

HDLC

- Subconjuntos o variantes
  - PPP
  - LAPB
  - LAPM
  - LAPD
  - LAPF
  - LLC

PPP

- Usado en Internet
  - computadora - enrutador ISP
  - enrutador - enrutador
- Es similar a HDLC
  - puede ser orientado a caracteres o a bits
  - utiliza la dirección 0xFF
  - envía tramas UI
  - tiene un nuevo campo: Protocolo

PPP

- Protocolo
  - IP 0x0021
  - LCP 0xc021
  - IPCP 0x8021
  - ML PPP 0x003d

PPP

- LCP permite negociar:
  - Tamaño máximo de las tramas
  - Protocolo de autenticación
    - PAP (Protocolo 0xc023)
    - CHAP (Protocolo 0xc223)
  - Compresión del campo Protocolo
  - Supresión de los campos Dirección y Control

PPP

- IPCP permite negociar:
  - Dirección IP de la computadora
  - Compresión de los encabezados TCP/IP
    - Protocolo 0x002d

LAPB

- Comandos y respuestas
  - Información
  - Supervisión
    - RR RNR
    - REJ
  - No numeradas
    - SABM SABME UA DM
    - DISC UA

Compresión

- Reducción de *información*
  - redundante
  - poco perceptible
- Compresión
  - Sin pérdida
  - Con pérdida
    - JPEG
    - MPEG
    - MP3

Compresión

- Codificación (sin pérdida)
  - Run-length
  - Estadística:
    - Huffman
    - Lempel-Ziv

Compresión

- Run-length
  - Envía la longitud de secuencias de símbolos repetidos
  - símbolo símbolo símbolo cuenta

Compresión

- La entropía de un símbolo mide la cantidad de información (en *bits*) que contiene

$$-\log_2 p(i)$$

- La entropía de una fuente es el valor esperado de la cantidad de información de los símbolos que produce

$$H = -\Sigma p(i)\log_2 p(i) \qquad (1 <= i <= M)$$

Compresión

- La entropía de la fuente es el promedio del mínimo número de *bits por símbolo* que se necesitan para representar su información (sin pérdida)
- El promedio de la longitud de los símbolos codificados de un mensaje es mayor o igual a la entropía de la fuente.

Compresión

- AAAABBCD
- Símbolos en el mensaje: M=4
- $\log_2 M = 2$ bits por símbolo
- Entropía H

  $-(4/8)\log_2(4/8)-(2/8)\log_2(2/8)-(1/8)\log_2(1/8)-(1/8)\log_2(1/8)$

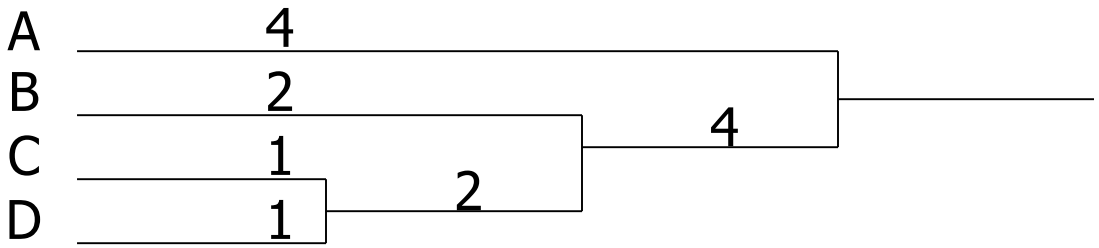
  1.75 bits por símbolo
- Redundancia R
  - $R = \log_2 M - H = 0.25$

Compresión

- Huffman
  - Construye un árbol binario basado en la probabilidad de ocurrencia de cada símbolo
  - Asigna a los símbolos más frecuentes códigos cortos
- Huffman dinámico
  - El árbol se construye dinámicamente y varía con el tiempo

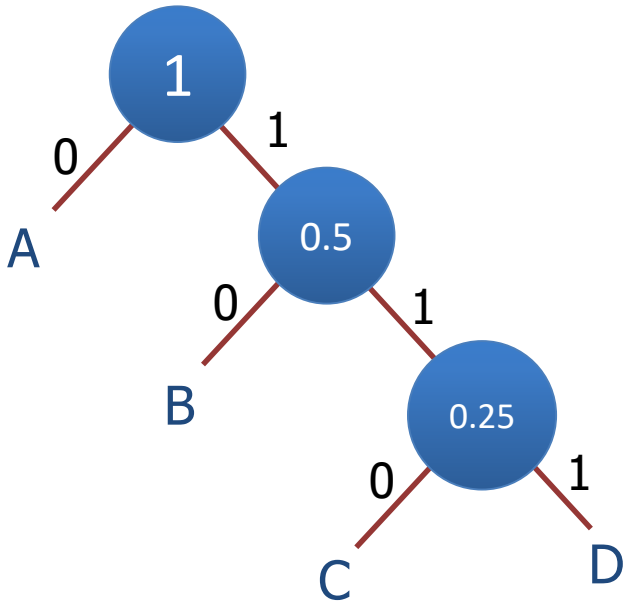
Compresión

- AAAABBCD



- Longitud promedio de los símbolos codificados
  - 1*(4/8) + 2*(2/8) + 3*(1/8) + 3*(1/8)
  - 1.75 bits por símbolo

Compresión Hufmann



Mensaje	A A A B C A A A B A D B	Total
Original	00 00 00 01 10 00 00 00 01 00 11 01	24 bits
Comprimido	0 0 0 10 110 0 0 0 10 0 111 10	19 bit

Compresión

- Lempel-Ziv (LZ77 y 78)
  - Basados en un diccionario (dinámico) de cadenas
  - Envían el lugar, en el diccionario, donde se encuentra una cadena ya vista
- Variaciones de LZ78 se usan en:
  - compress (de UNIX)
  - V.42bis

Compresión LZ78

Pos	1	2	3	4	5	6	7	8	9
Car	A	B	B	C	B	C	A	B	A

Paso	Posición	Diccionario	Salida
1	1	A	(0,A)
2	2	B	(0,B)
3	3	BC	(2,C)
4	5	BCA	(3,A)
5	8	BA	(2,A)

Ejemplo LZW.- Comprimir “itty bitty bit bin”

Input		New dictionary entry		9-bit characters transmitted		New dictionary entry		Output
105	i	–	–	256	(start)	–	–	–
116	t	258	it	105	i	–	–	i
116	t	259	tt	116	t	258	it	t
121	y	260	ty	116	t	259	tt	t
32	space	261	y-space	121	y	260	ty	y
98	b	262	space-b	32	space	261	y-space	space
105	i	263	bi	98	b	262	space-b	b
116	t	–	–	–	–	–	–	–
116	t	264	itt	258	it	263	bi	it
121	y	–	–	–	–	–	–	–
32	space	265	ty-space	260	ty	264	itt	ty
98	b		–	–	–	–	–	–
105	i	266	space-bi	262	space-b	265	ty-space	space-b
116	t	–	–	–		–	–	–
32	space	267	it-space	258	it	266	space-bi	it
98	b	–	–	–		–	–	–
105	i	–	–	–		–	–	–
110	n	268	space-bin	266	space-bi	267	it-space	space-bi
–	–	–	–	110	n	268	space-bin	n
–	–	–	–	257	(stop)	–	–	–

8-bit characters input

Table 3.2: LZW Example 1 Transmission Summary