9.1 Introduction

The Internet is a combination of networks glued together by connecting devices. If a packet is to travel from a host to another host, it needs to pass through these networks. Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

1. Nodes and links
2. Services
3. Two categories of links
4. Two sublayers

Figure 9.1 *Communication at the data-link layer*

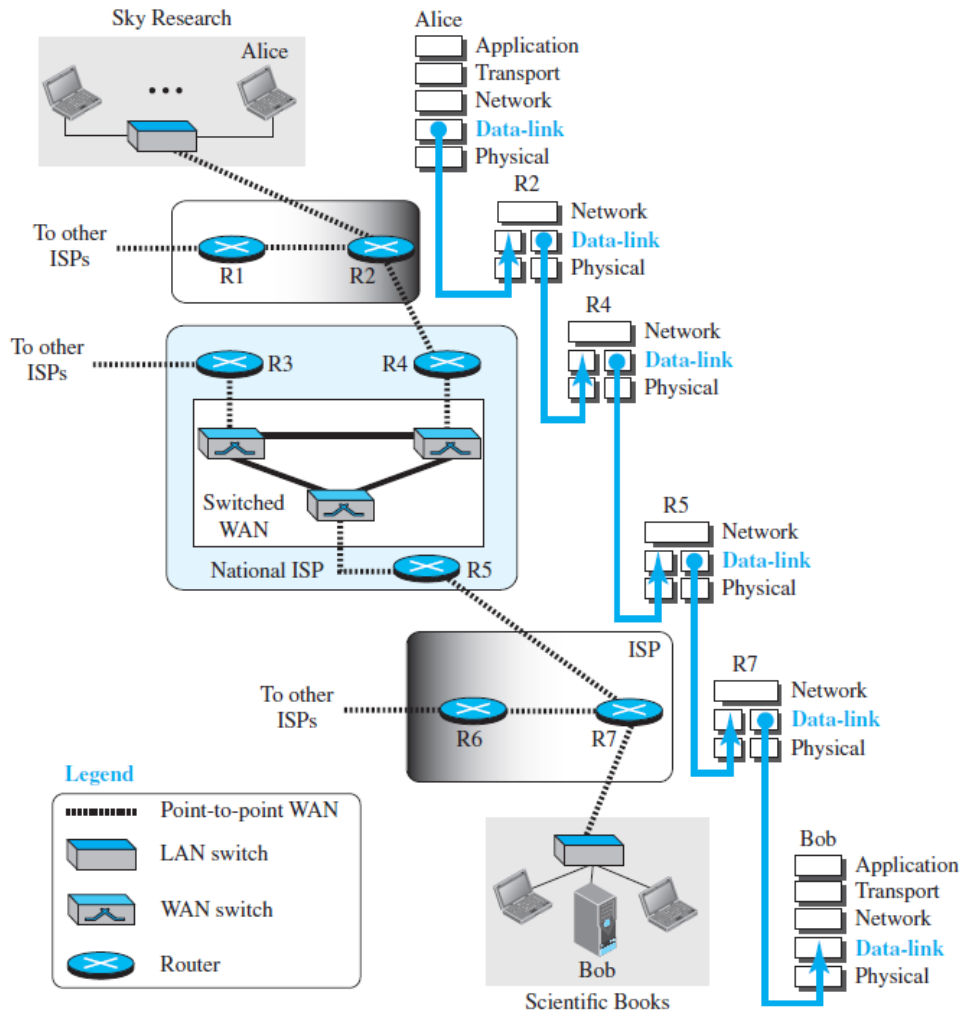
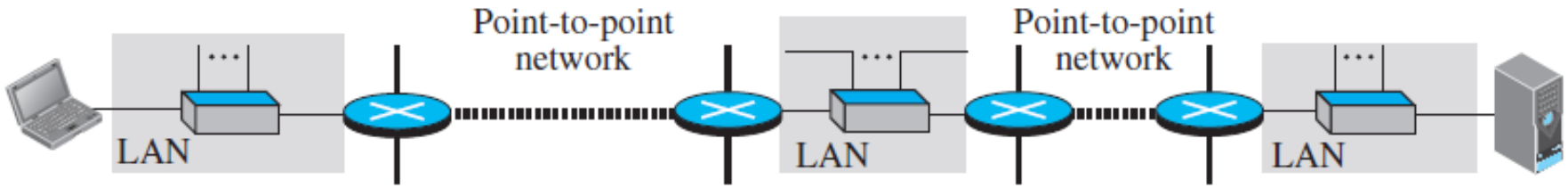
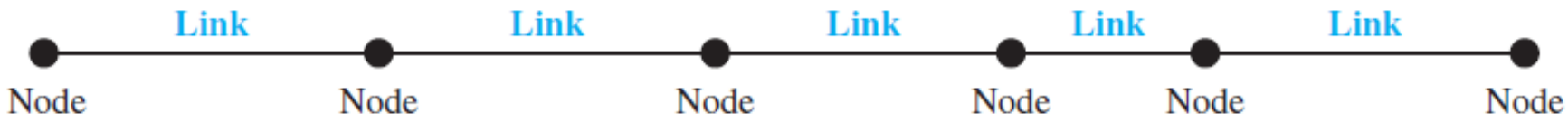


Figure 9.2 *Nodes and Links*



a. A small part of the Internet

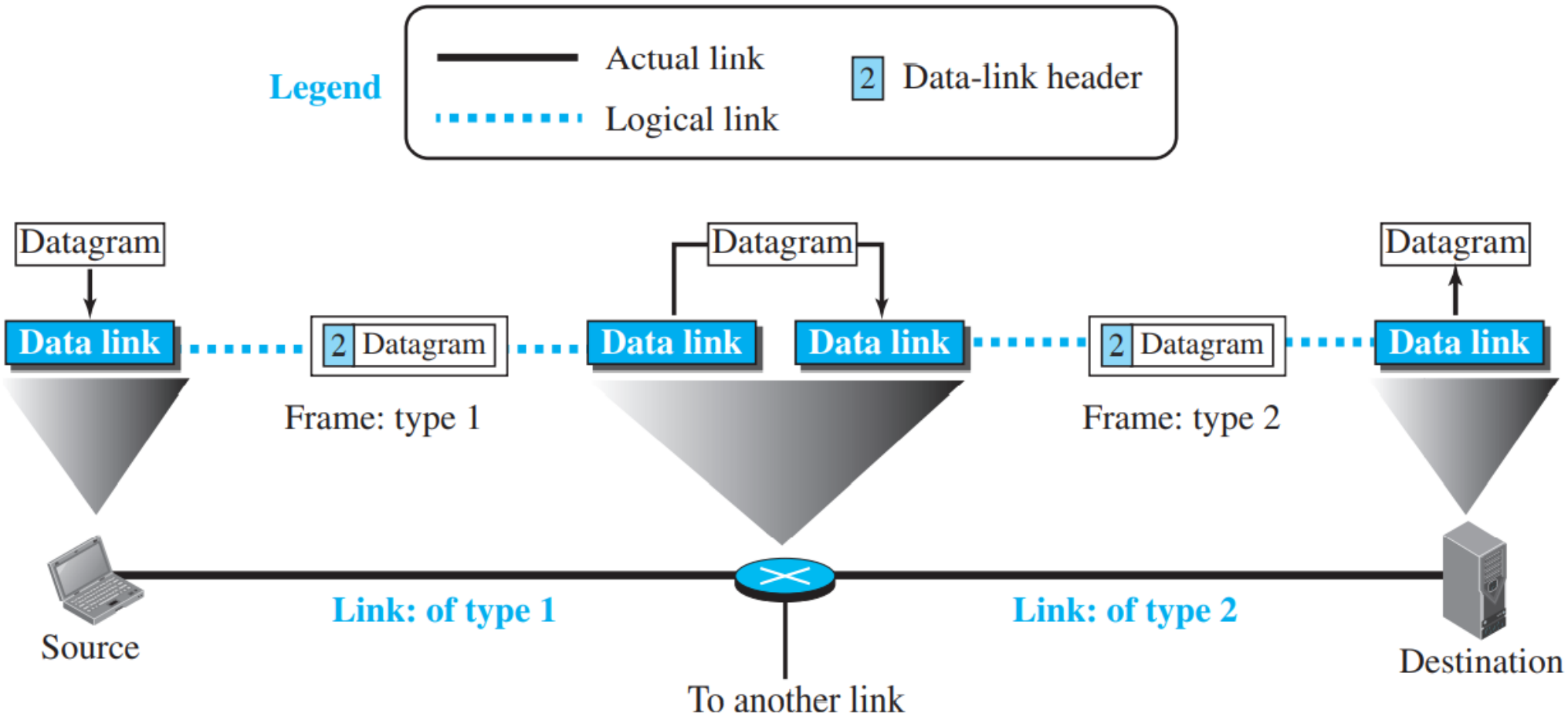


b. Nodes and links

Services

- Encapsulation – sender end
- Decapsulation – receiver end
- Intermediate nodes – do both –
  - As each link may be using a different protocol with a different frame format
  - Even with the same protocol the link-layer addresses are normally different

Figure 9.3 *A communication with only three nodes*



Framing

the first service provided by the data-link layer is framing.

A packet at the data-link layer is normally called a *frame*.

Flow Control

Whenever we have a producer and a consumer, we need to think about flow control.

Rate of **produced frames if > rate of consumed frames**, **buffer** is required at receiving end
Buffer size – limited
Receiver can either **drop frames or inform to stop or slow down**.
Different data-link-layer protocols use different strategies for flow control.

Error Control

a frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media.
Since electromagnetic signals are susceptible to error, a frame is susceptible to error.
The error needs first to be detected. After detection, it needs to be either corrected at the receiver node or discarded and retransmitted by the sending node.

Congestion Control

A link may be congested with frames, which may result in frame loss, most data-link-layer protocols do not directly use a congestion control to alleviate congestion.
Congestion control is considered an issue in the network layer or the transport layer because of its end-to-end nature.

Two categories of link

Data-link layer controls how the medium is used.

Two categories:

In a point-to-point link,

uses the **whole capacity** of the medium

the link is **dedicated** to the two devices;

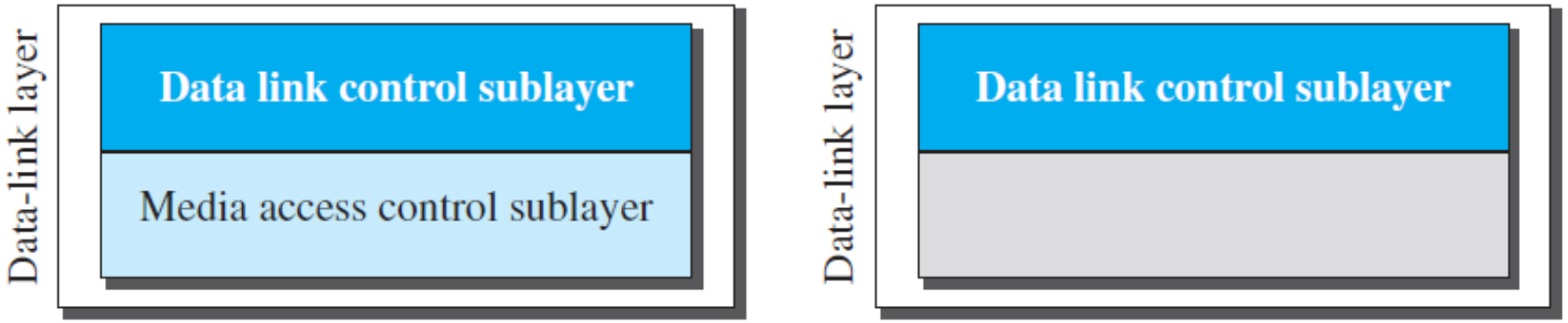
Ex. Communication through landline phones

In a broadcast link,

the link is **shared between several pairs** of devices

Ex. Communication through **cell phones**

Figure 9.4 *Dividing the data-link layer into two sublayers*



a. Data-link layer of a broadcast link

b. Data-link layer of a point-to-point link

Based on the functionalities of DLL, two sublayers are seen –

data link control (DLC) and media access control (MAC).

DLC- deals with all issues common to both **point-to-point and broadcast links**;

MAC - deals only with issues **specific to broadcast links**

9.2 LINK-LAYER ADDRESSING

A *link-layer address* - *link address / physical address /* a *MAC address*.
This is required especially in a connectionless networks.
The source and destination **IP addresses define** the **two ends** but cannot define which **links the datagram** should pass through
IP addresses cannot be changed
Hence DLL Addressing is required
Figure explanation

Ip address for router is needed for the communications in these protocols are between routers.
-more than one IP address in a router, one for each interface?
A router with n interfaces is connected to the Internet at n points.
-destination IP addresses in a packet determined?
the application layer uses the services of DNS to find the destination address of the packet and passes it to the network layer to be inserted in the packet.

9.2 LINK-LAYER ADDRESSING

size of link-layer addresses?

depends on the protocol used by the link. Although we have only one IP protocol for the whole Internet, we may be using different data-link protocols in different links

Three Types of addresses

Unicast

one to one communication

is destined only for one entity in the link.

Ex. A3:34:45:11:92:F1 (12 hexa decimal digits)

Multicast

one-to-many communication

Ex: A2:34:45:11:92:F1 - 2nd digit should be even

Broadcast Address

Broadcasting means one-to-all communication.

FF:FF:FF:FF:FF:FF

1. Three types of addresses
2. Address Resolution protocol
3. An example of communication

Broadcast

Address Resolution Protocol (ARP)

Figure 9.6 *Position of ARP in TCP/IP protocol suite*

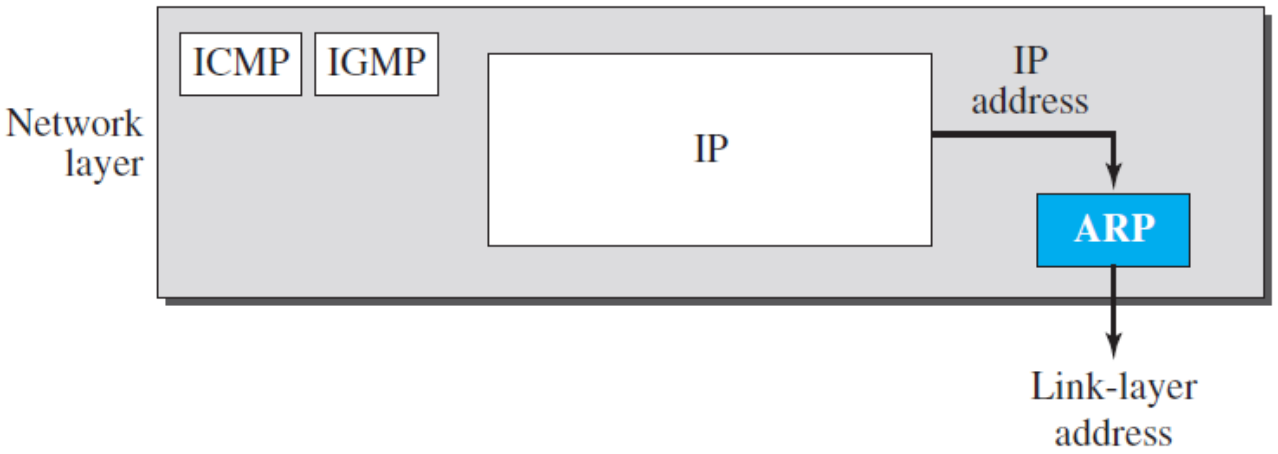


Figure 2.20 *IP addresses*

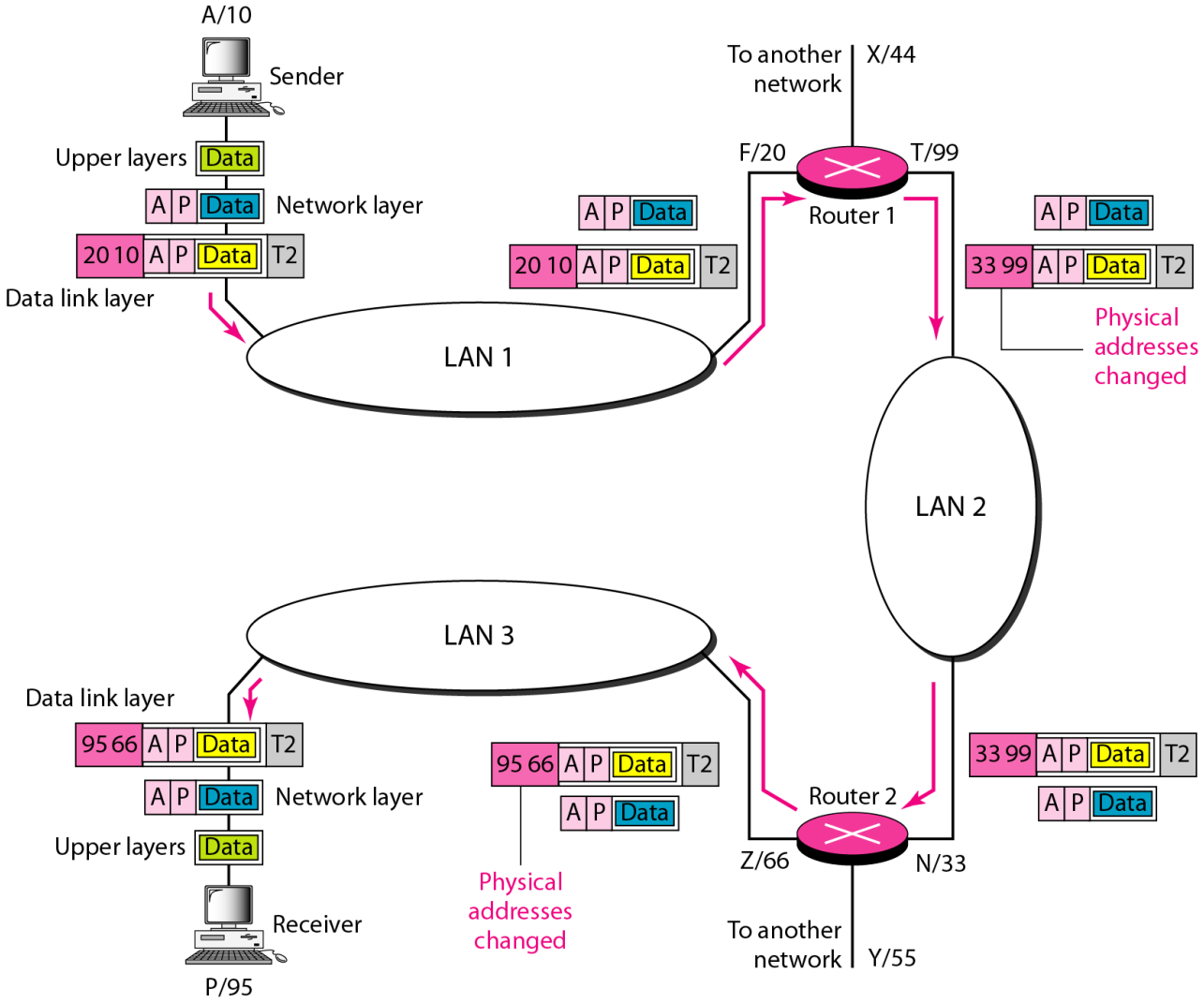


Figure 9.5 *IP addresses and link-layer addresses in a small internet*

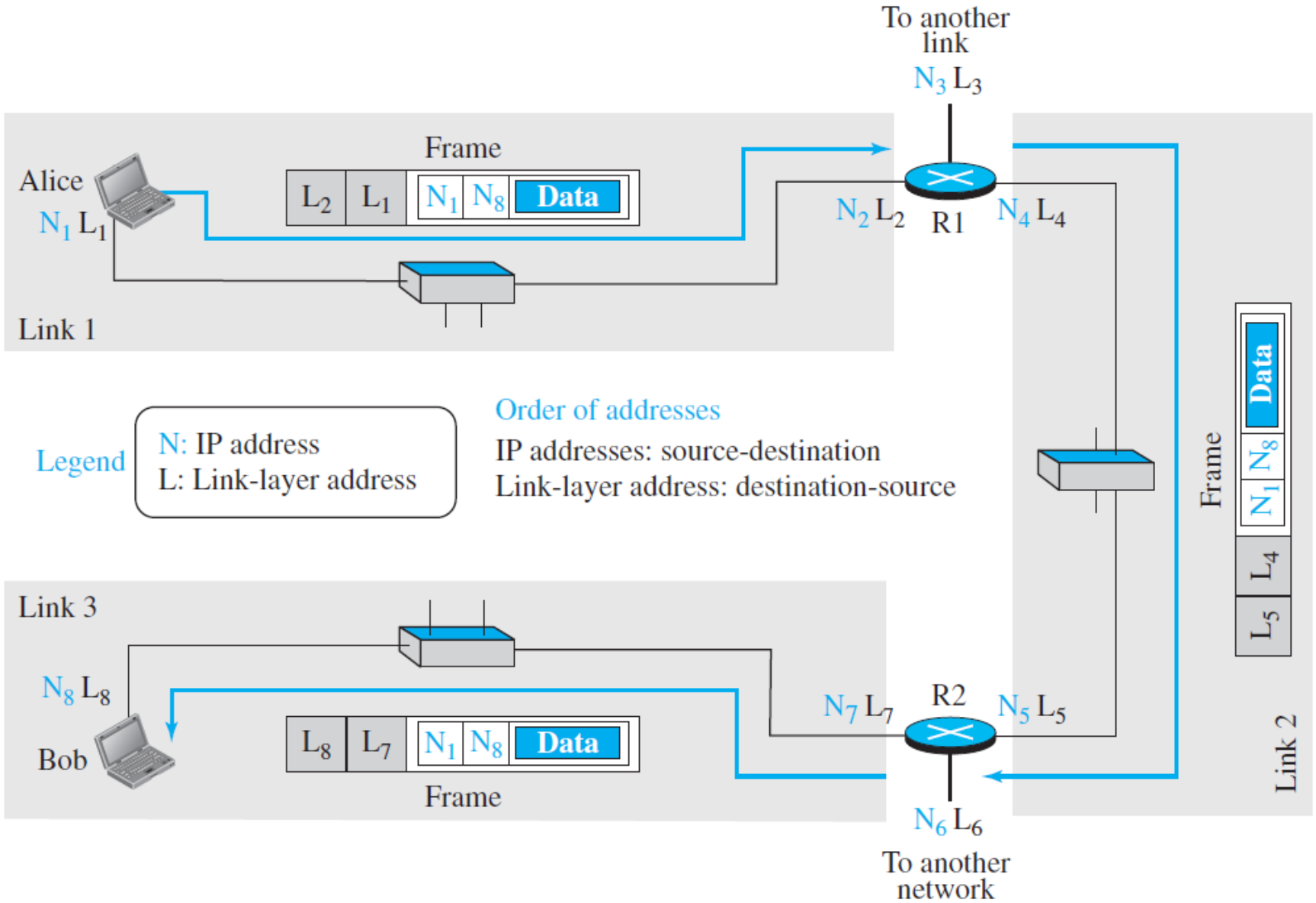
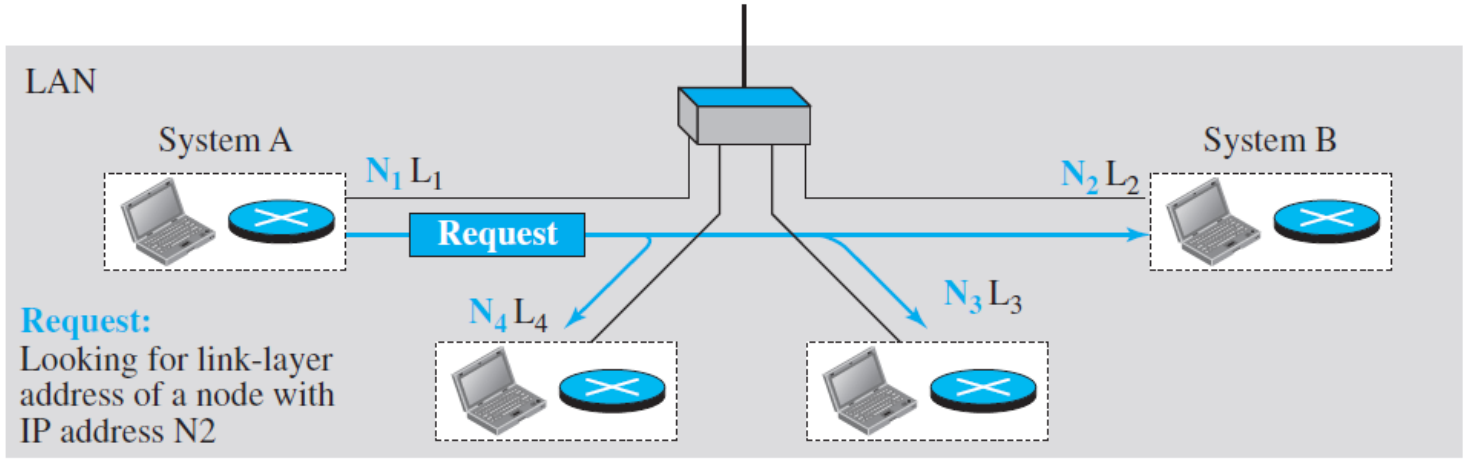
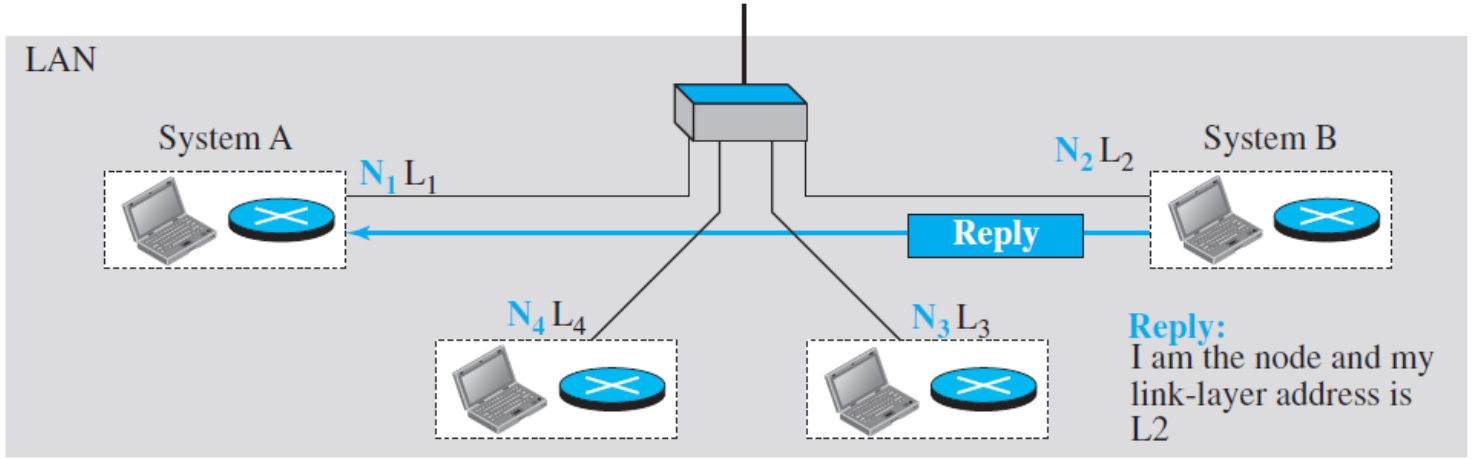


Figure 9.7 *ARP operation*



a. ARP request is broadcast



b. ARP reply is unicast

Caching

Why bx an ARP why not the complete datagram?

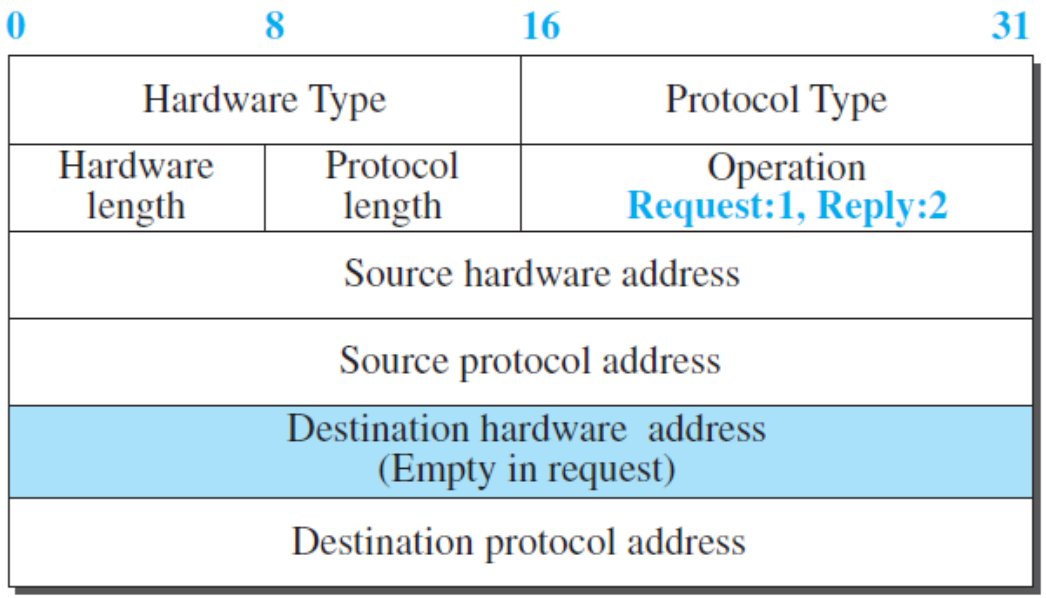
Improve efficiency

only the this request disturbs all nodes

Considering 10 datagrams – 10 packets would have disturbed all the nodes

Once received, the link layer address can be cached and used for further communications

Figure 9.8 *ARP packet*



Hardware: LAN or WAN protocol
Protocol: Network-layer protocol

IPv4 protocol is (0800)

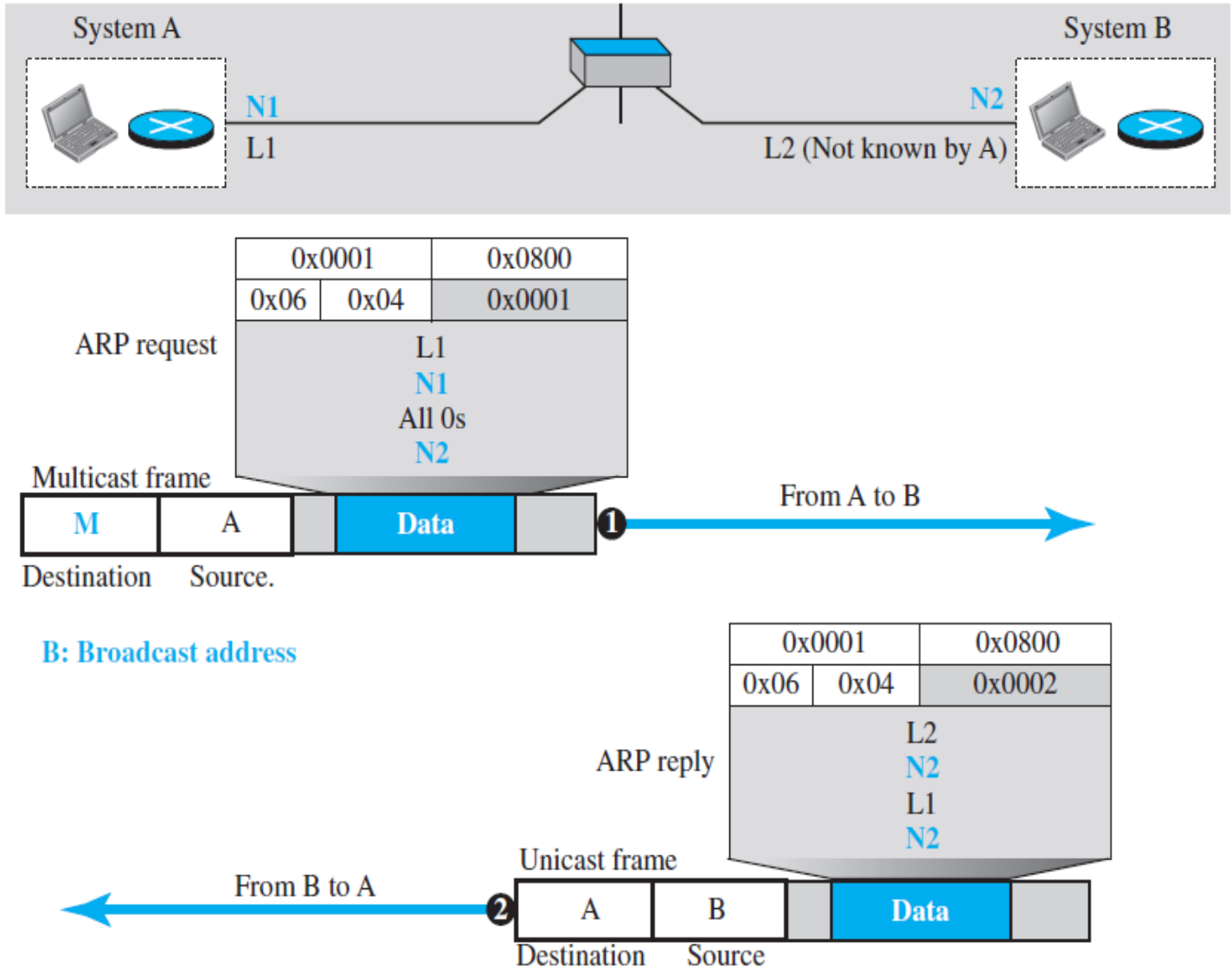
Hardware addr – MAC

Protocol addr - IP

Example 9.4

A host with IP address **N1** and MAC address **L1** has a packet to send to another host with IP address **N2** and physical address **L2** (which is unknown to the first host). The two hosts are on the same network.

Figure 9.9 *Example 9.4*



An Example of Communication

To show how communication is done at the data-link layer and how link-layer addresses are found, let us go through a simple example. Assume Alice needs to send a datagram to Bob, who is three nodes away in the Internet.

Figure 9.10 *The internet for our example*

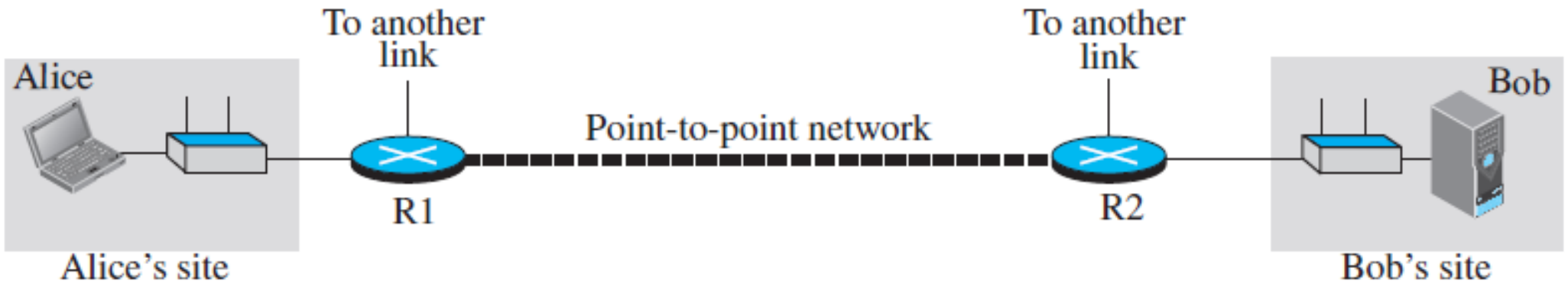


Figure 9.11 *Flow of packets at Alice's computer*

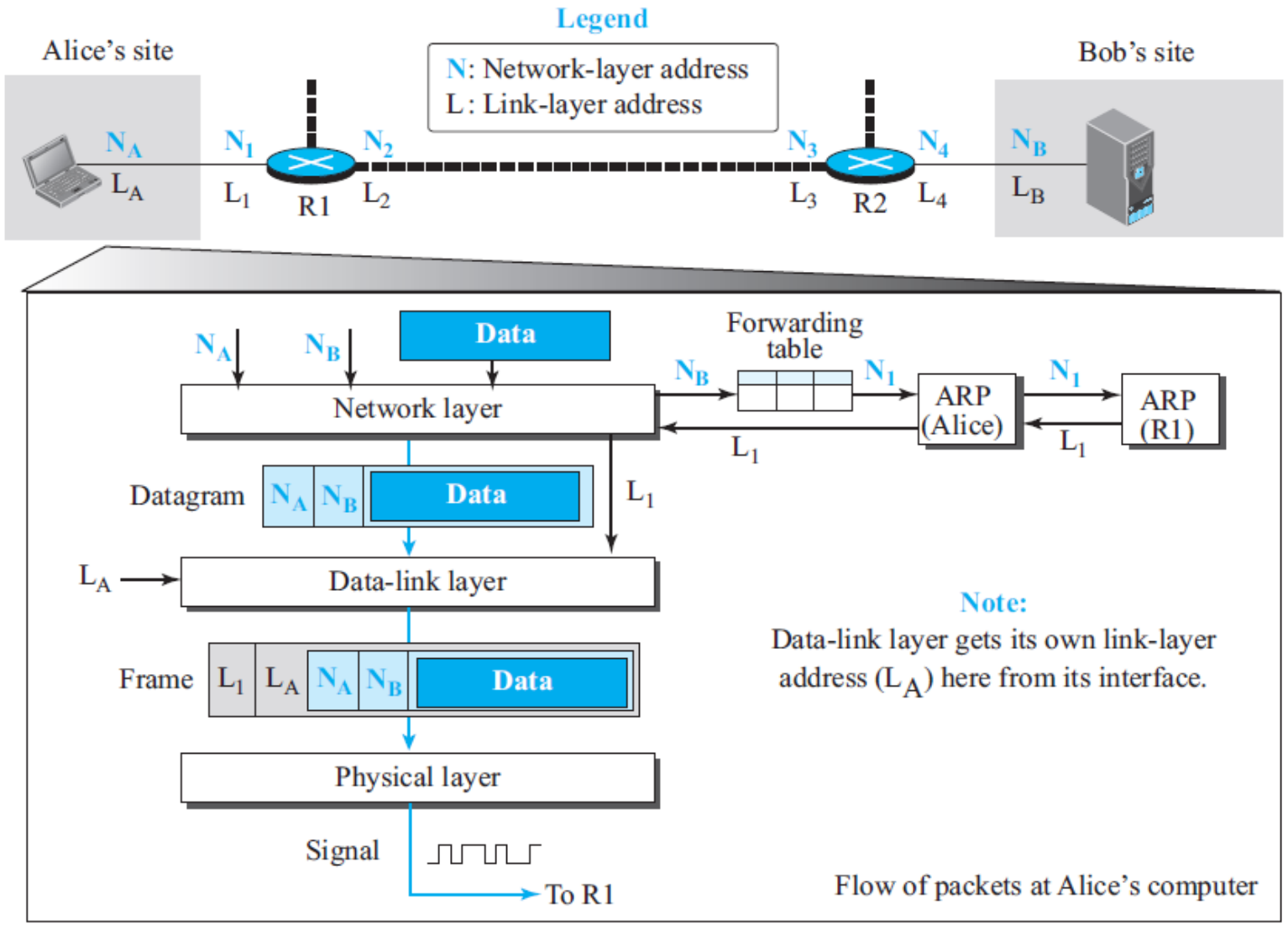


Figure 9.12 *Flow of activities at router R1*

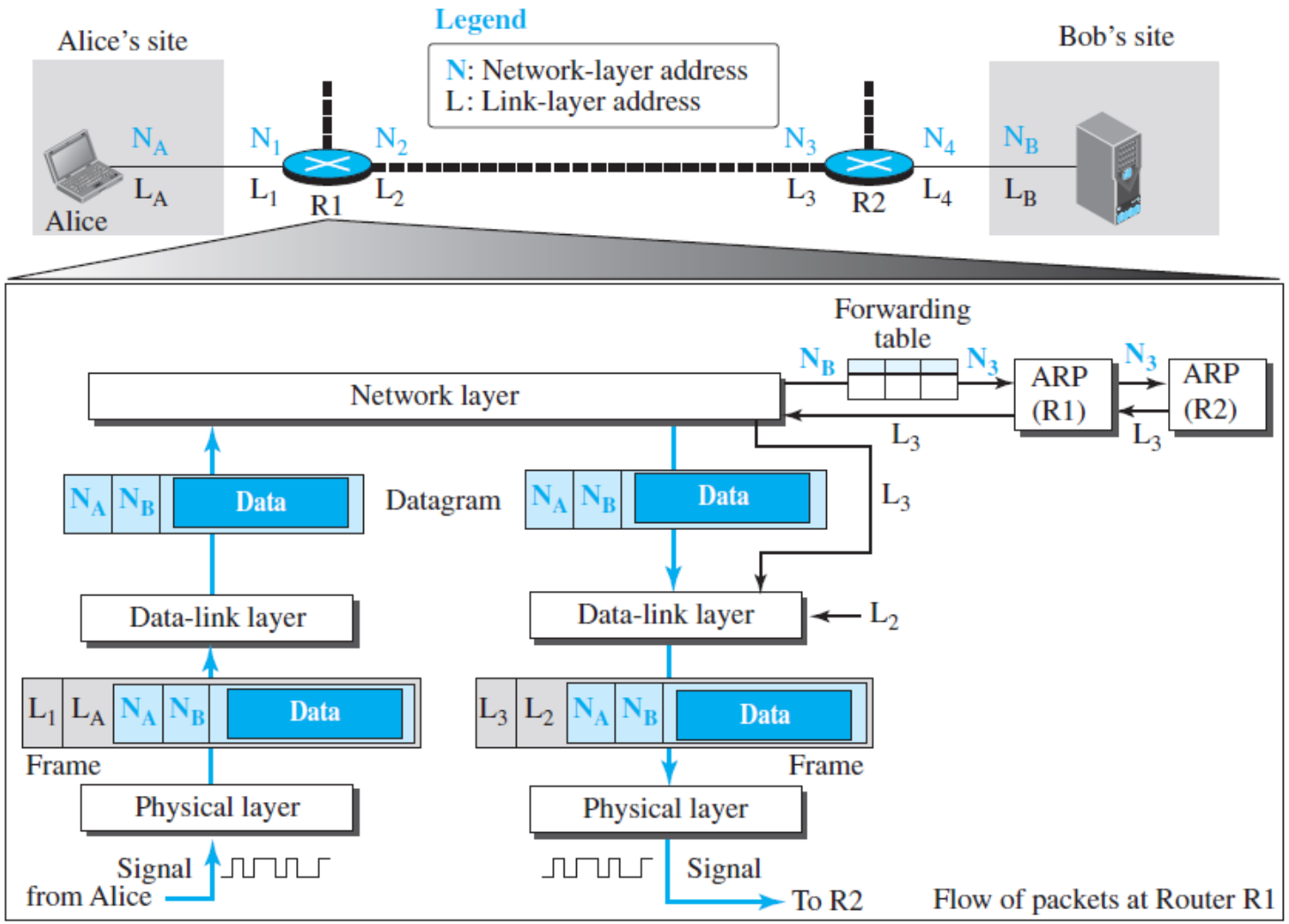


Figure 9.13 *Activities at router R2.*

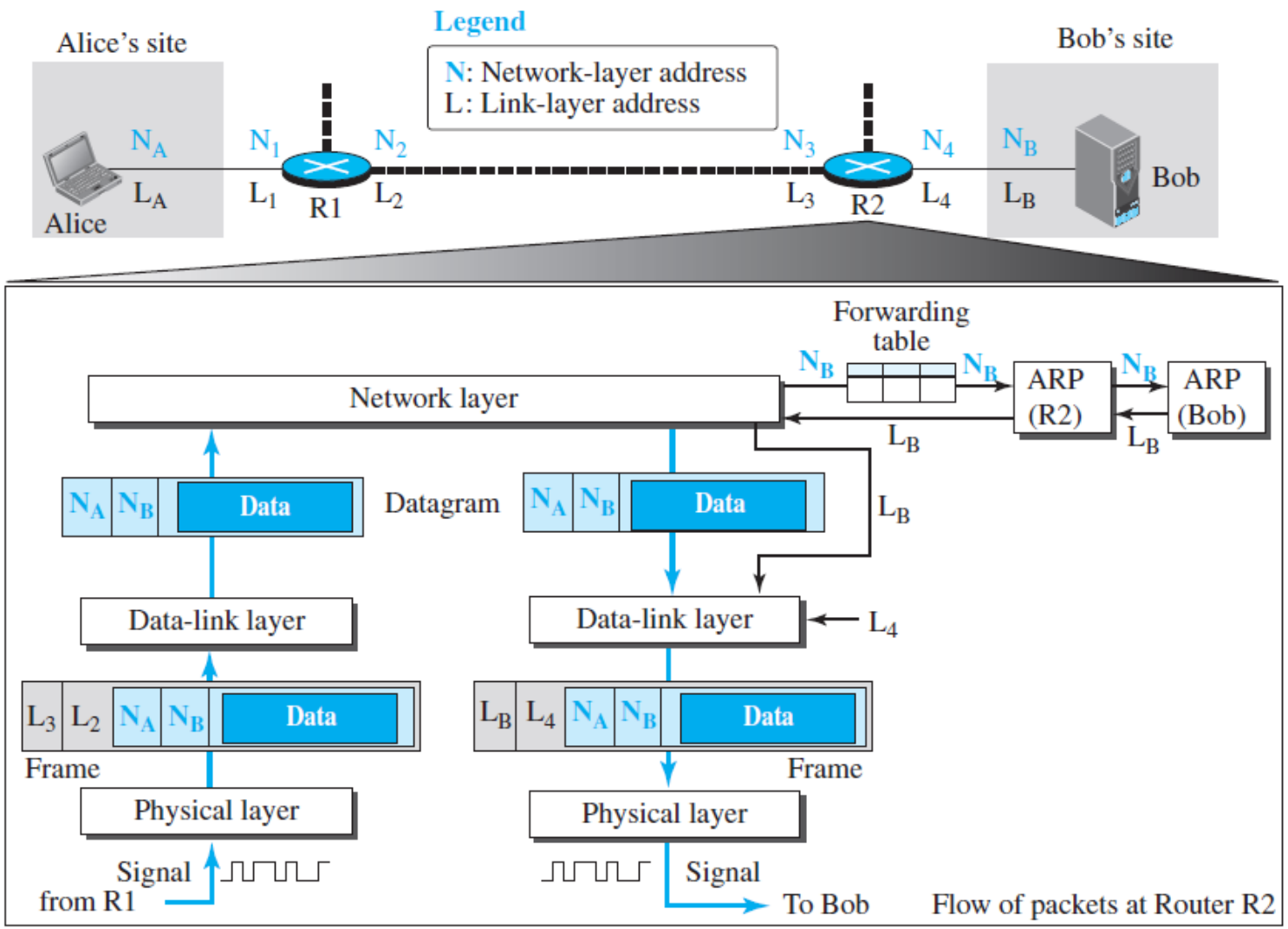
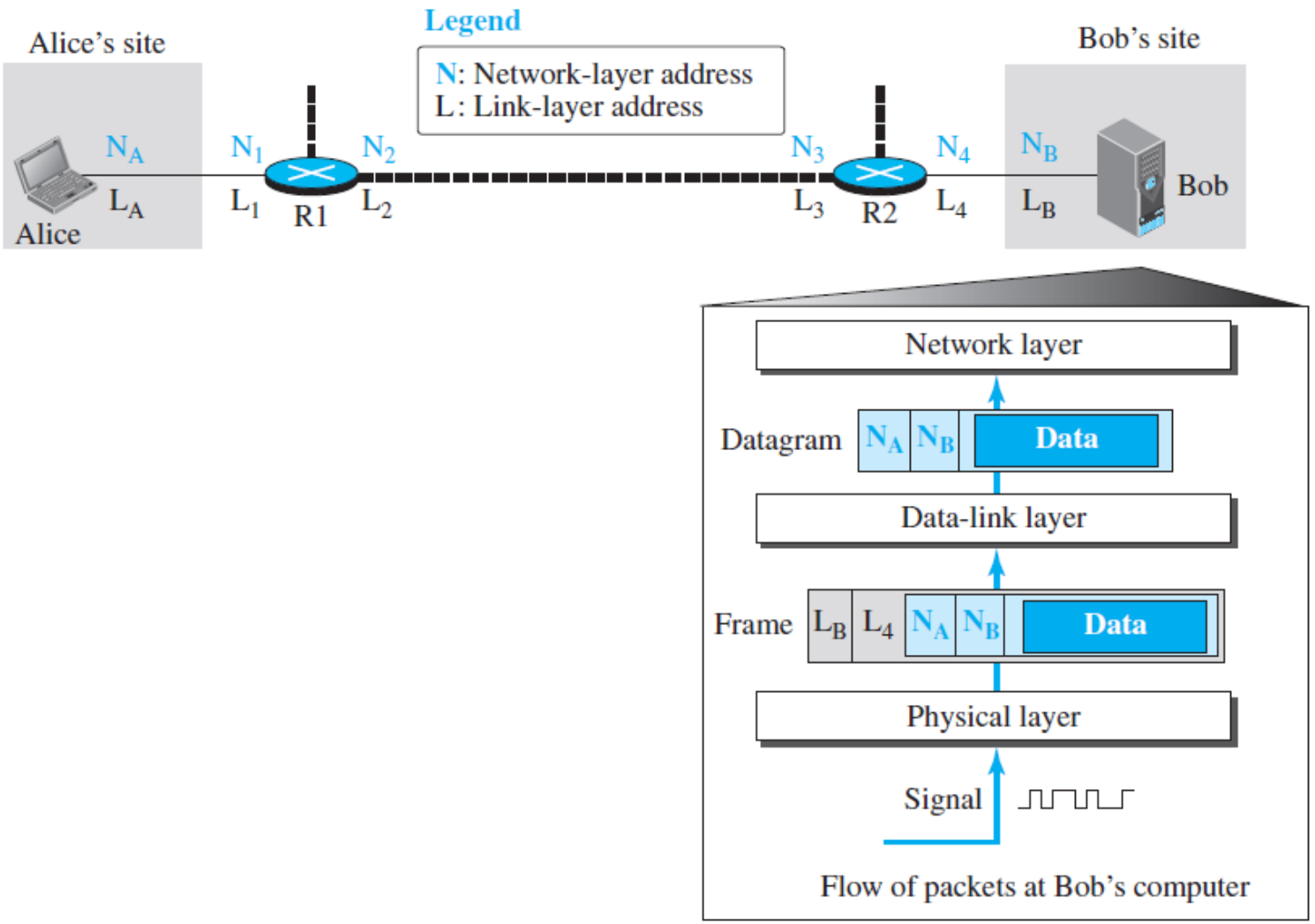


Figure 9.14 *Activities at Bob's site*



Chapter 10

Error Detection and Correction

Note

Data can be corrupted during transmission.

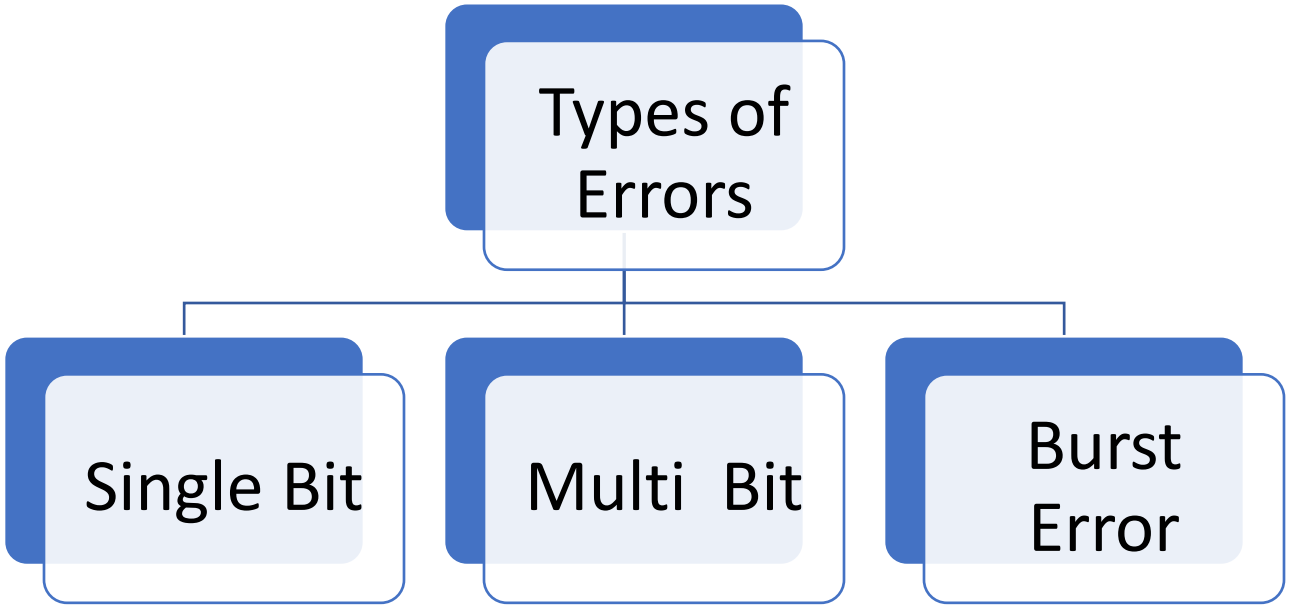
Some applications require that errors be detected and corrected.

10-1 INTRODUCTION

Let us first discuss some issues related, directly or indirectly, to error detection and correction.

Topics discussed in this section:

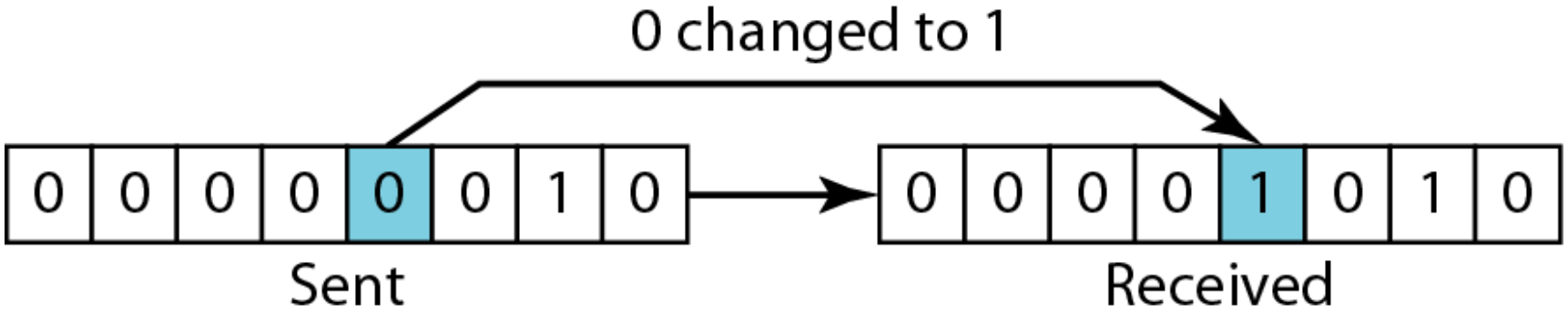
Types of Errors
Redundancy
Detection Versus Correction
Forward Error Correction Versus Retransmission
Coding
Modular Arithmetic


Types of Errors
Single Bit
Multi Bit
Burst Error

Note

In a single-bit error, only 1 bit in the data unit has changed.

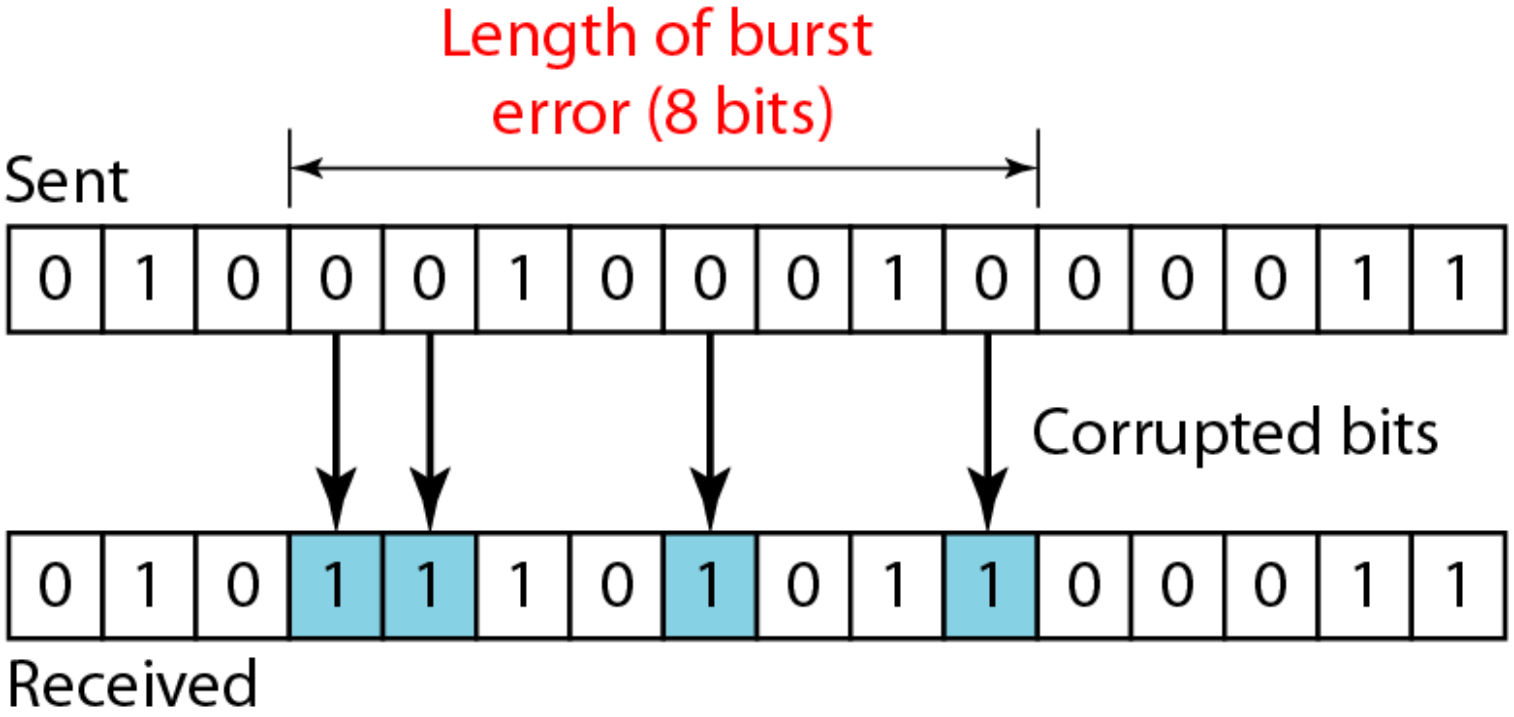
Figure 10.1 *Single-bit error*



Note

A burst error means that 2 or more bits in the data unit have changed.

Figure 10.2 *Burst error of length 8*



Note

To detect or correct errors, we need to send extra (redundant) bits with data. This is called Redundant Bit.

Detection versus Correction

- The correction of errors is more difficult than the detection.
- In detection, only check is made if error is present? Answer is simple, yes or no.
- In correction, once the check is made and error is detected, it also locates the error. Answer is additional information of location of error to perform correction.

Coding

- Redundancy is achieved through various coding schemes.
- We can divide coding schemes into two broad categories: **block coding** and **convolution coding**.
- Block coding.
  - Datawords: Message is divided into blocks of k bits each
  - Codewords: r redundant bits are added to each block to make the length n = k + r. (**n > k)**
  - **$2^k < 2^n$**
  - The block coding process is one-to-one
  - 2^n – 2^k codewords that are not used are invalid or illegal
- If the receiver receives an invalid codeword, this indicates that the data was corrupted during transmission.
- To perform coding, we need encoding and decoding which is shown in next figure.

Error Detection

- Error Detection
- The receiver can detect error/ a change in the original codeword, if it follows these two conditions:
  1. The receiver has (or can find) a list of valid codewords.
  2. The original codeword has changed to an invalid one

Figure 10.3 *The structure of encoder and decoder*

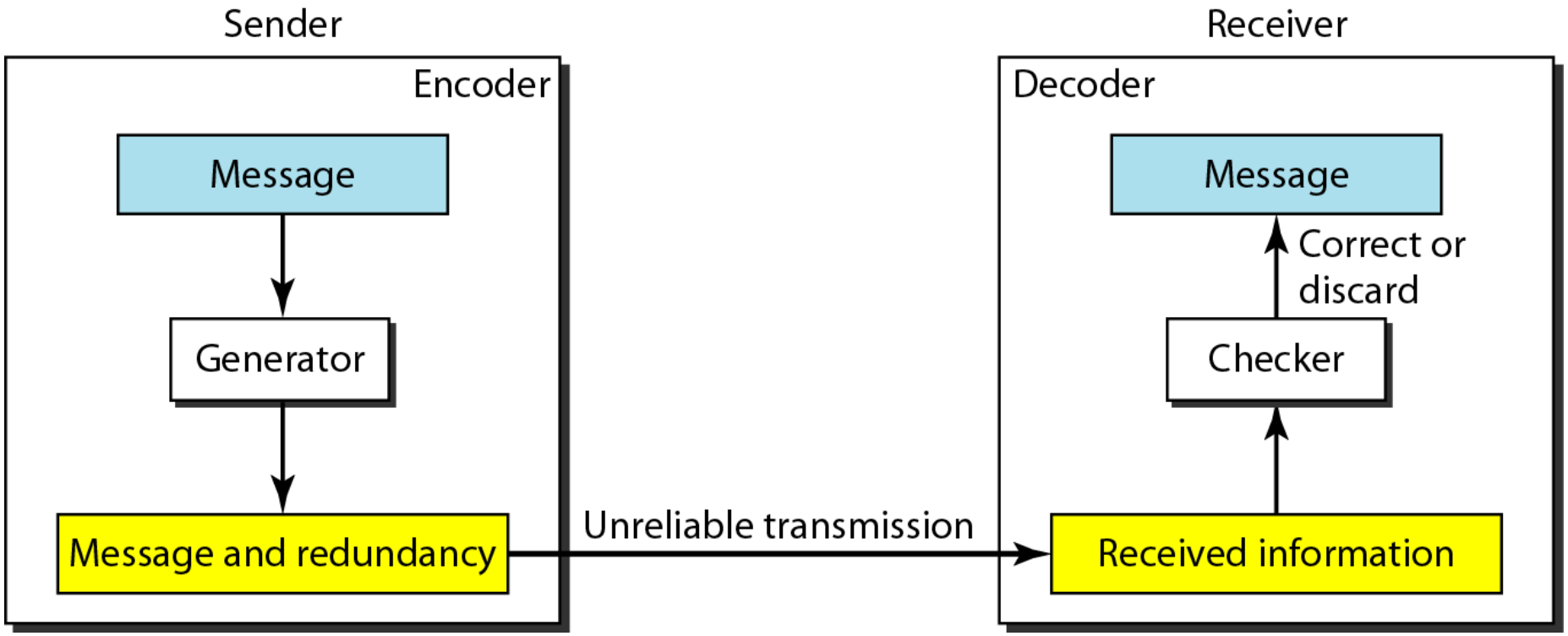
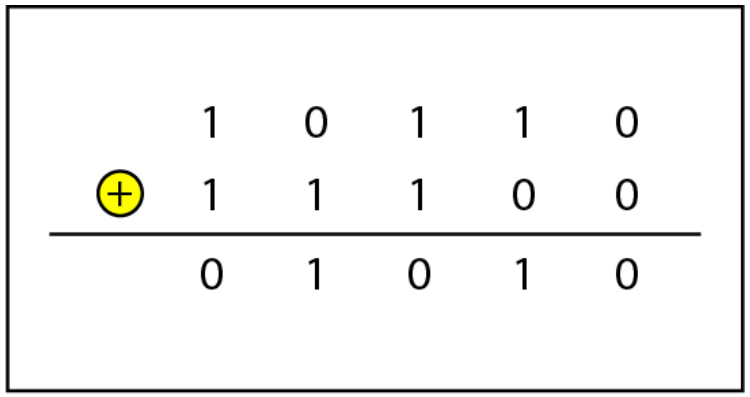


Figure 10.4 *XORing of two single bits or two words*

a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.



c. Result of XORing two patterns

Table 10.1 *A code for error detection in Example 10.1*

Dataword	*Codeword*	*Dataword*	*Codeword*
00	000	10	101
01	011	11	110

If 01 is received as 011 – valid codeword

010 - invalid codeword

110 – valid codeword (incorrect / erroneous)

Note

The Hamming distance d(x , y) between two words x and y is the number of differences between corresponding bits.

10.

Example 10.4

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance d(000, 011) is 2 because

$$000 \oplus 011 \text{ is } 011 \text{ (two 1s)}$$

2. The Hamming distance d(10101, 11110) is 3 because

$$10101 \oplus 11110 \text{ is } 01011 \text{ (three 1s)}$$

Note

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Example 10.5

Find the minimum Hamming distance of the coding scheme in Table 10.1.

Solution

We first find all Hamming distances.

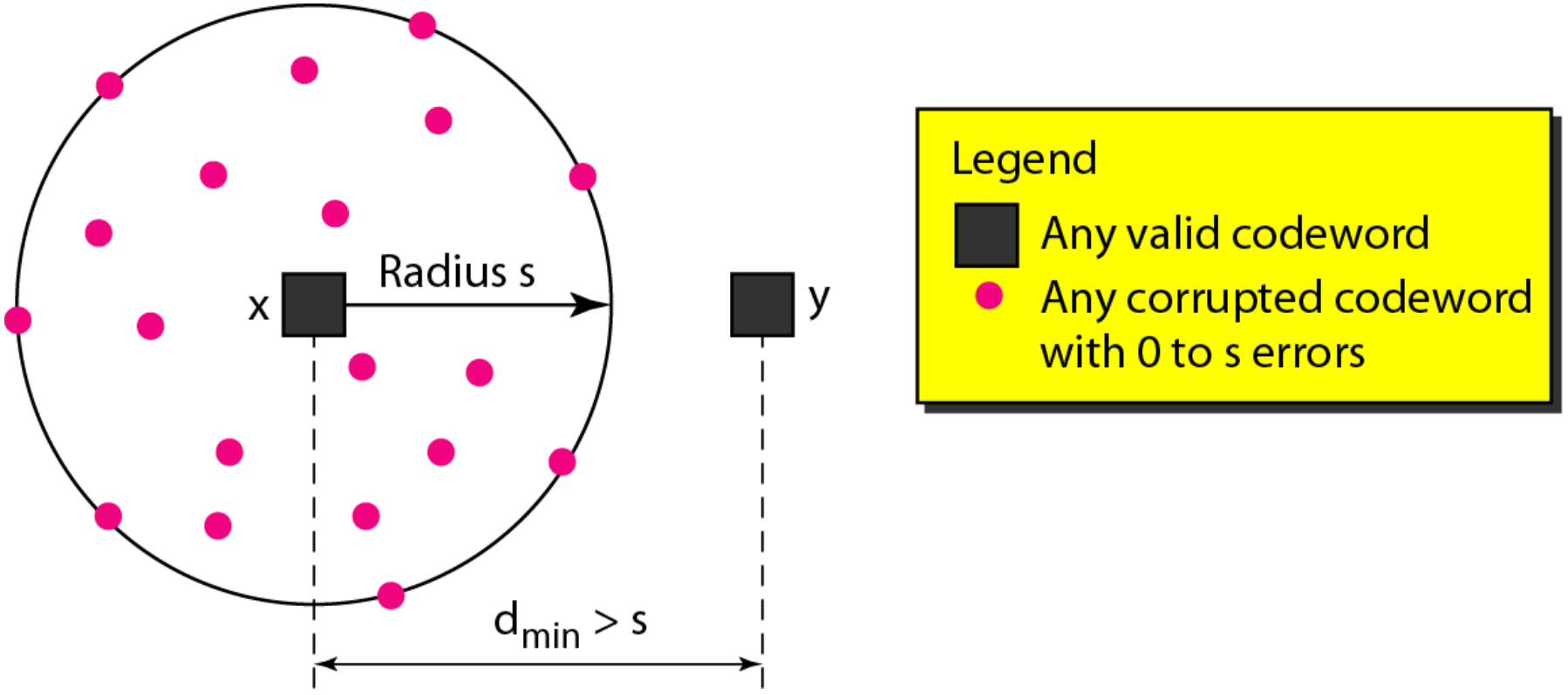
$d(000, 011) = 2$ $d(000, 101) = 2$ $d(000, 110) = 2$ $d(011, 101) = 2$
$d(011, 110) = 2$ $d(101, 110) = 2$

The d_{min} in this case is 2.

Example 10.7

The minimum Hamming distance for our first code scheme (Table 10.1) is 2. This code guarantees detection of only a single error. For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

Figure 10.8 *Geometric concept for finding d_{min} in error detection*



Note

To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

10-3 LINEAR BLOCK CODES

Almost all block codes used today belong to a subset called linear block codes. A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

Topics discussed in this section:

Minimum Distance for Linear Block Codes
Some Linear Block Codes

Note

In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

Linear block code

- Code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

Table 10.1 *A code for error detection in Example 10.1*

Dataword	*Codeword*	*Dataword*	*Codeword*
00	000	10	101
01	011	11	110

Minimum Distance for Linear Block Codes

The minimum Hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

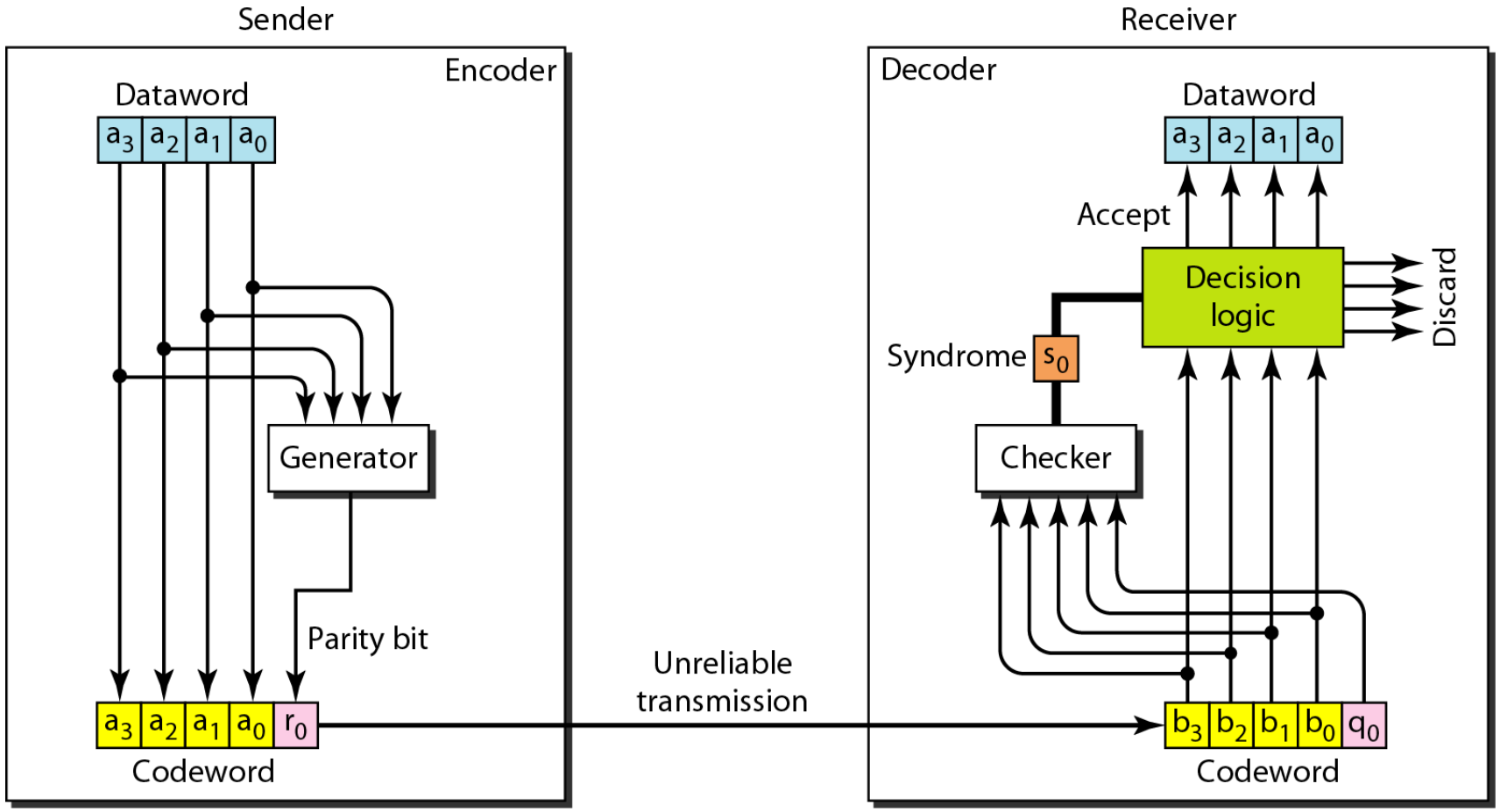
Parity-Check Code

- error-detecting code – linear block code
- n = k + 1 -- extra bit is parity bit
- Table 10.1 - a parity-check code (k = 2 and n = 3).

Table 10.3 ***Simple parity-check code C(5, 4)***

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Figure 10.10 *Encoder and decoder for simple parity-check code*



Even Parity -- r0 = a3 + a2 + a1 + a0 (modulo-2)

Example 10.12

Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

1. *No error occurs; the received codeword is 10111. The syndrome is 0. The dataword 1011 is created.*
2. *One single-bit error changes a_1. The received codeword is 10011. The syndrome is 1. No dataword is created.*
3. *One single-bit error changes r_0. The received codeword is 10110. The syndrome is 1. No dataword is created.*

Example 10.12 (continued)

4. An error changes r_0 and a second error changes a_3. The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value.
5. Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

Note

A simple parity-check code can detect an odd number of errors.

Example 10.10

Let us see if the two codes we defined in Table 10.1 and Table 10.2 belong to the class of linear block codes.

1. *The scheme in Table 10.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.*

2. *The scheme in Table 10.2 is also a linear block code. We can create all four codewords by XORing two other codewords.*

Example 10.11

In our first code (Table 10.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$. In our second code (Table 10.2), the numbers of 1s in the nonzero codewords are 3, 3, and 4. So in this code we have $d_{min} = 3$.

Note

A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$. Even parity (ensures that a codeword has an even number of 1's) and odd parity (ensures that there are an odd number of 1's in the codeword)

10-4 CYCLIC CODES

***Cyclic codes** are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.*

For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

Topics discussed in this section:

Cyclic Redundancy Check
Hardware Implementation
Polynomials
Cyclic Code Analysis
Advantages of Cyclic Codes
Other Cyclic Codes

Cyclic codes

- if we call the bits in the first word a0 to a6, and the bits in the second word b0 to b6, we can shift the bits by using the following: In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.
- b1 = a0 b2 = a1 b3 = a2 b4 = a3 b5 = a4 b6 = a5 b0 = a6

Table 10.6 *A CRC code with C(7, 4)*

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

10.

Figure 10.14 *CRC encoder and decoder*

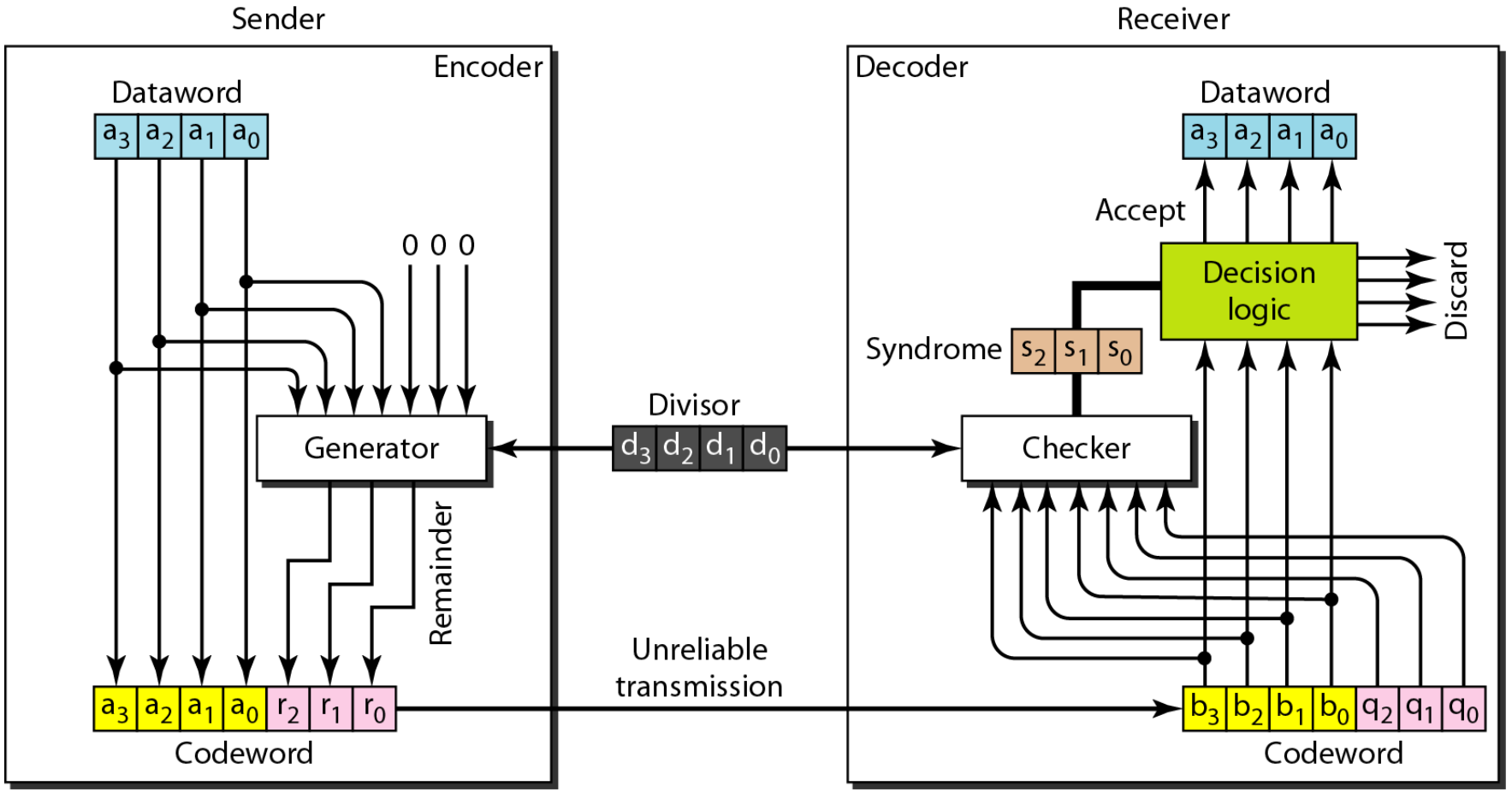


Figure 10.15 *Division in CRC encoder*

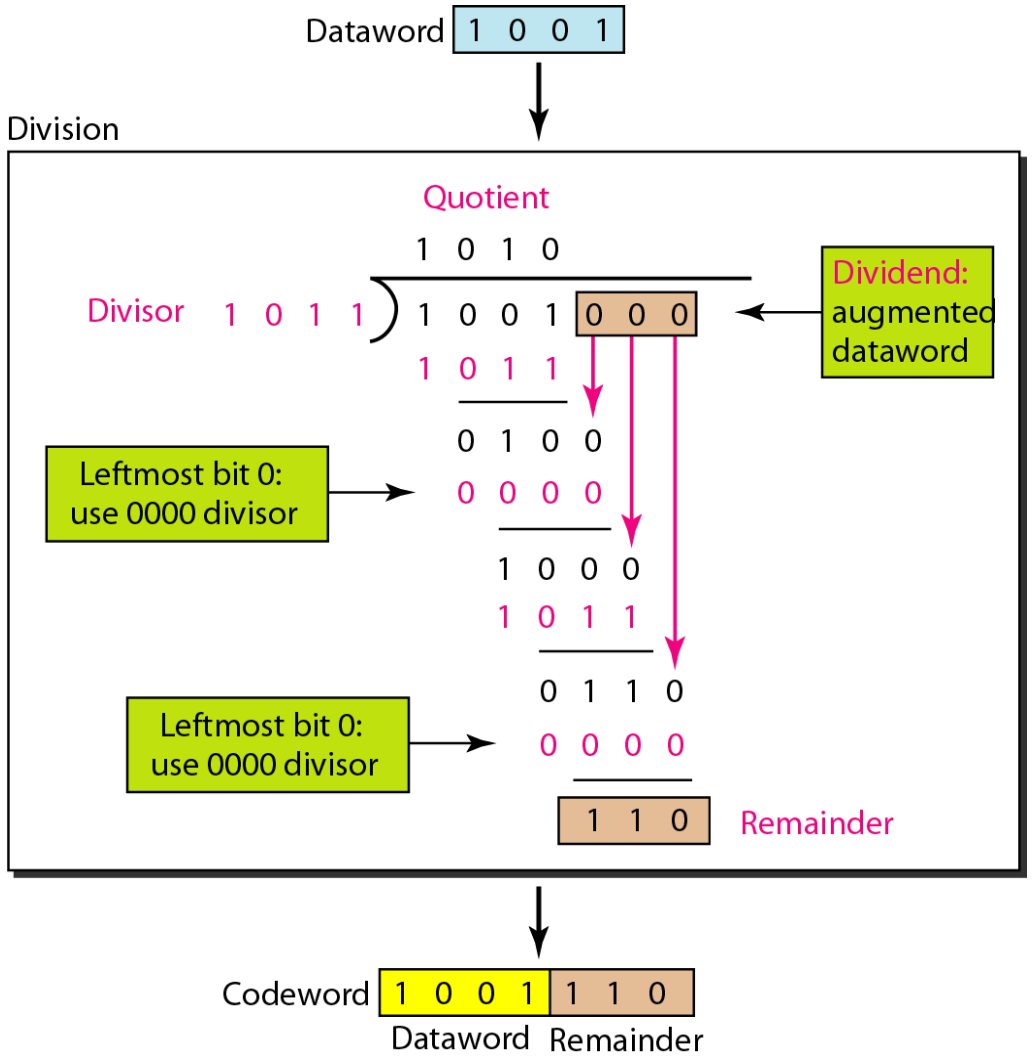


Figure 10.16 *Division in the CRC decoder for two cases*

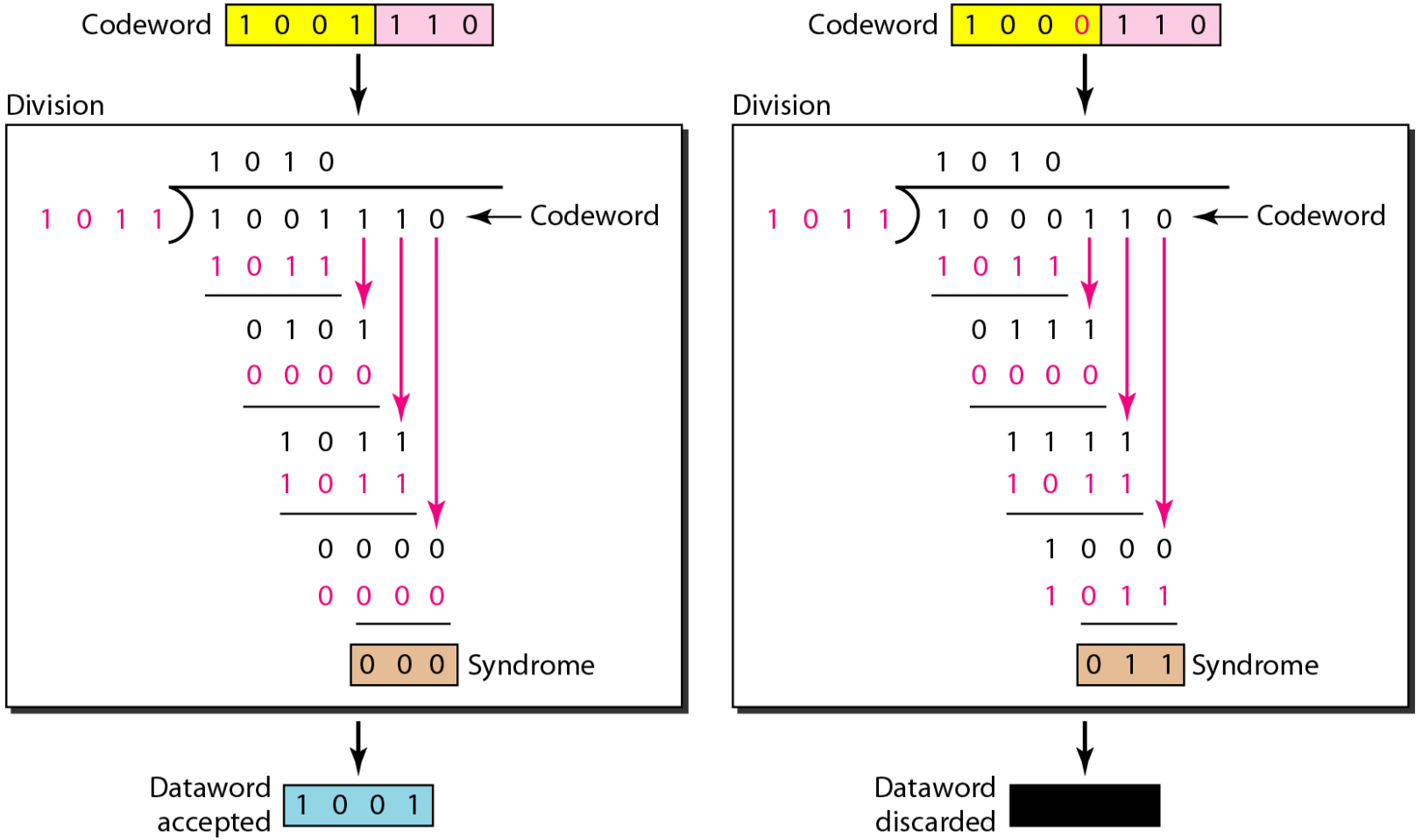


Figure 10.17 *Hardwired design of the divisor in CRC*

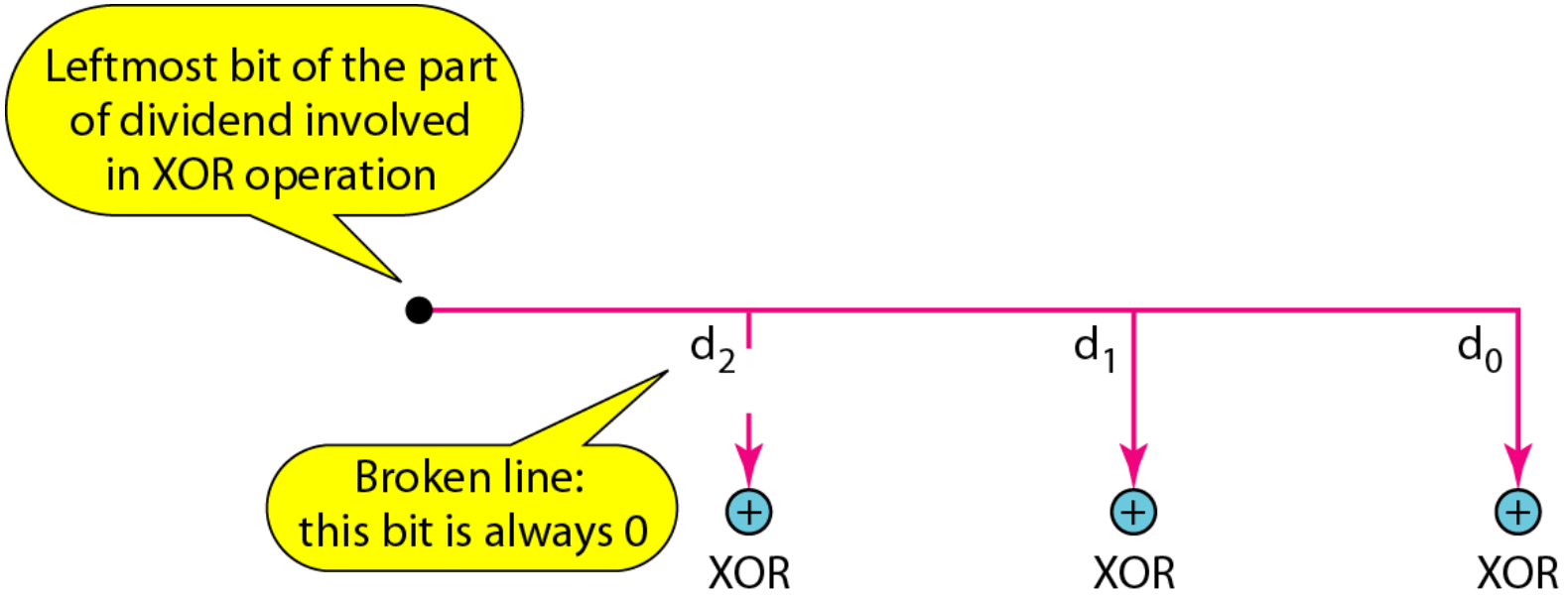


Figure 10.18 *Simulation of division in CRC encoder*

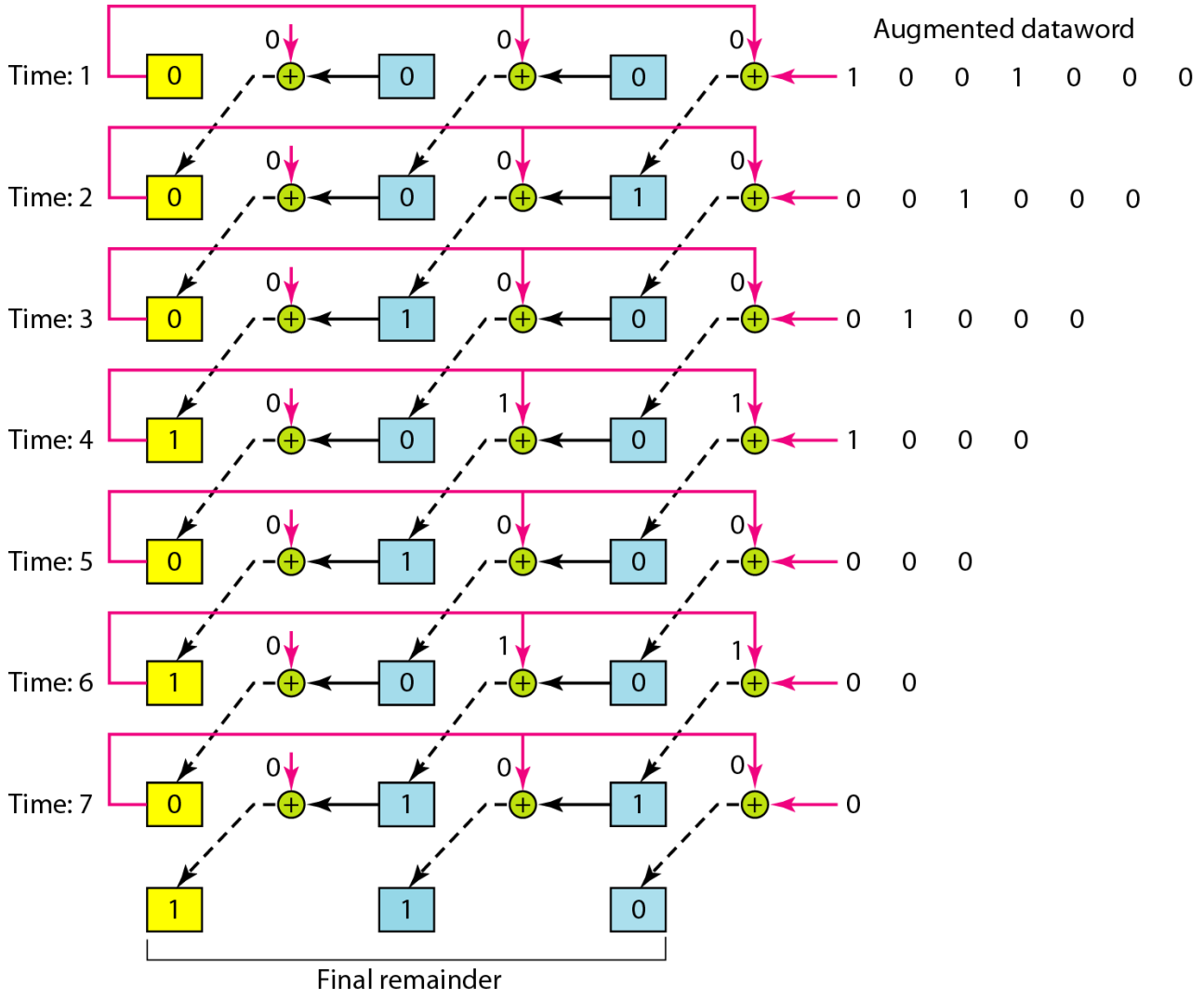


Figure 10.19 *The CRC encoder design using shift registers*

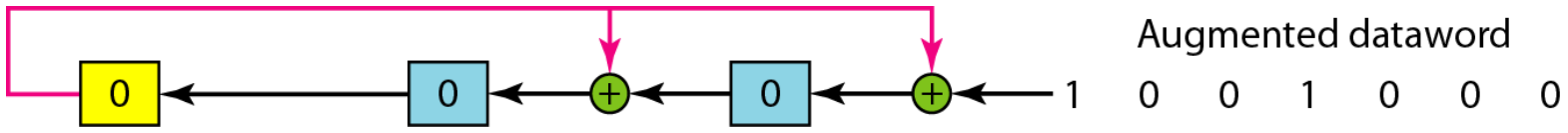
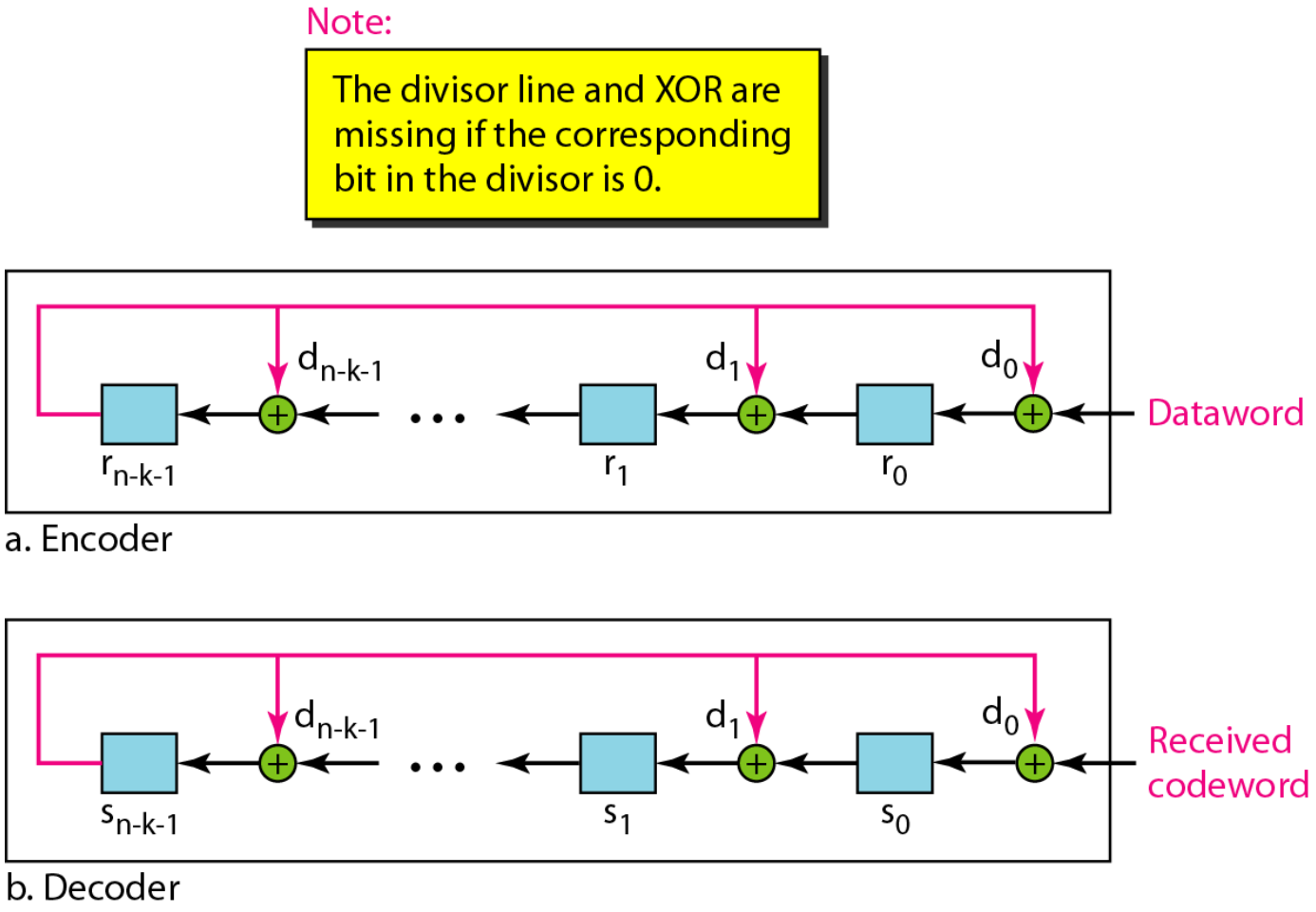


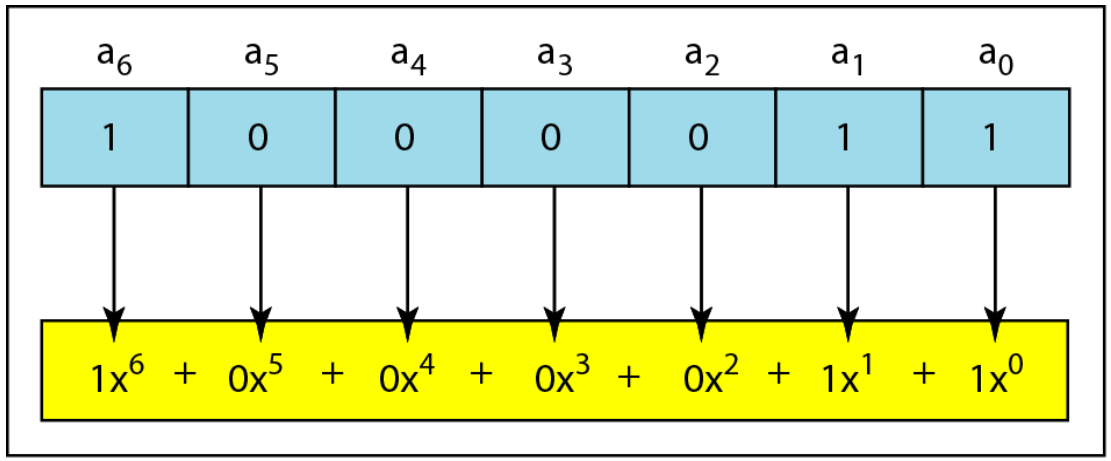
Figure 10.20 *General design of encoder and decoder of a CRC code*



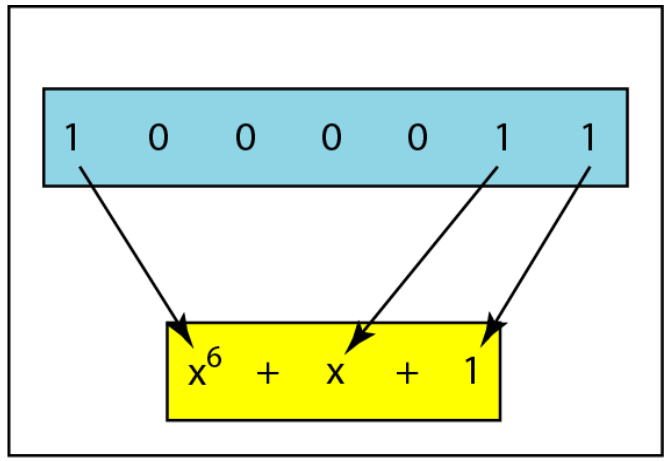
Using Polynomials

- We can use a polynomial to represent a binary word.
- Each bit from right to left is mapped onto a power term.
- The rightmost bit represents the “0” power term. The bit next to it the “1” power term, etc.
- If the bit is of value zero, the power term is deleted from the expression.

Figure 10.21 *A polynomial to represent a binary word*

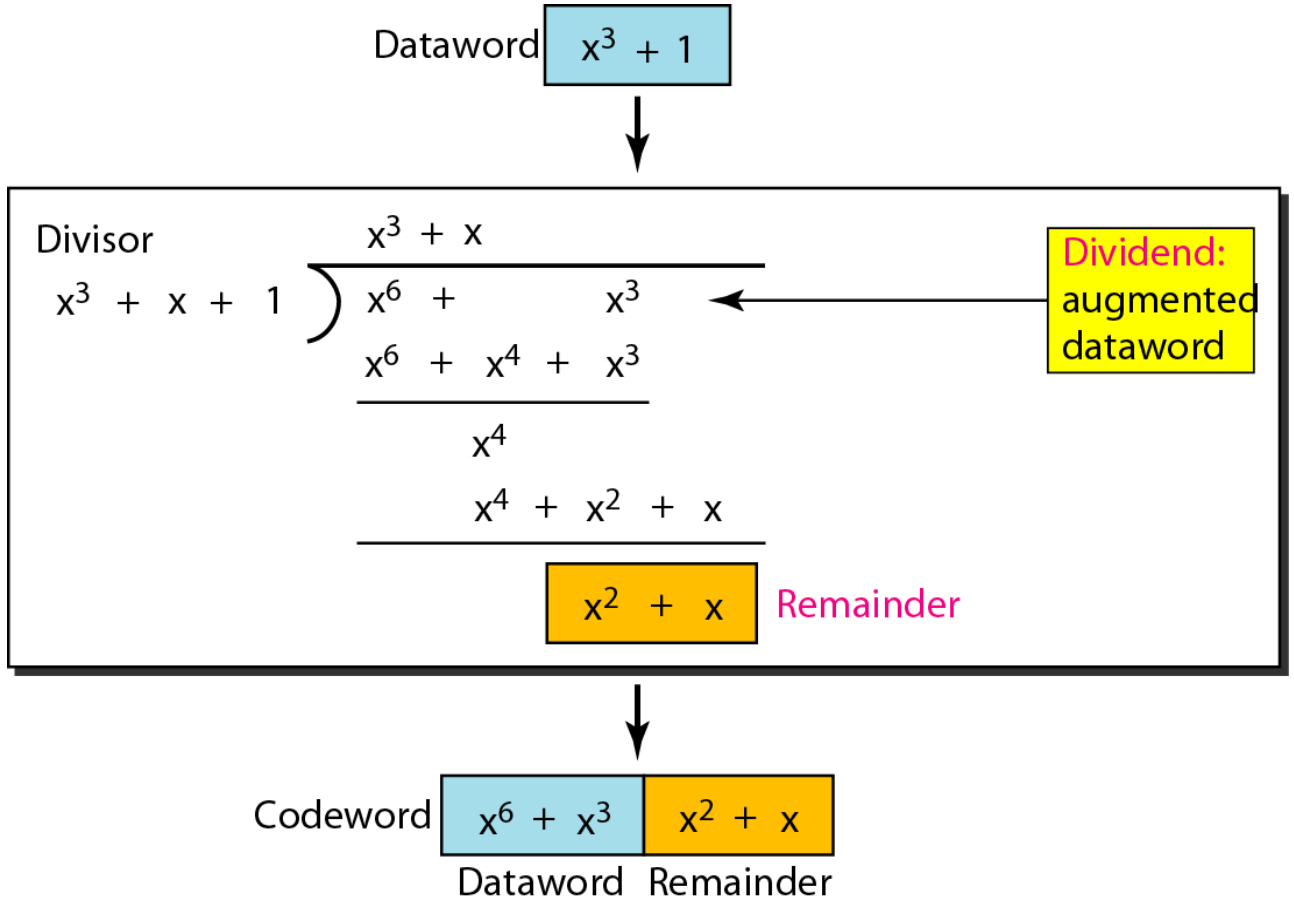


a. Binary pattern and polynomial



b. Short form

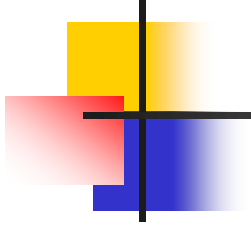
Figure 10.22 *CRC division using polynomials*



Note

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

10.



Note

In a cyclic code,

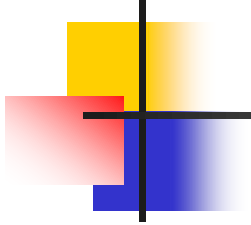
If s(x) ≠ 0, one or more bits is corrupted.

If s(x) = 0, either

a. No bit is corrupted. or

b. Some bits are corrupted, but the decoder failed to detect them.

10.



Note

In a cyclic code, those *e*(x) errors that are divisible by *g*(x) are not caught.
Received codeword (c(x) + e(x))/g(x) = c(x)/g(x) + e(x)/gx
The first part is by definition divisible the second part will determine the error. If "0" conclusion -> no error occurred.
Note: that could mean that an error went undetected.

10.

Note

If the generator has more than one term and the coefficient of x^0 is 1, all single errors can be caught.

Example 10.15

Which of the following g(x) values guarantees that a single-bit error is caught? For each case, what is the error that cannot be caught?

a. $x + 1$ ***b.*** x^3 ***c.*** 1

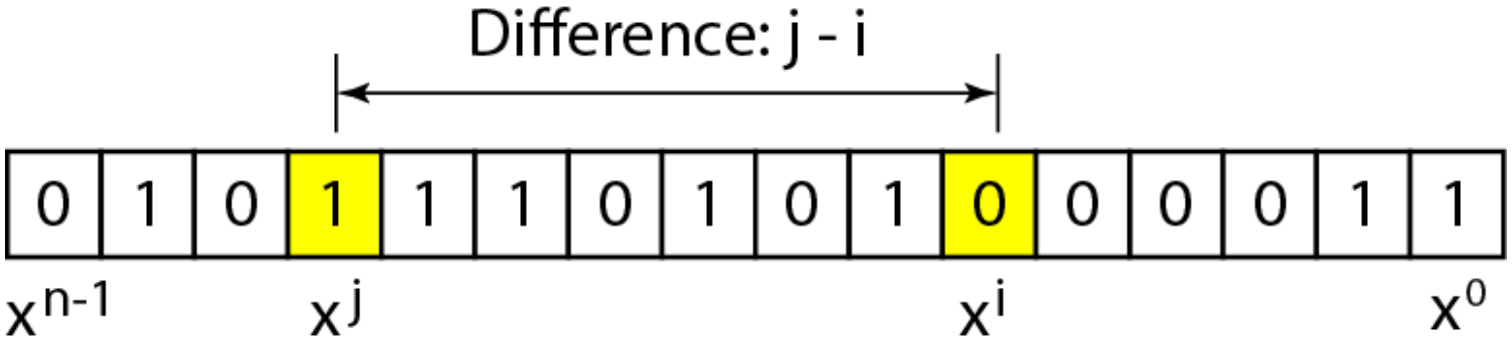
Solution

a. *No x^i can be divisible by $x + 1$. Any single-bit error can be caught.*

b. *If i is equal to or greater than 3, x^i is divisible by g(x). All single-bit errors in positions 1 to 3 are caught.*

c. *All values of i make x^i divisible by g(x). No single-bit error can be caught. This g(x) is useless.*

Figure 10.23 *Representation of two isolated single-bit errors using polynomials*



10.

Note

If a generator cannot divide $x^t + 1$ (t between 0 and n – 1), then all isolated double errors can be detected.

Example 10.16

Find the status of the following generators related to two isolated, single-bit errors.

a. $x + 1$ *b.* $x^4 + 1$ *c.* $x^7 + x^6 + 1$ *d.* $x^{15} + x^{14} + 1$

Solution

a. This is a very poor choice for a generator. Any two errors next to each other cannot be detected.

b. This generator cannot detect two errors that are four positions apart.

c. This is a good choice for this purpose.

d. This polynomial cannot divide $x^t + 1$ *if t is less than 32,768. A codeword with two isolated errors up to 32,768 bits apart can be detected by this generator.*

Note

A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.

10.

Note

- All burst errors with $L \leq r$ will be detected.
- All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.
- All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^{r}$.

Example 10.17

Find the suitability of the following generators in relation to burst errors of different lengths.

a. $x^6 + 1$ ***b.*** $x^{18} + x^7 + x + 1$ ***c.*** $x^{32} + x^{23} + x^7 + 1$

Solution

a. ***This generator can detect all burst errors with a length less than or equal to 6 bits; 3 out of 100 burst errors with length 7 will slip by; 16 out of 1000 burst errors of length 8 or more will slip by.***

Example 10.17 (continued)

b. ***This generator can detect all burst errors with a length less than or equal to 18 bits; 8 out of 1 million burst errors with length 19 will slip by; 4 out of 1 million burst errors of length 20 or more will slip by.***

c. ***This generator can detect all burst errors with a length less than or equal to 32 bits; 5 out of 10 billion burst errors with length 33 will slip by; 3 out of 10 billion burst errors of length 34 or more will slip by.***

Note

A good polynomial generator needs to have the following characteristics:

1. **It should have at least two terms.**
2. **The coefficient of the term x^0 should be 1.**
3. **It should not divide $x^t + 1$, for t between 2 and $n - 1$.**
4. **It should have the factor $x + 1$.**

Table 10.7 *Standard polynomials*

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

10.

10-5 CHECKSUM

The last error detection method we discuss here is called the checksum. The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking

Topics discussed in this section:

Idea
One's Complement
Internet Checksum

Example 10.18

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

Example 10.19

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12, 0, 6, −36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

Example 10.20

How can we represent the number 21 in one's complement arithmetic using only four bits?

Solution

The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

Example 10.21

How can we represent the number −6 in one's complement arithmetic using only four bits?

Solution

In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 1001. If we consider only unsigned numbers, this is 9. In other words, the complement of 6 is 9. Another way to find the complement of a number in one's complement arithmetic is to subtract the number from $2^n - 1$ (16 − 1 in this case).

Example 10.22

Let us redo Exercise 10.19 using one's complement arithmetic. Figure 10.24 shows the process at the sender and at the receiver. The sender initializes the checksum to 0 and adds all data items and the checksum (the checksum is considered as one data item and is shown in color). The result is 36. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6. In the figure, we have shown the details in binary. The sum is then complemented, resulting in the checksum value 9 (15 − 6 = 9). The sender now sends six data items to the receiver including the checksum 9.

Example 10.22 (continued)

The receiver follows the same procedure as the sender. It adds all data items (including the checksum); the result is 45. The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes 0. Since the value of the checksum is 0, this means that the data is not corrupted. The receiver drops the checksum and keeps the other data items. If the checksum is not zero, the entire packet is dropped.

Figure 10.24 *Example 10.22*

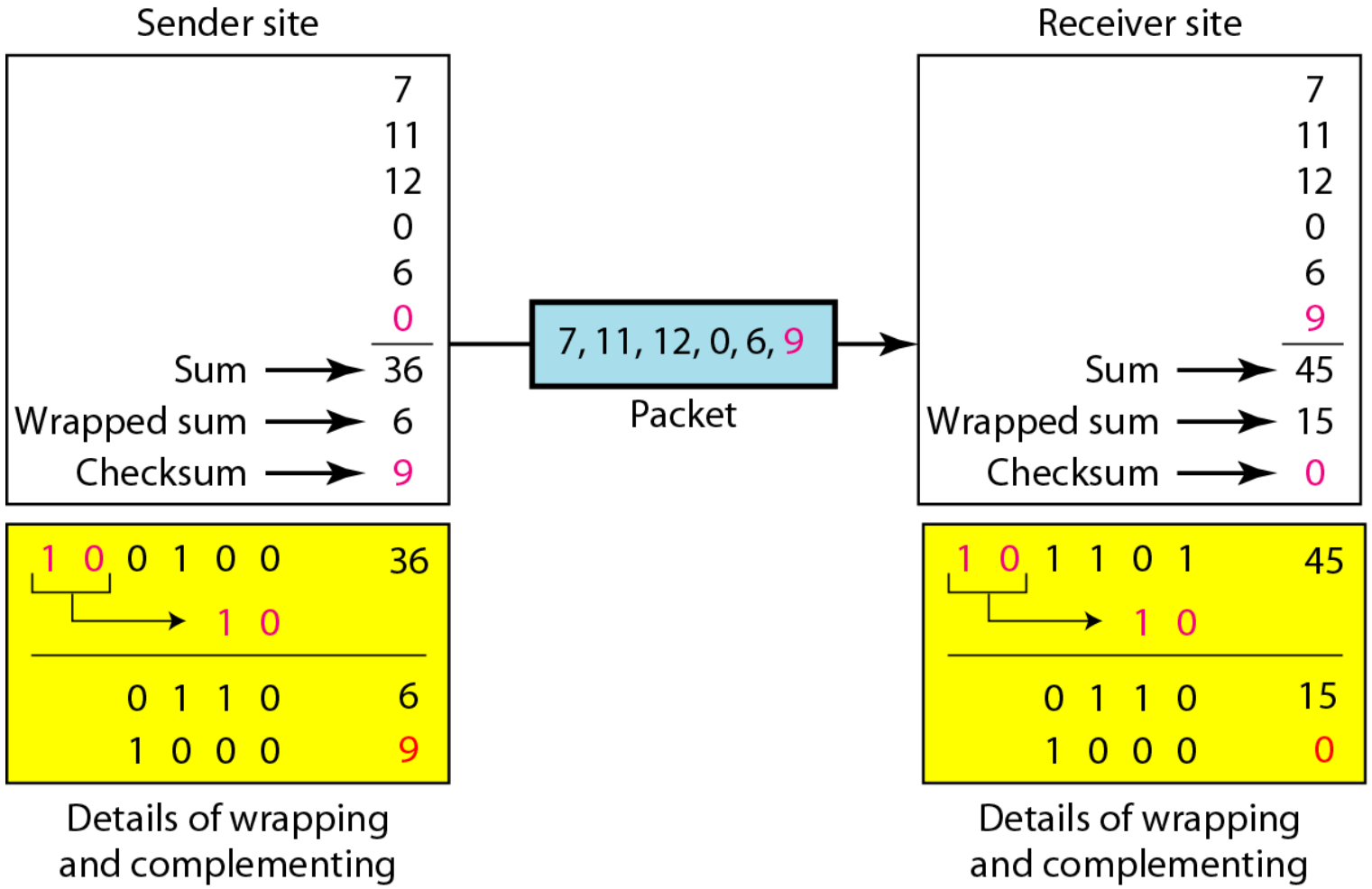
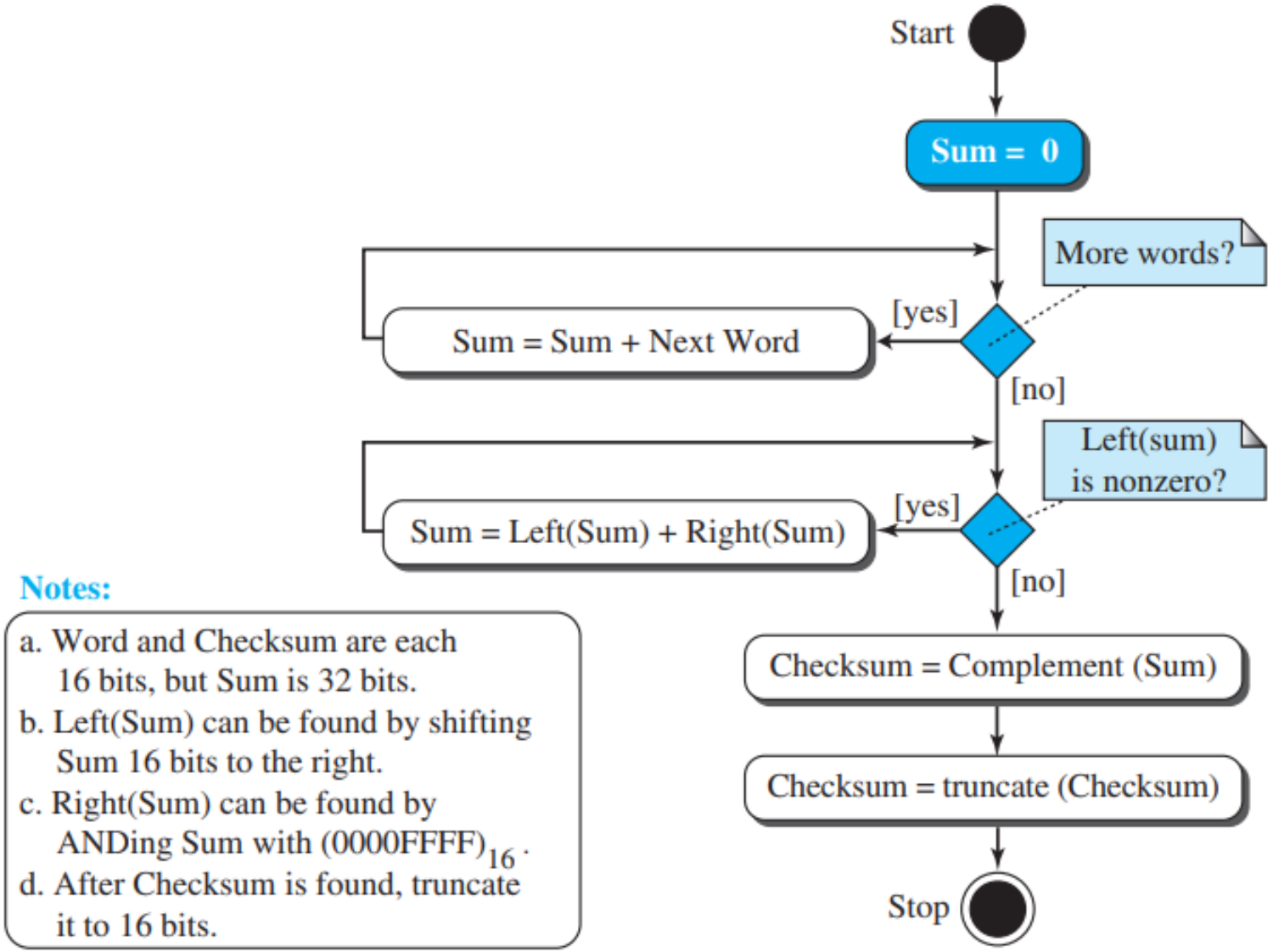


Figure 10.17 *Algorithm to calculate a traditional checksum*



Notes:

a. Word and Checksum are each 16 bits, but Sum is 32 bits.
b. Left(Sum) can be found by shifting Sum 16 bits to the right.
c. Right(Sum) can be found by ANDing Sum with $(0000FFFF)_{16}$.
d. After Checksum is found, truncate it to 16 bits.

Table 10.5 *Procedure to calculate the traditional checksum*

Sender	*Receiver*
1. The message is divided into 16-bit words.	1. The message and the checksum are received.
2. The value of the checksum word is initially set to zero.	2. The message is divided into 16-bit words.
3. All words including the checksum are added using one's complement addition.	3. All words are added using one's complement addition.
4. The sum is complemented and becomes the checksum.	4. The sum is complemented and becomes the new checksum.
5. The checksum is sent with the data.	5. If the value of the checksum is 0, the message is accepted; otherwise, it is rejected.

Note

Sender site:

1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

Note

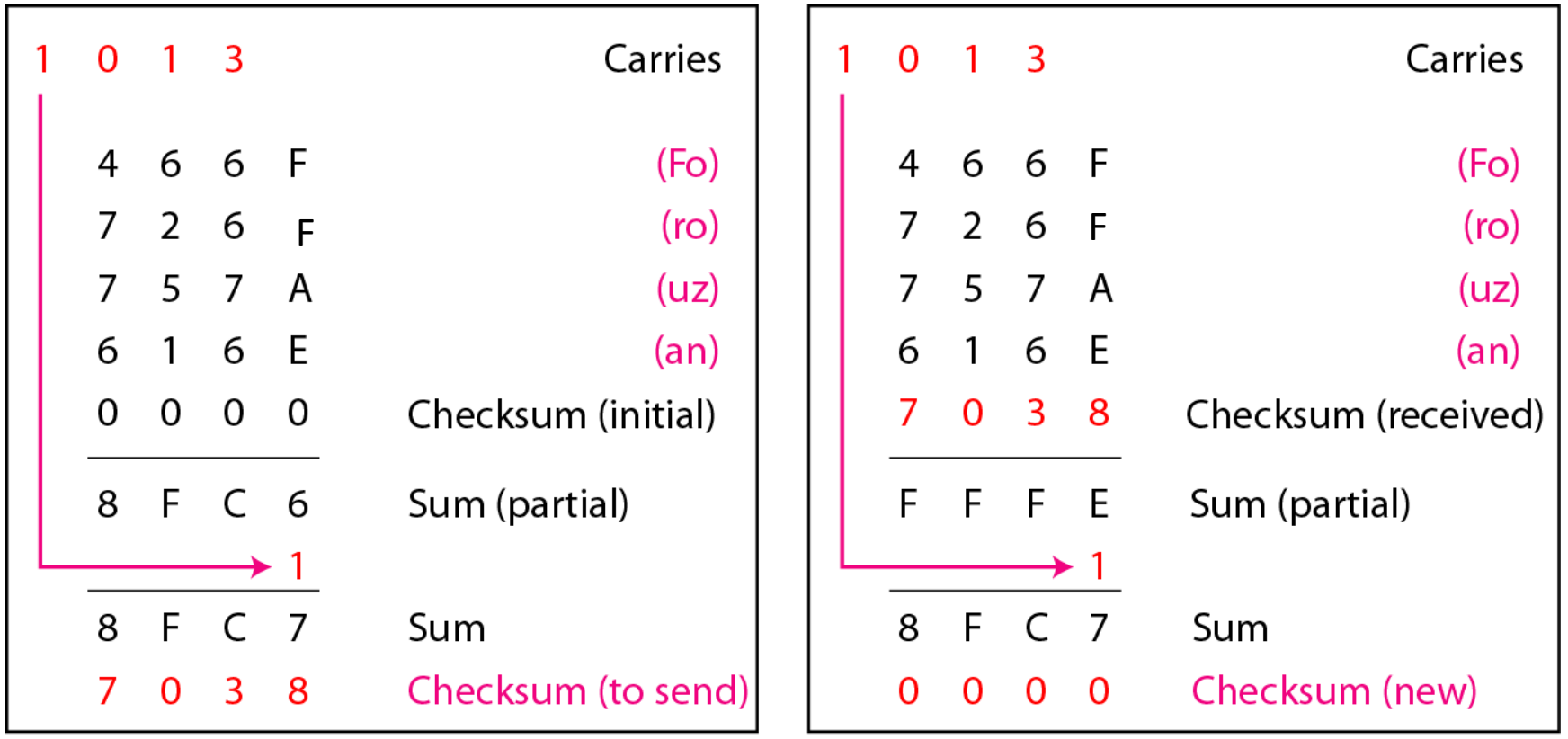
Receiver site:

1. **The message (including checksum) is divided into 16-bit words.**
2. **All words are added using one's complement addition.**
3. **The sum is complemented and becomes the new checksum.**
4. **If the value of checksum is 0, the message is accepted; otherwise, it is rejected.**

Example 10.23

Let us calculate the checksum for a text of 8 characters ("Forouzan"). The text needs to be divided into 2-byte (16-bit) words. We use ASCII (see Appendix A) to change each byte to a 2-digit hexadecimal number. For example, F is represented as 0x46 and o is represented as 0x6F. Figure 10.25 shows how the checksum is calculated at the sender and receiver sites. In part a of the figure, the value of partial sum for the first column is 0x36. We keep the rightmost digit (6) and insert the leftmost digit (3) as the carry in the second column. The process is repeated for each column. Note that if there is any corruption, the checksum recalculated by the receiver is not all 0s. We leave this an exercise.

Figure 10.25 *Example 10.23*



a. Checksum at the sender site

a. Checksum at the receiver site

Note : F + F + A + E + 0 -> 15+15+10+14=54 54/16 Quo-> 3 (carry), Re-6

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

- Try

- (x3 + x2 + x + 1) / (x2 + 1)
- Assume a packet is made only of four 16-bit words (A7A2)16, (CABF)16, (903A)16, and (A123)16. Manually simulate the algorithm in Figure 10.17 to find the checksum.
- Given the dataword 101001111 and the divisor 10111, show the generation of the CRC codeword at the sender site (using binary division).

TEST1 PORTIONS END HERE

Chapter 11

Data Link Control

11-1 FRAMING

The data link layer needs to pack bits into frames, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.

Topics discussed in this section:

Fixed-Size Framing

Variable-Size Framing

Figure 11.1 *A frame in a character-oriented protocol*

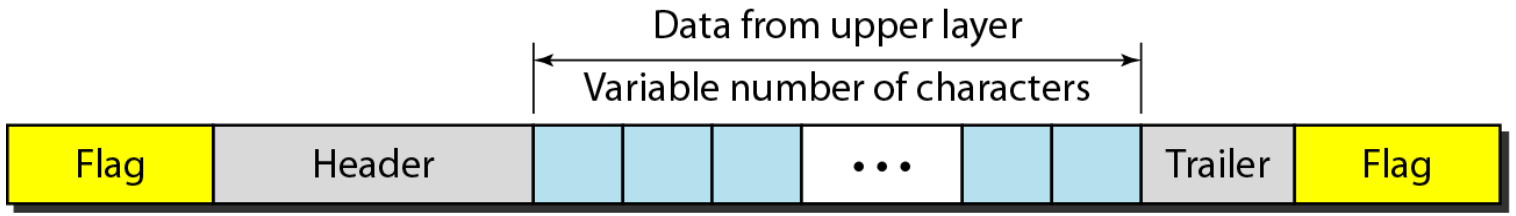
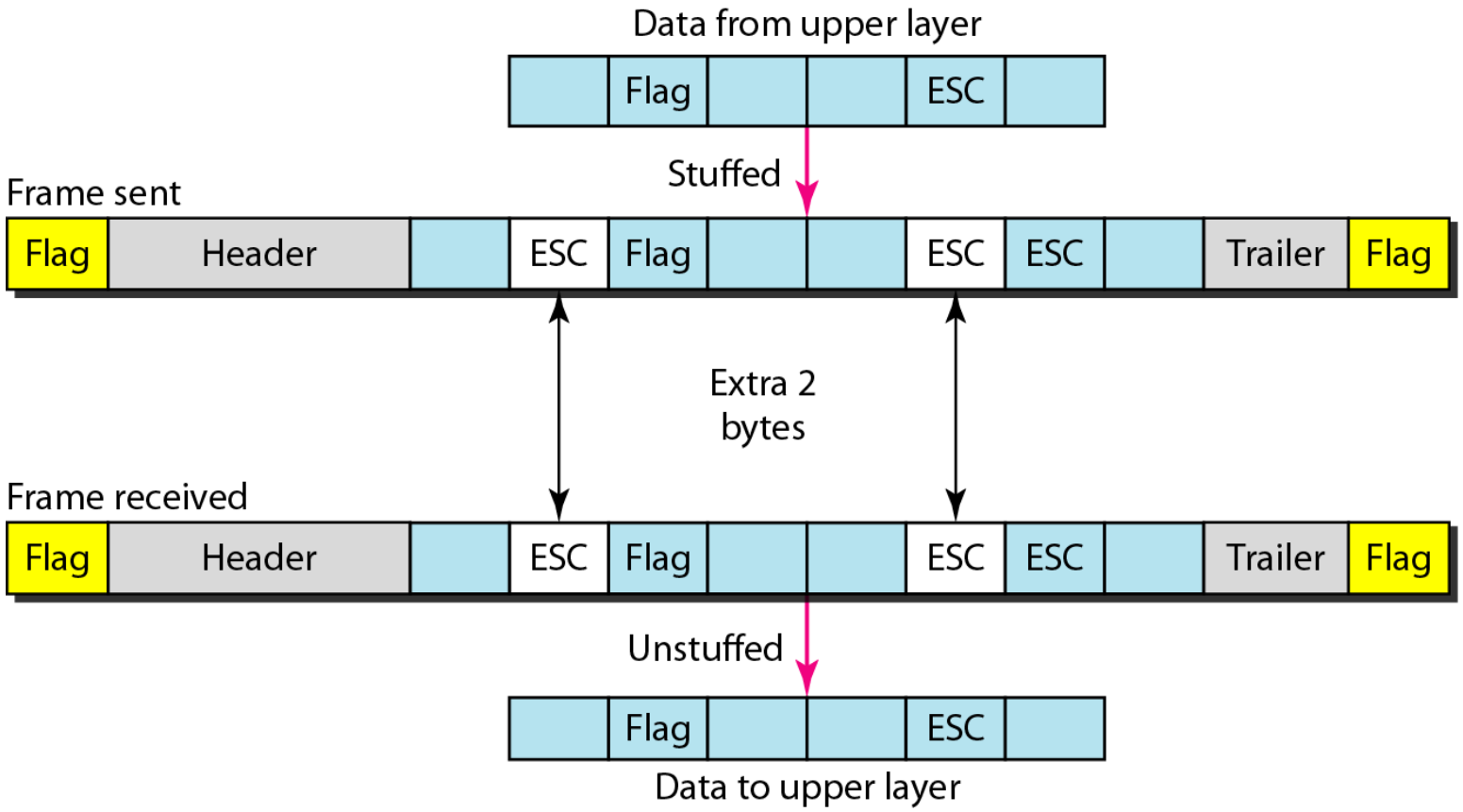


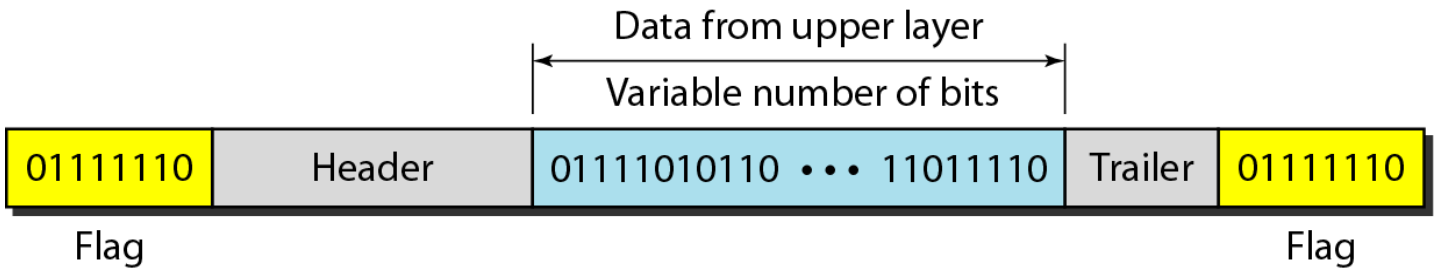
Figure 11.2 *Byte stuffing and unstuffing*



Note

Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

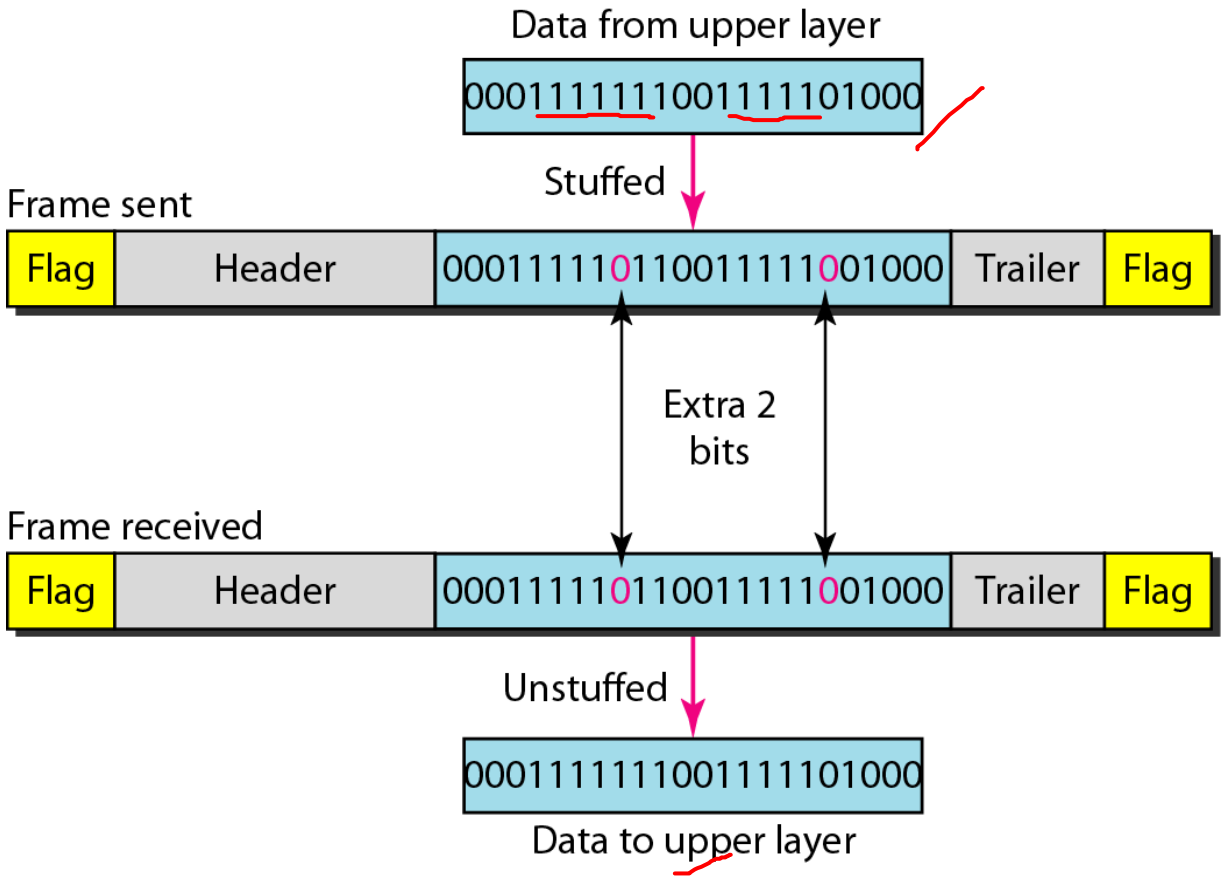
Figure 11.3 *A frame in a bit-oriented protocol*



Note

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

Figure 11.4 *Bit stuffing and unstuffing*



11-2 FLOW AND ERROR CONTROL

The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link control.

Topics discussed in this section:

Flow Control
Error Control

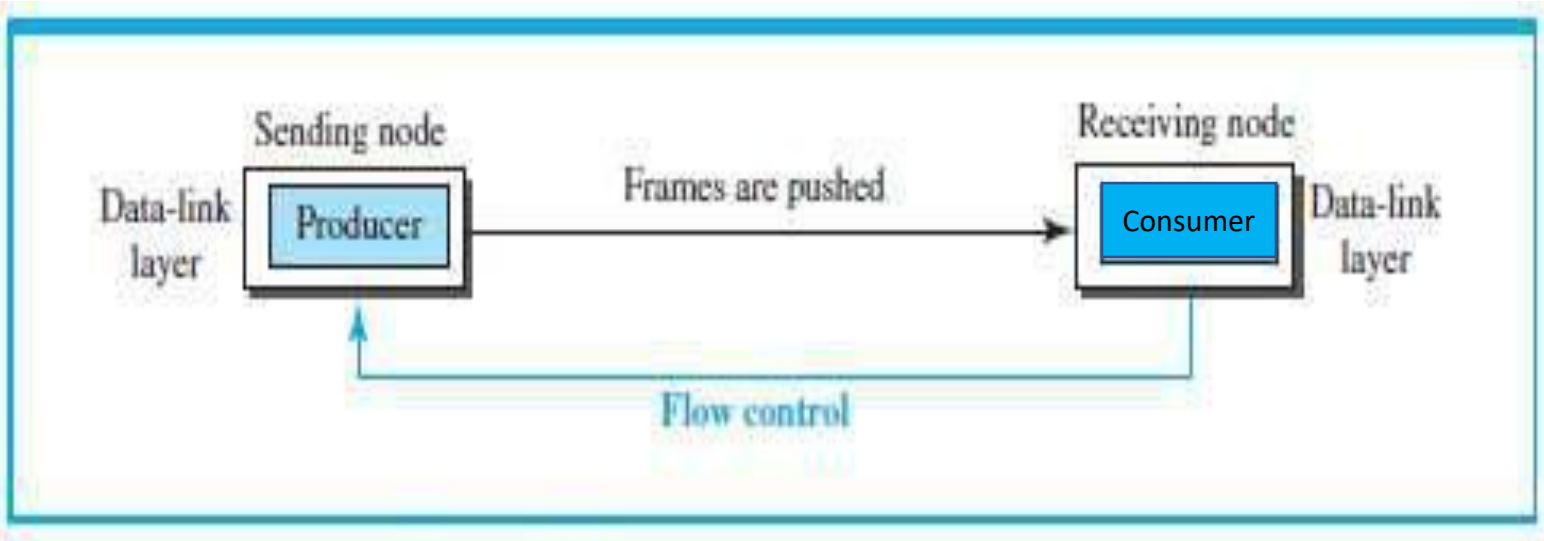


Figure 11.5 *Flow control at the data-link layer*

Note

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

Note

Error control in the data link layer is based on automatic repeat request, which is the retransmission of data.

Note

Flow control can be implemented by using buffer.
A buffer is a set of memory locations that can hold packets at the sender and receiver.

Note

Combination of Flow and Error Control

1) Connectionless Protocol

2) Connection Oriented Protocol

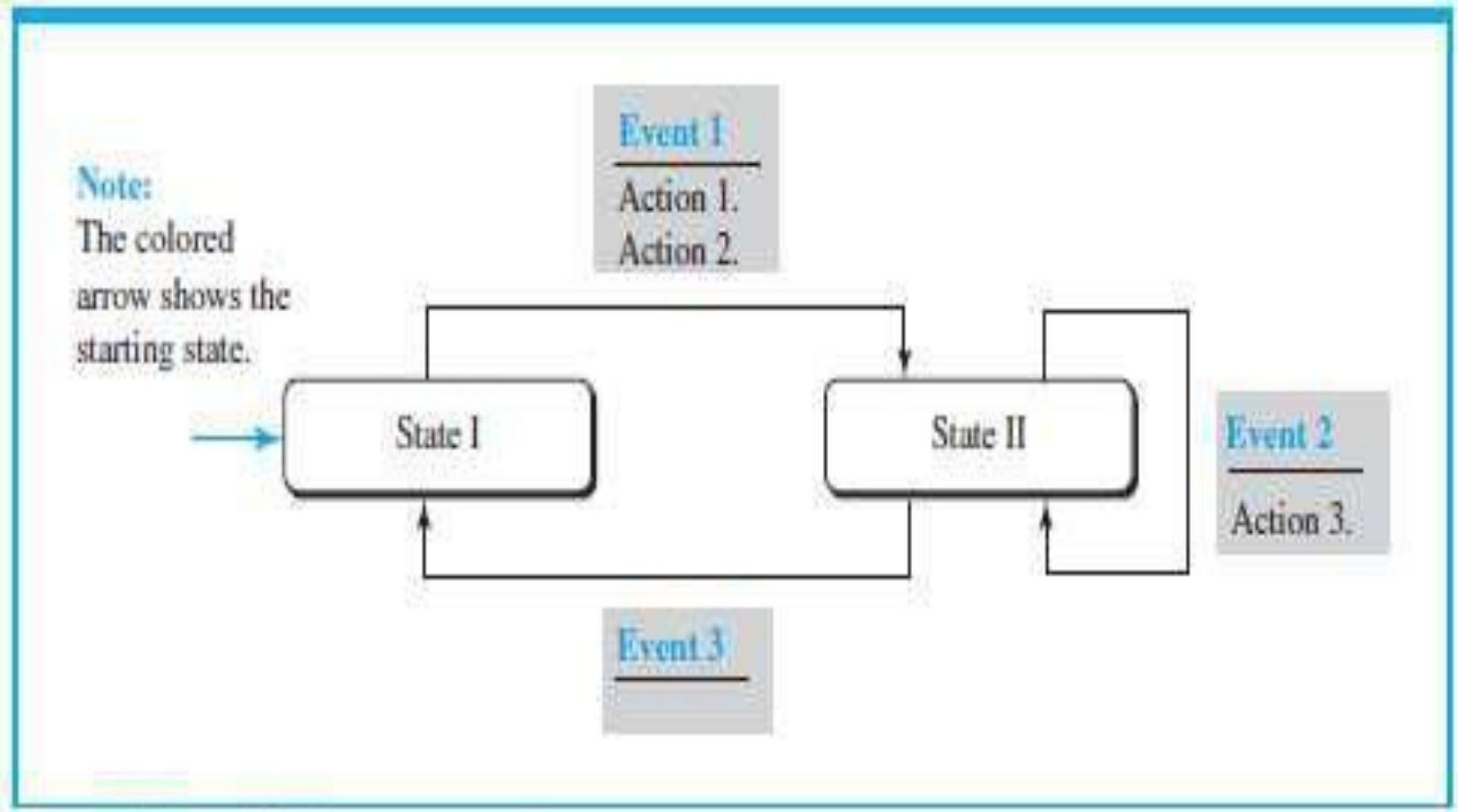
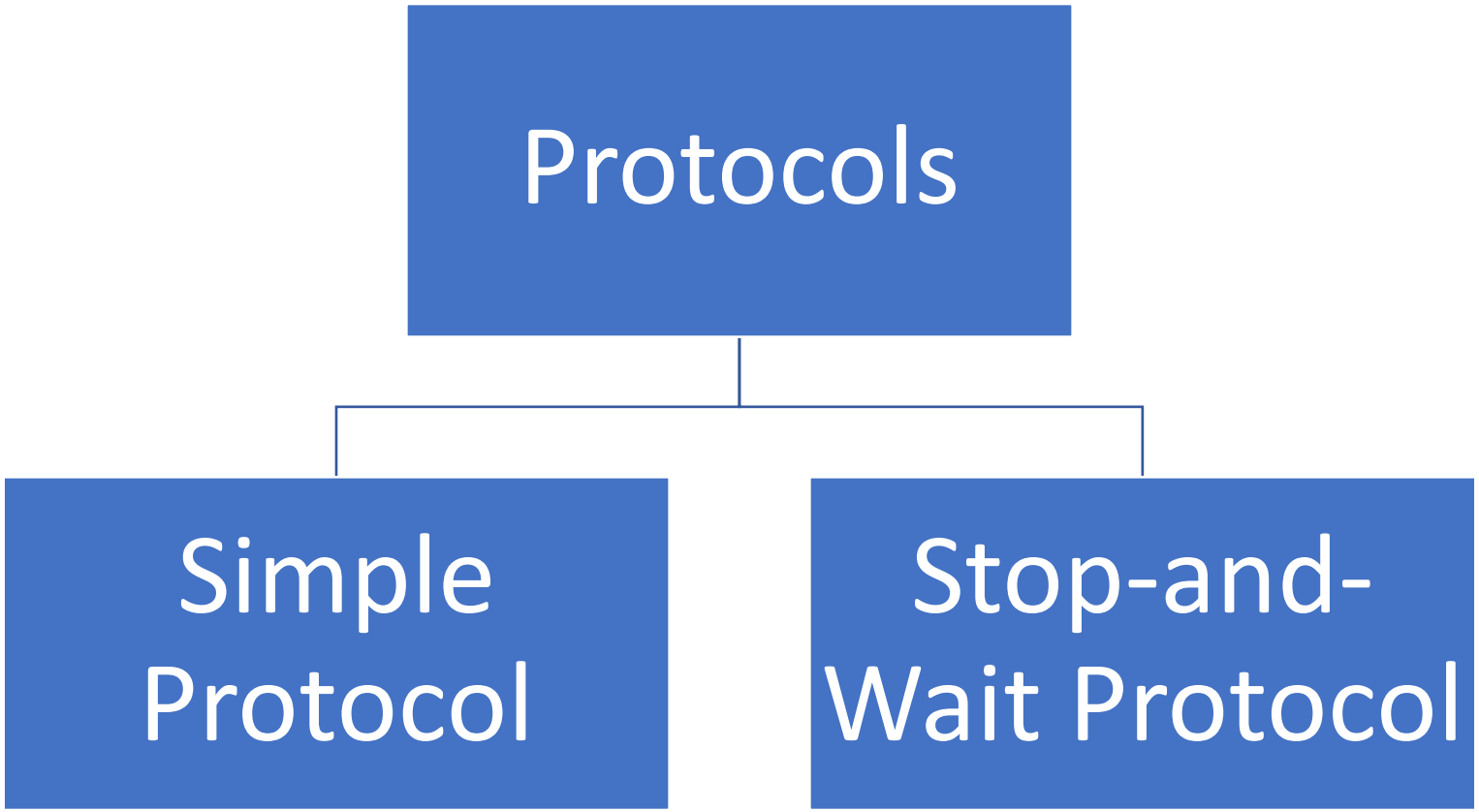


Figure 11.6 *Connectionless and connection-oriented service represented as FSMs*

11-3 PROTOCOLS

Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages. To make our discussions language-free, we have written in pseudocode a version of each protocol that concentrates mostly on the procedure instead of delving into the details of language rules.

Figure 11.5 *Taxonomy of protocols discussed in this chapter*



11-4 NOISELESS CHANNELS

Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel.

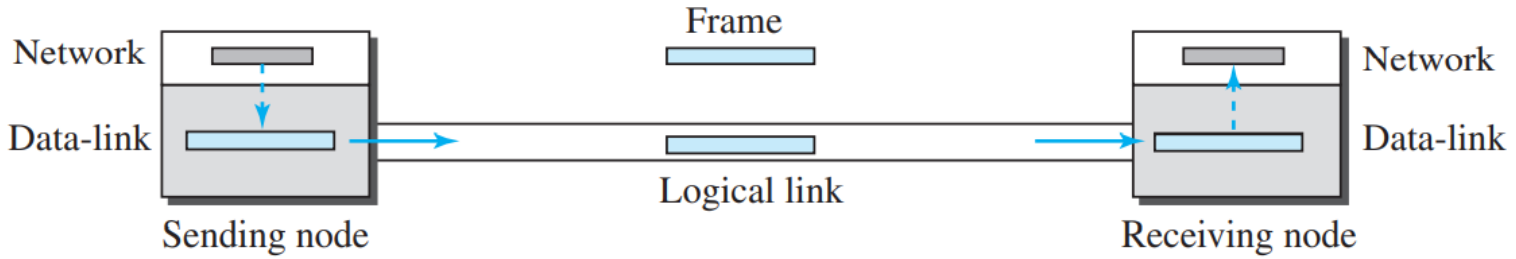
Topics discussed in this section:

Simplest Protocol

Stop-and-Wait Protocol

SIMPLE PROTOCOL

Figure 11.7 *Simple protocol*



Assumption – best case - No check on reception – every packet transmitted is received and sent to NL

Figure 11.8 *FSMs for the simple protocol*

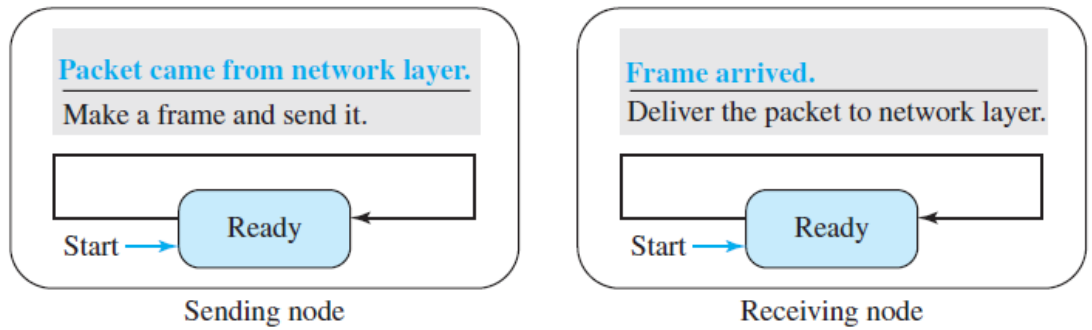
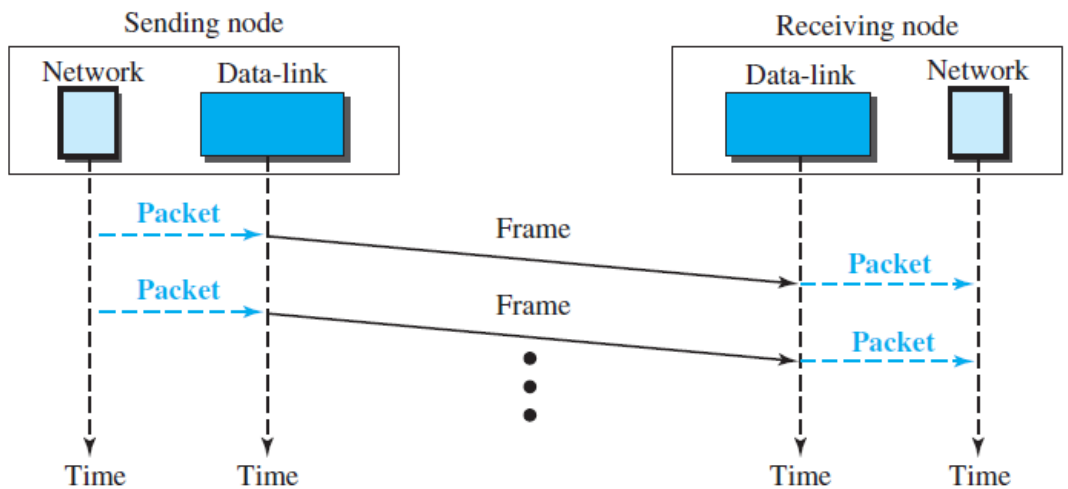


Figure 11.9 *Flow diagram for Example 11.2*



Design of Stop-and-Wait Protocol

Figure 11.10 *Stop-and-Wait protocol*

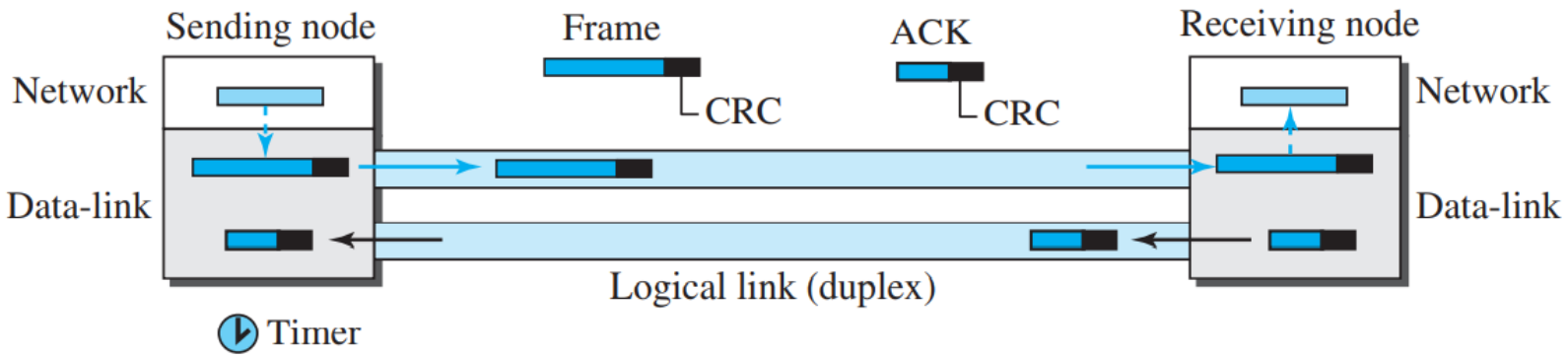


Figure 11.11 *FSM for the Stop-and-Wait protocol*

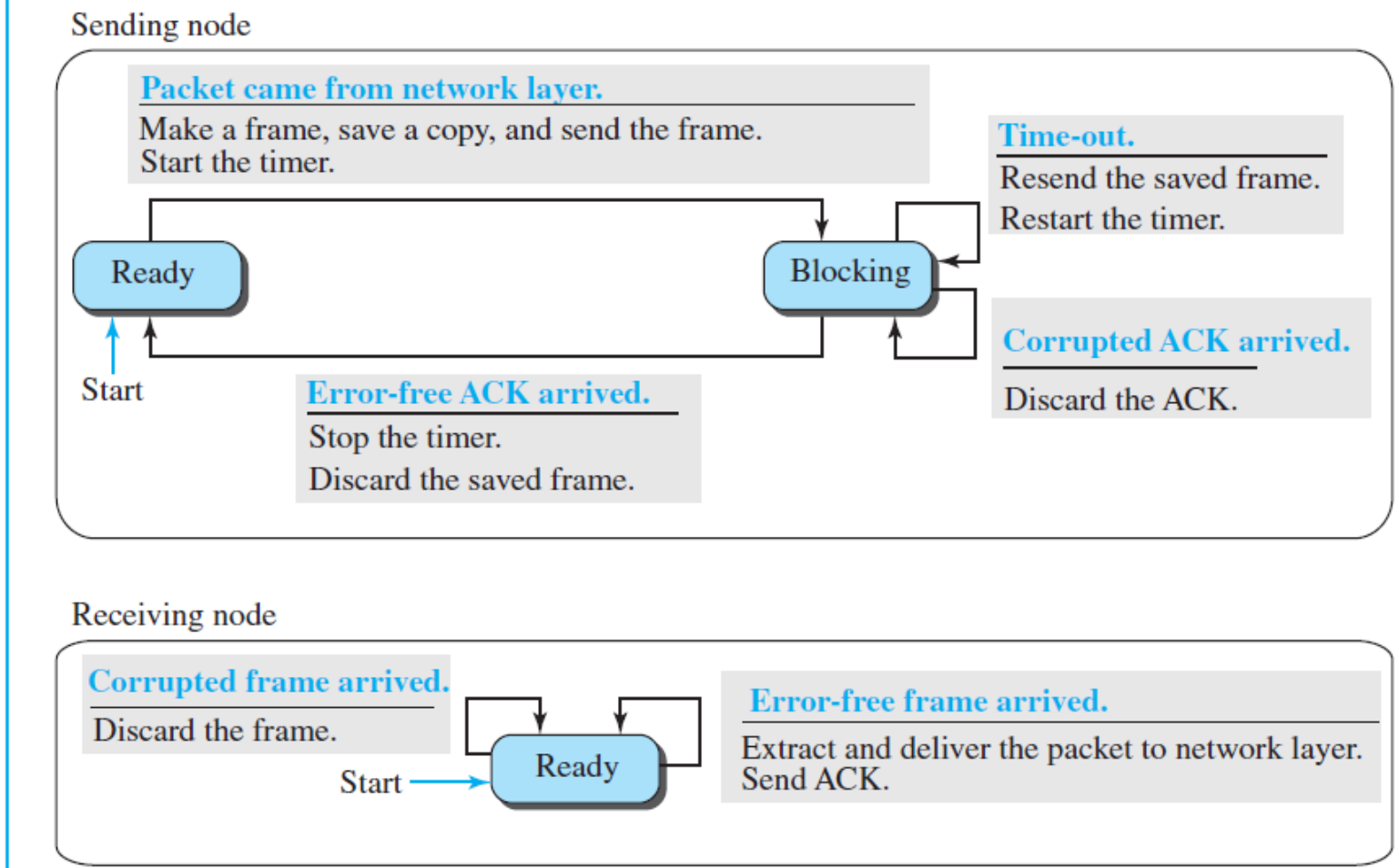
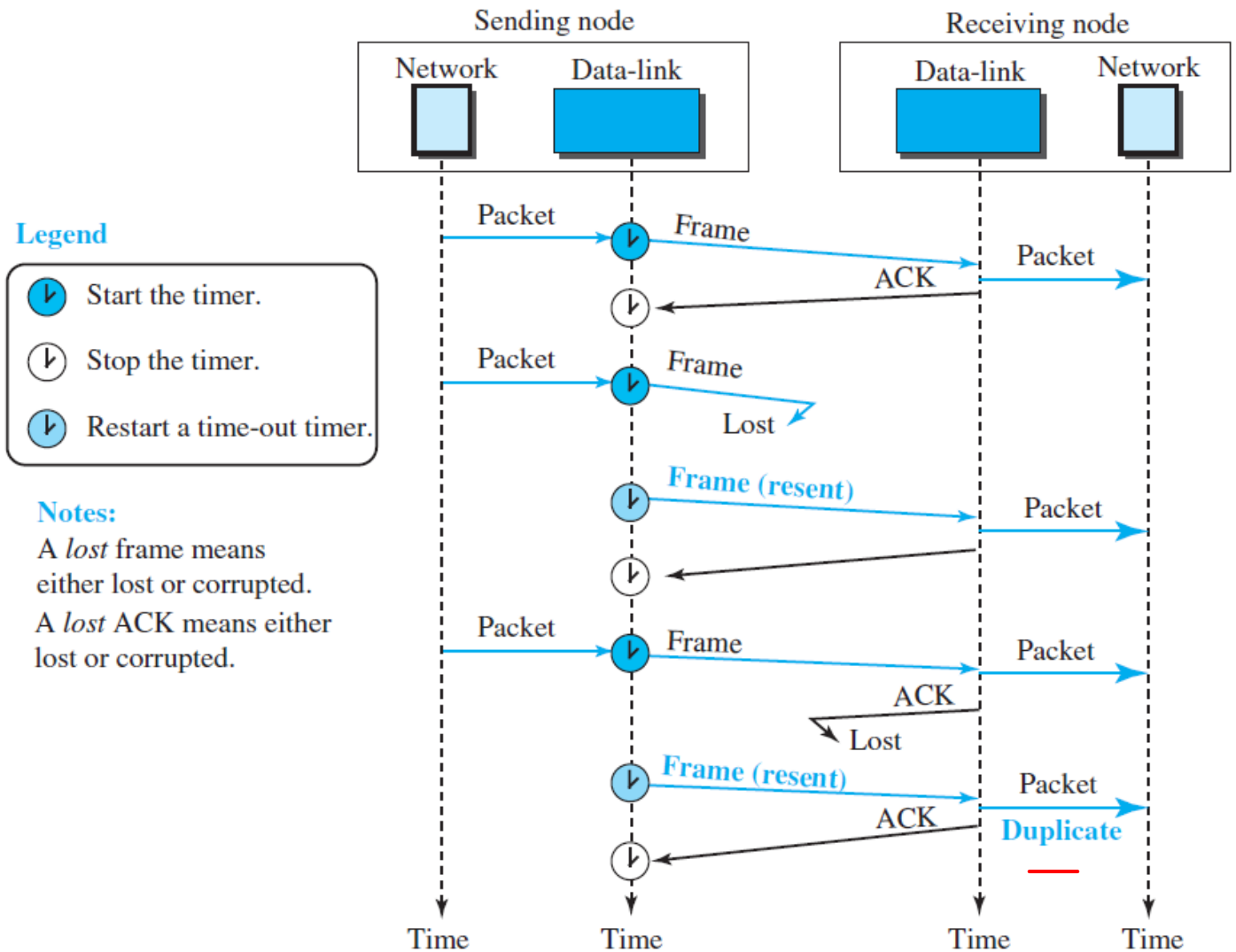


Figure 11.12 *Flow diagram for Example 11.3*



Sequence and Acknowledgment Numbers:

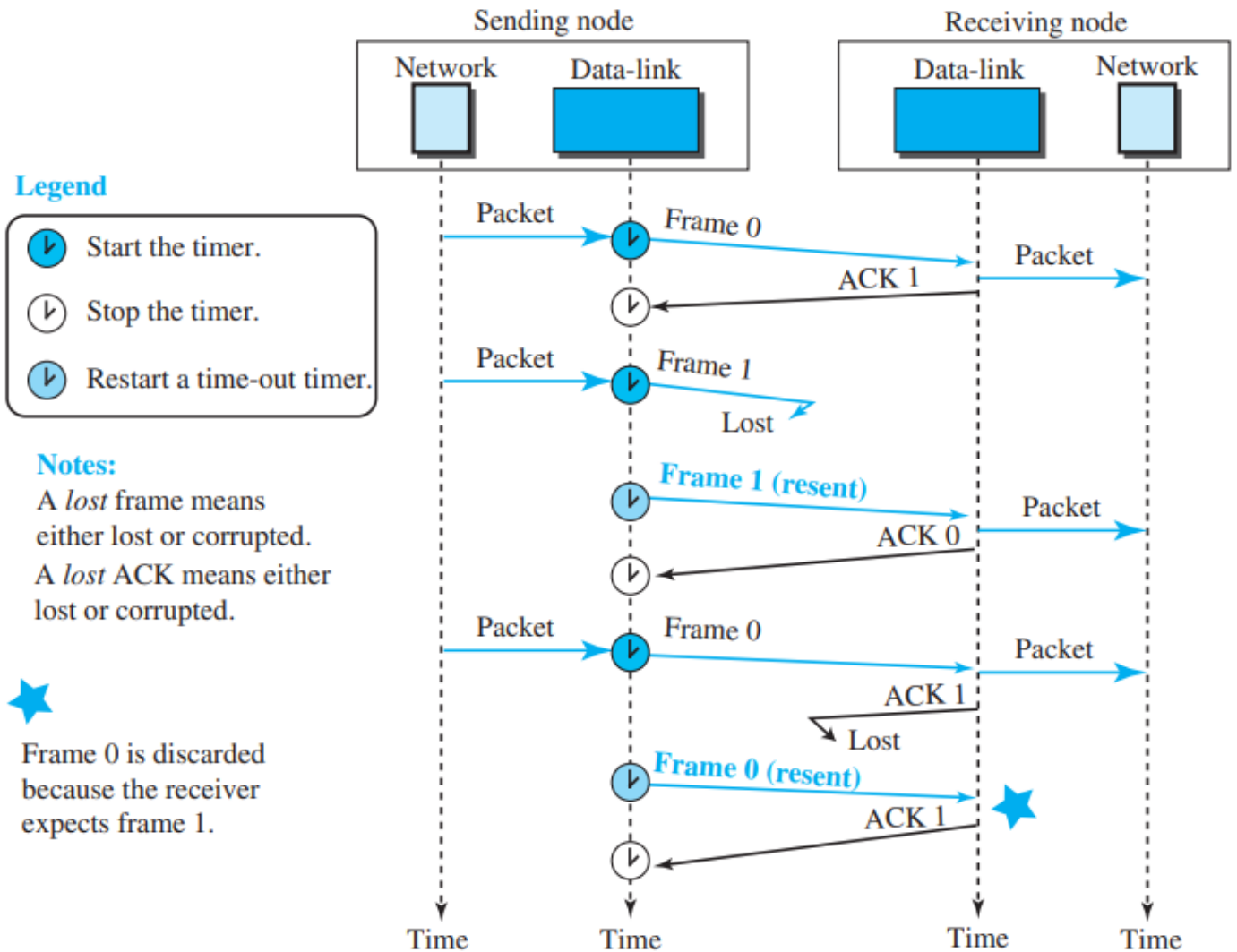
Duplicate packets, as much as corrupted packets, need to be avoided.

We need to add sequence numbers to the data frames and acknowledgment numbers to the ACK frames

Sequence numbers are 0, 1, 0, 1, 0, 1, . . . ; the acknowledgment numbers can also be 1, 0, 1, 0, 1, 0, …

An acknowledgment number always defines the sequence number of the next frame to receive.

Figure 11.13 *Flow diagram for Example 11.4*



Piggybacking:

To make the communication more efficient, the data in one direction is piggybacked with the acknowledgment in the other direction.

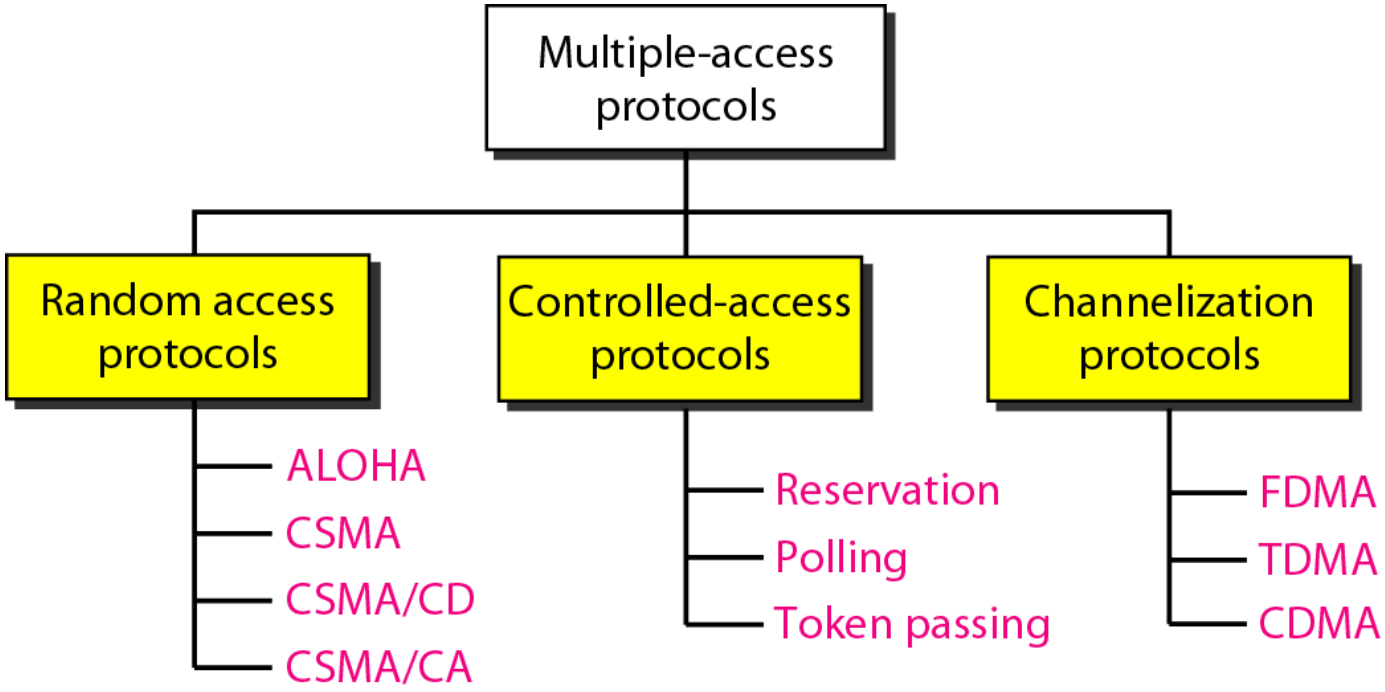
When node A is sending data to node B, Node A also acknowledges the data received from node B. Because piggybacking makes communication at the datalink layer more complicated, it is not a common practice.

Chapter 12

Multiple Access

Figure 12.2 *Taxonomy of multiple-access protocols discussed in this chapter*

When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link



RANDOM ACCESS

No station is superior to another station and none is assigned control over another

Each station can transmit when it desires on the condition that it follows the predefined procedure, including testing the state of the medium.

Two features

1. no scheduled time for a station to transmit

Transmission is random, hence named random access

2. no rules specify which station should send next.

Stations compete with one another to access the medium. That is why these methods are also called contention methods.

If more than one station tries to send, there is an access conflict—collision—and the frames will be either destroyed or modified.

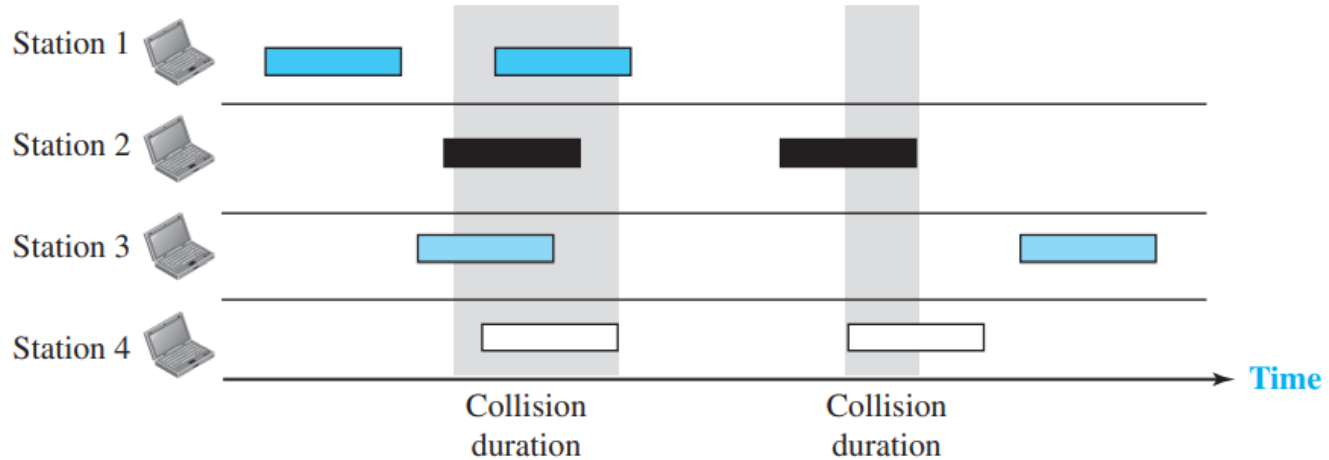
a very interesting protocol known as ALOHA, which used a very simple procedure called multiple access (MA).

ALOHA

It was designed for a radio (wireless) LAN, but it can be used on any shared medium.

medium is shared between the stations.

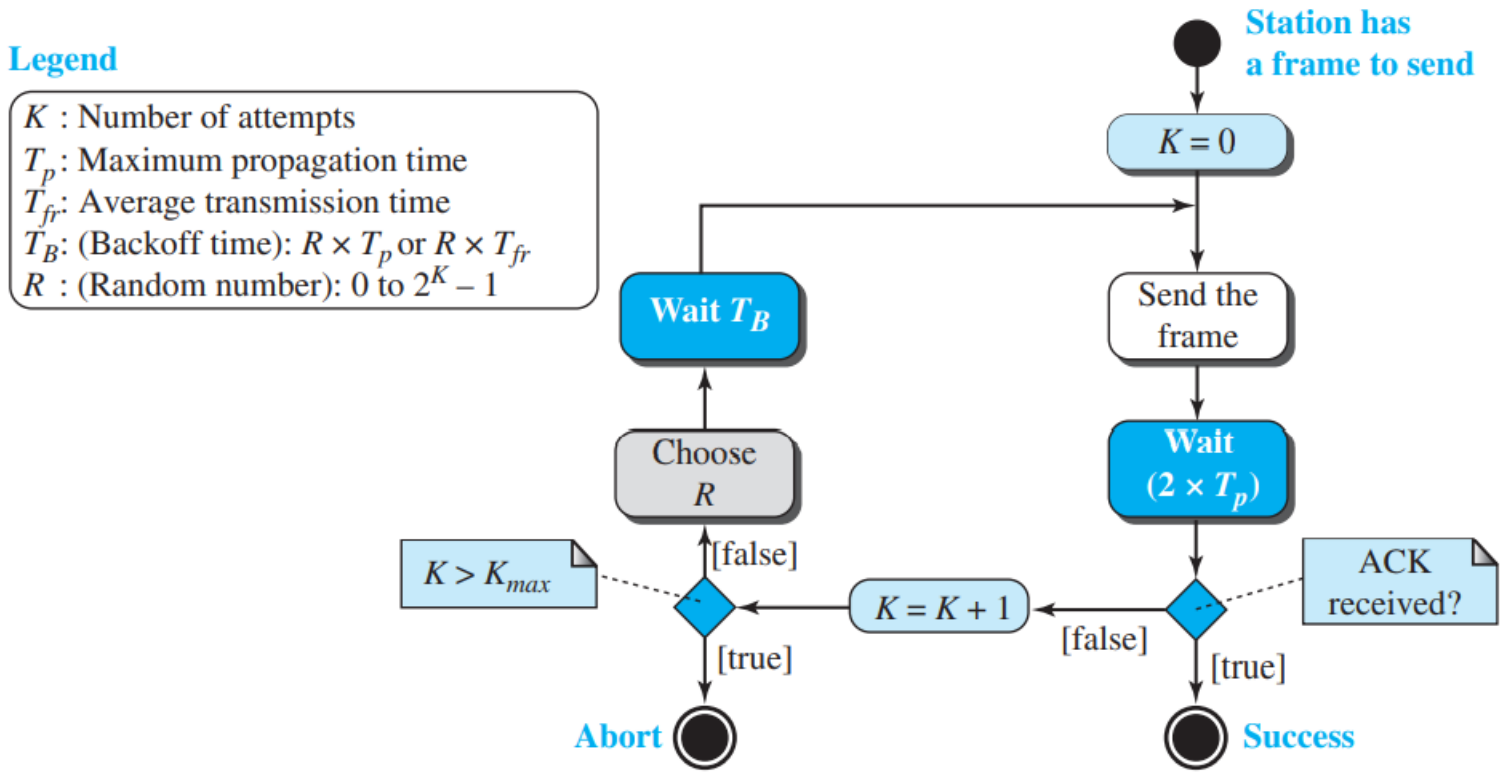
Figure 12.2 *Frames in a pure ALOHA network*



The pure ALOHA protocol relies on acknowledgments from the receiver.

If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.
the frames will collide again if all of them send after timeout
when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the backoff time T_B.
second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts Kmax, a station must give up and try later

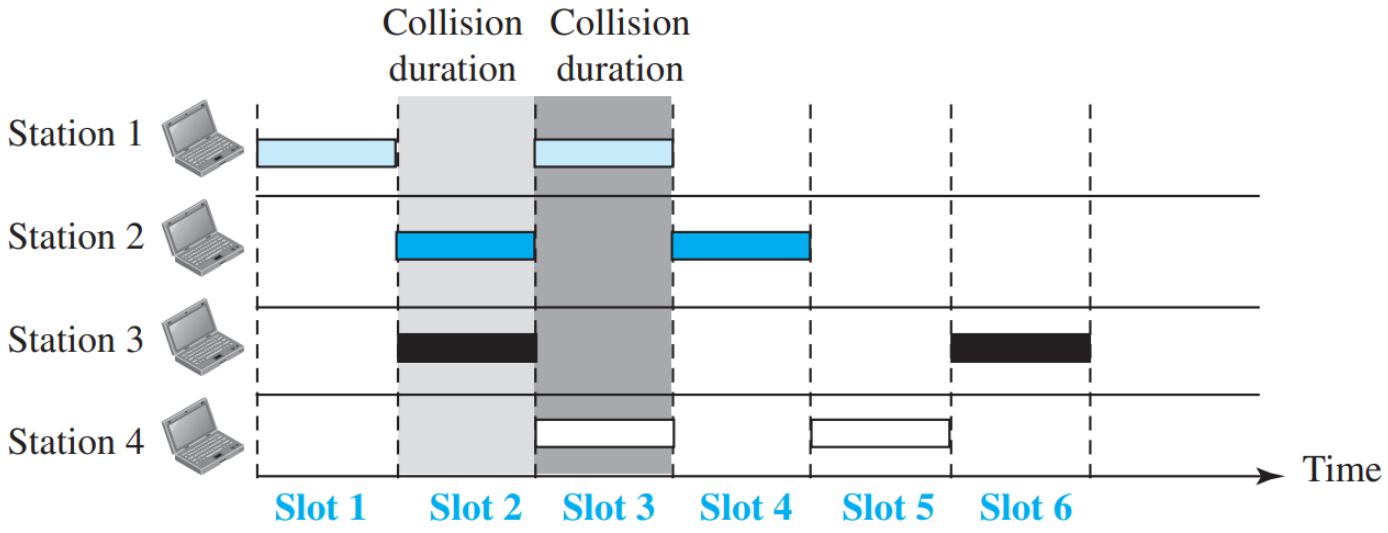
Figure 12.3 *Procedure for pure ALOHA protocol*



Slotted ALOHA:

We divide the time into slots of Tfr seconds and force the station to send only at the beginning of the time slot.

Figure 12.5 *Frames in a slotted ALOHA network*



CSMA

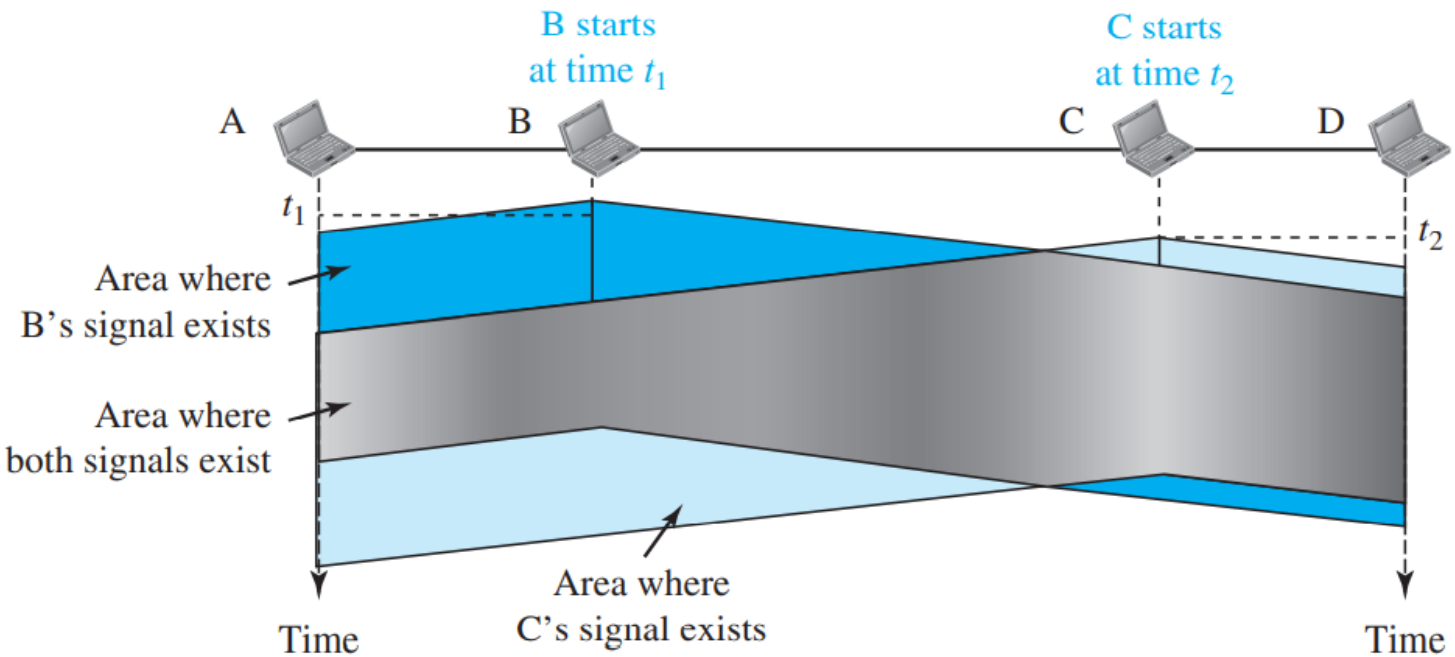
To minimize the chance of collision and, therefore, increase the performance

if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending

“sense before transmit” or “listen before talk.”

but it cannot eliminate it.

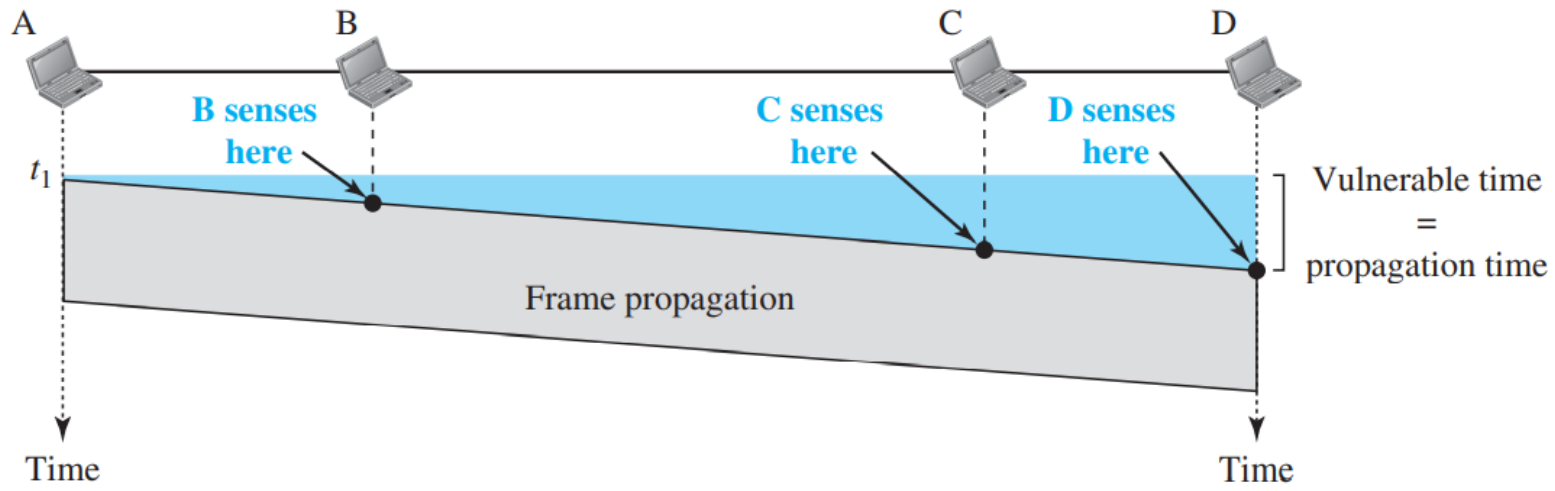
Figure 12.7 *Space/time model of a collision in CSMA*



The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station and for every station to sense it.

In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received. At time t1, station B senses the medium and finds it idle, so it sends a frame. At time t2 (t2 > t1), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed

Figure 12.8 *Vulnerable time in CSMA*



The vulnerable time for CSMA is the propagation time Tp. This is the time needed for a signal to propagate from one end of the medium to the other.

But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending

Persistence Methods

Figure 12.9 *Behavior of three persistence methods*

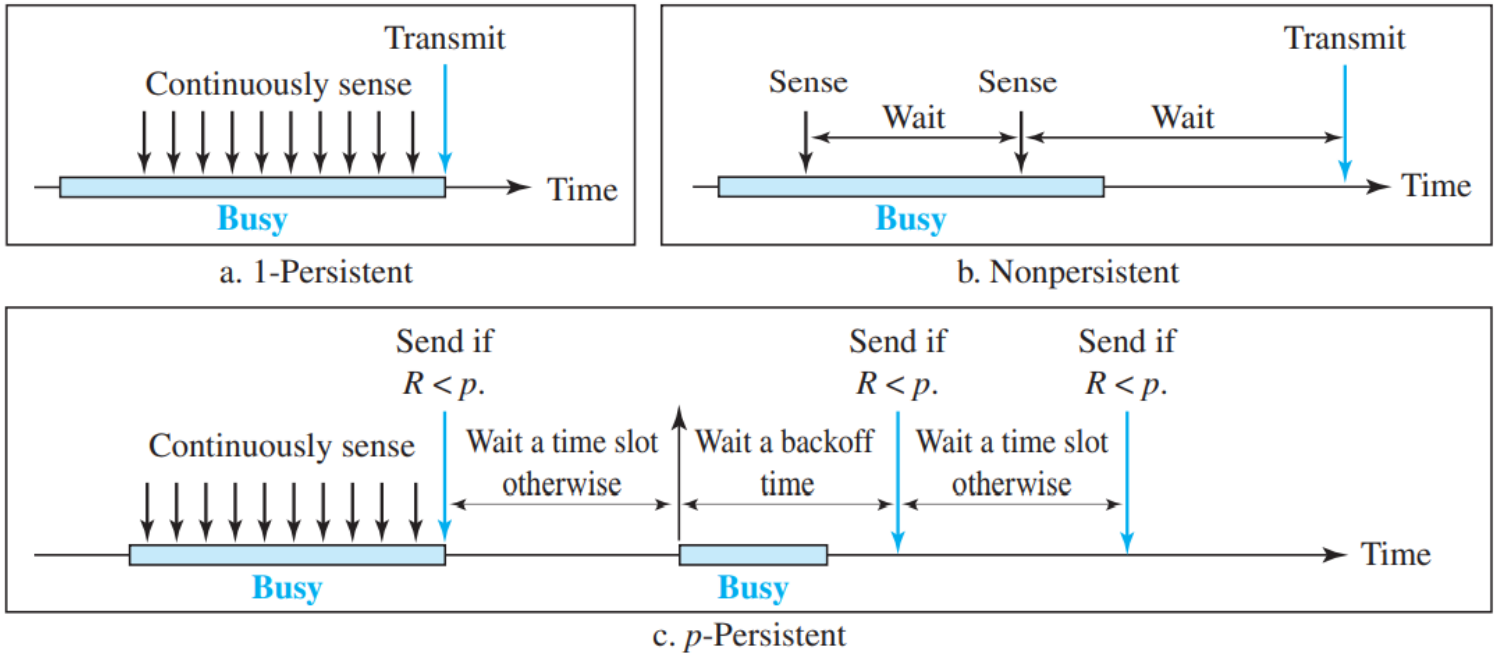
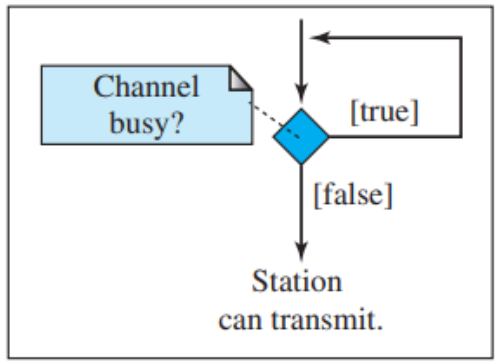
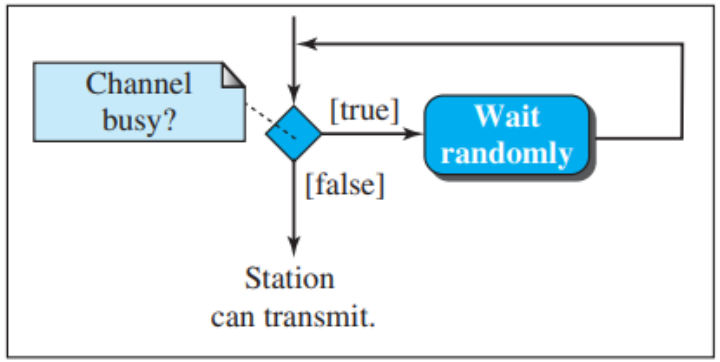


Figure 12.10 shows the flow diagrams for these methods.

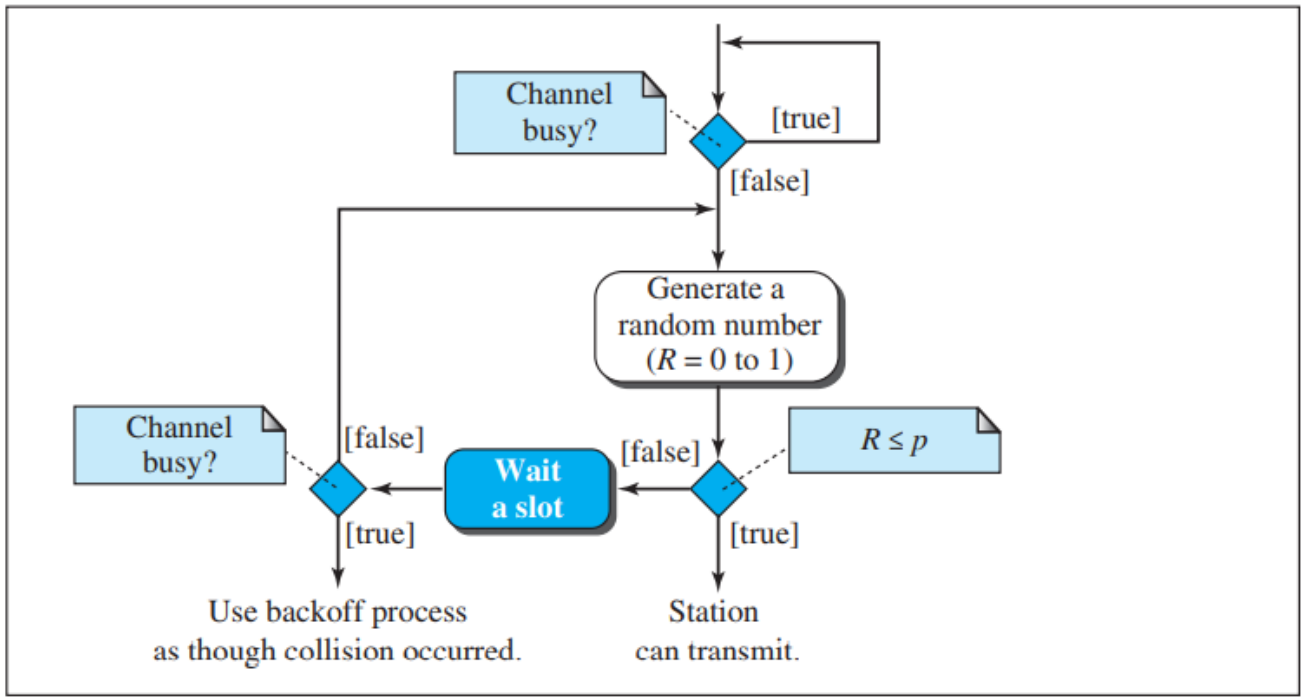
Figure 12.10 *Flow diagram for three persistence methods*



a. 1-Persistent



b. Nonpersistent



c. *p*-Persistent

In this method, after the station finds the line idle it follows these steps:

1.With probability p, the station sends its frame.

2. With probability q = 1 − p, the station waits for the beginning of the next time slot and checks the line again. a. If the line is idle, it goes to step 1. b. If the line is busy, it acts as though a collision has occurred and uses the backoff procedure.

Carrier sense multiple access with collision detection (CSMA/CD)

Figure 12.11 *Collision of the first bits in CSMA/CD*

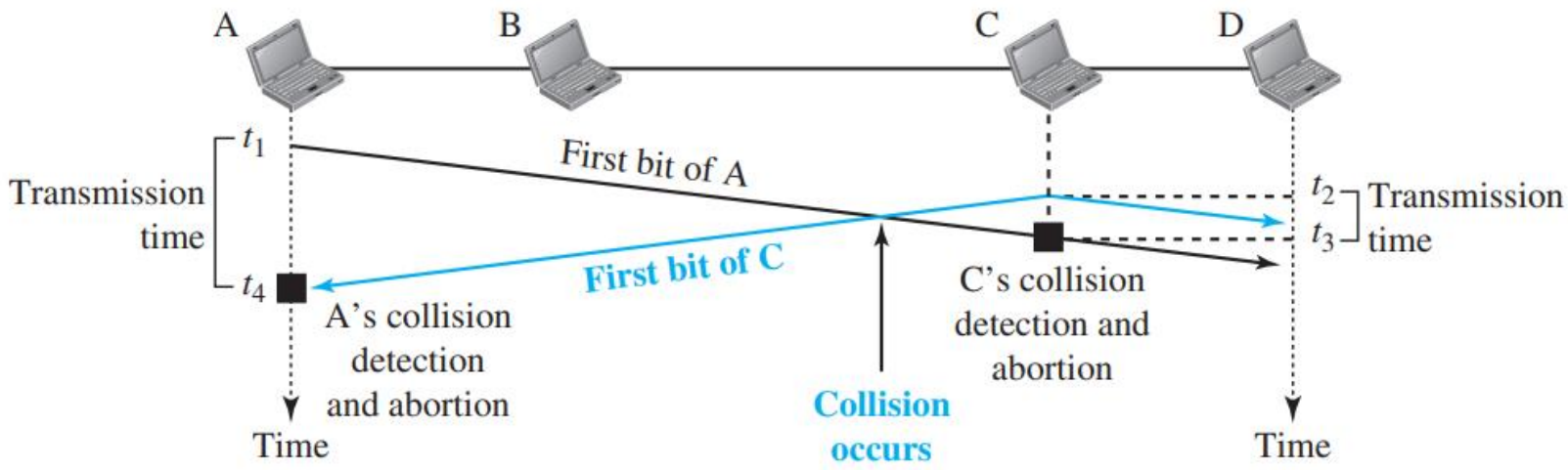


Figure 12.12 *Collision and abortion in CSMA/CD*

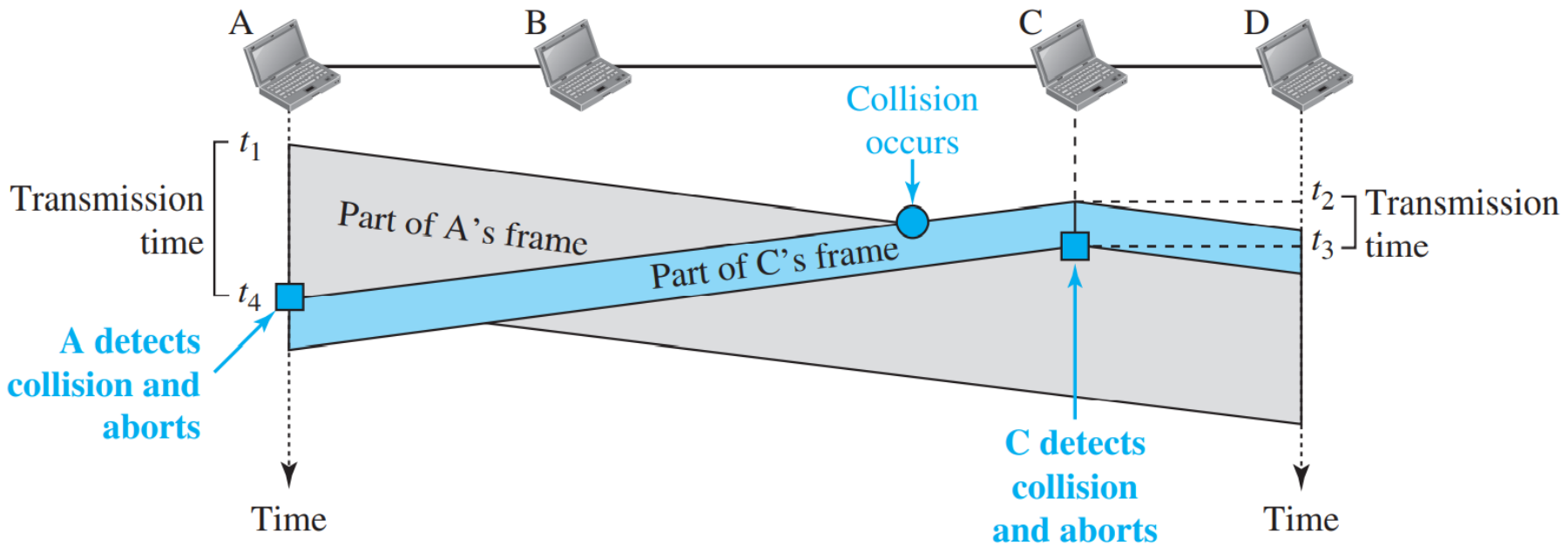
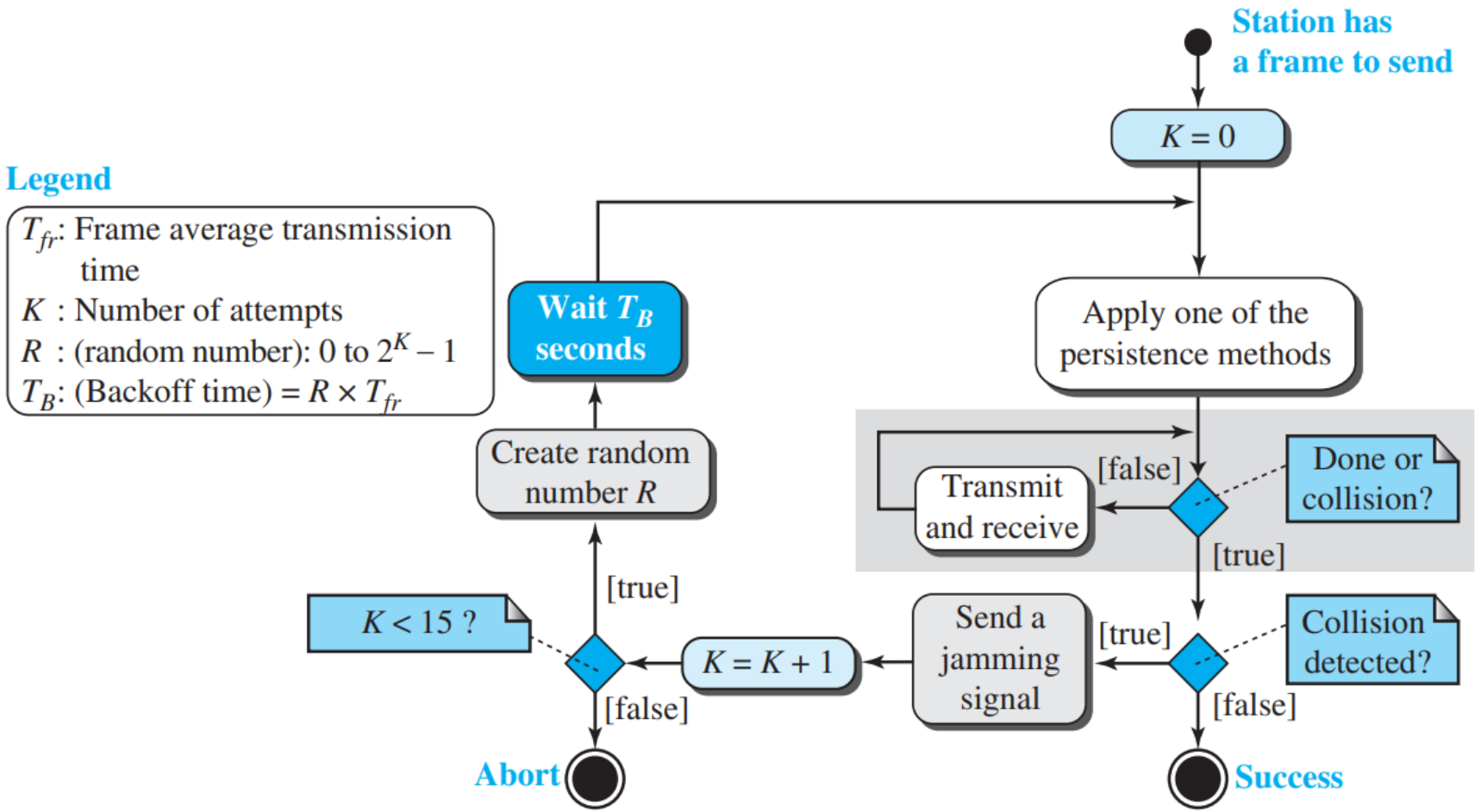


Figure 12.13 *Flow diagram for the CSMA/CD*



Differences from ALOHA protocol:

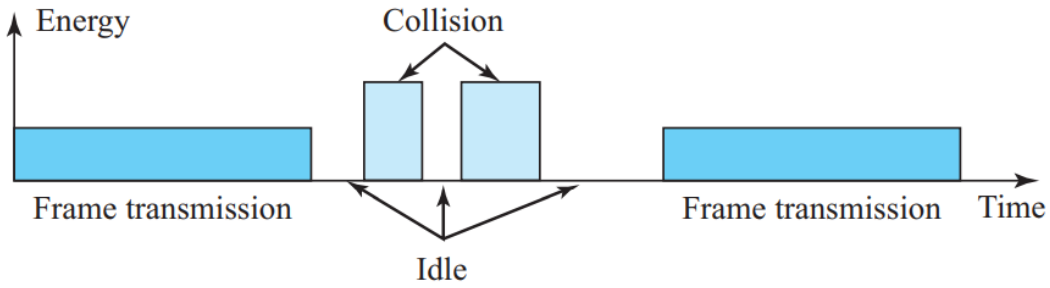
The first difference is the addition of the persistence process. We need to sense the channel before we start sending the frame by using one of the persistence processes we discussed previously (nonpersistent, 1-persistent, or p-persistent).

The second difference is the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In CSMA/CD, transmission and collision detection are continuous processes. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously (using two different ports or a bidirectional port). We use a loop to show that transmission is a continuous process. We constantly monitor in order to detect one of two conditions: either transmission is finished or a collision is detected. Either event stops transmission. When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred. The third difference is the sending of a short jamming signal to make sure that all other stations become aware of the collision

Energy Level :

The level of energy in a channel can have three values: zero, normal, and abnormal. At the zero level, the channel is idle. At the normal level, a station has successfully captured the channel and is sending its frame. At the abnormal level, there is a collision and the level of the energy is twice the normal level. A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, busy, or in collision mode.

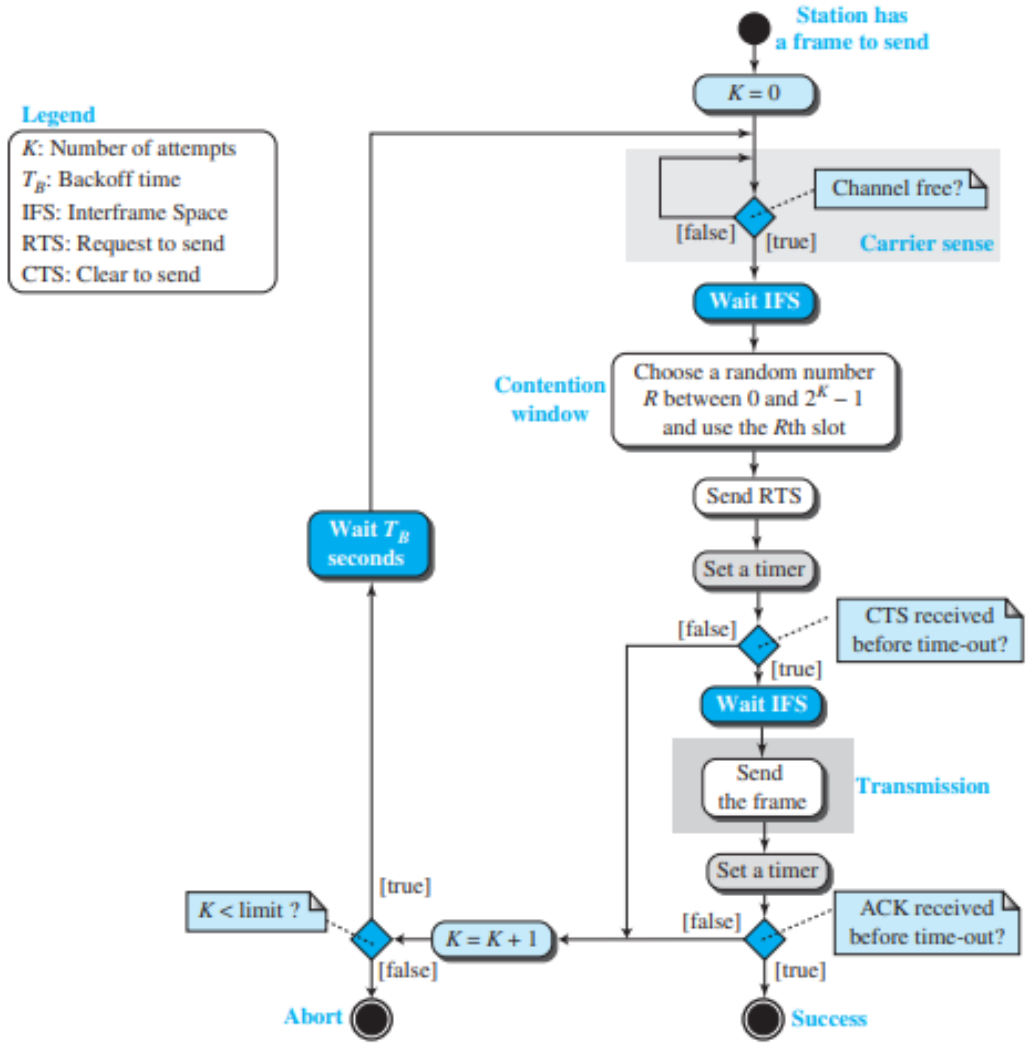
Figure 12.14 *Energy level during transmission, idleness, or collision*



12.1.4 CSMA/CA

Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for wireless networks. Collisions are avoided through the use of CSMA/CA's three strategies: the interframe space, the contention window, and acknowledgments, as shown in Figure 12.15. We discuss RTS and CTS frames later.

Figure 12.15 *Flow diagram of CSMA/CA*



Interframe Space (IFS).

even if the channel is found idle, transmissions deferred.

It waits for a period of time called the interframe space or IFS.

Reason: A distant station may have already started transmitting The IFS time allows the front of the transmitted signal by the distant station to reach this station. After waiting an IFS time, if the channel is still idle, the station can send, but it still needs to wait a time equal to the contention window (described next). The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned a shorter IFS has a higher priority.

❑ **Contention Window.**

an amount of time divided into slots.

A station that is ready to send chooses a random number of slots (binary exponential backoff strategy-set to one slot the first time and then doubles each time the station cannot detect an idle channel)after the IFS time.

similar to the p-persistent method except that a random outcome defines the number of slots taken by the waiting station.

One interesting point about the contention window is that the station needs to sense the channel after each time slot.

if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time.

Hidden-Station Problem

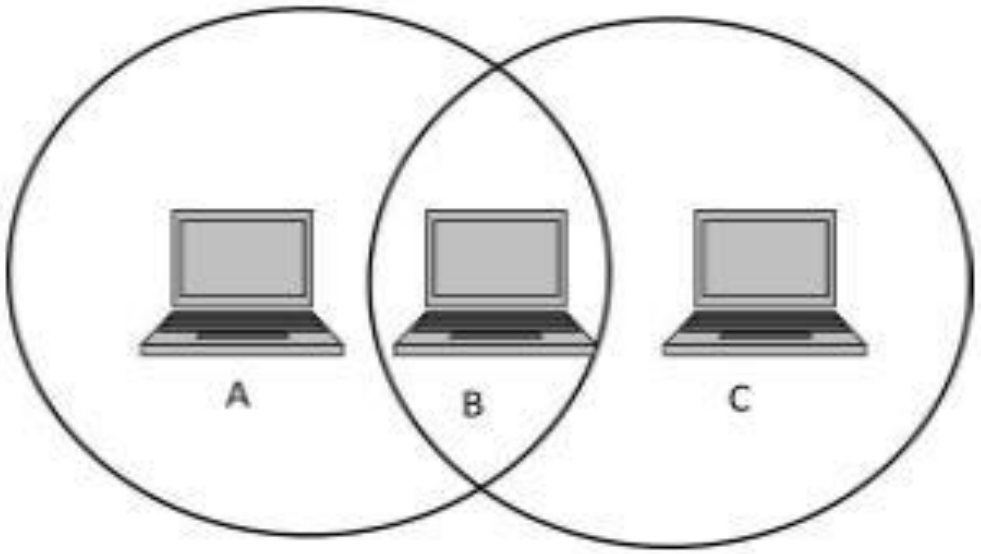
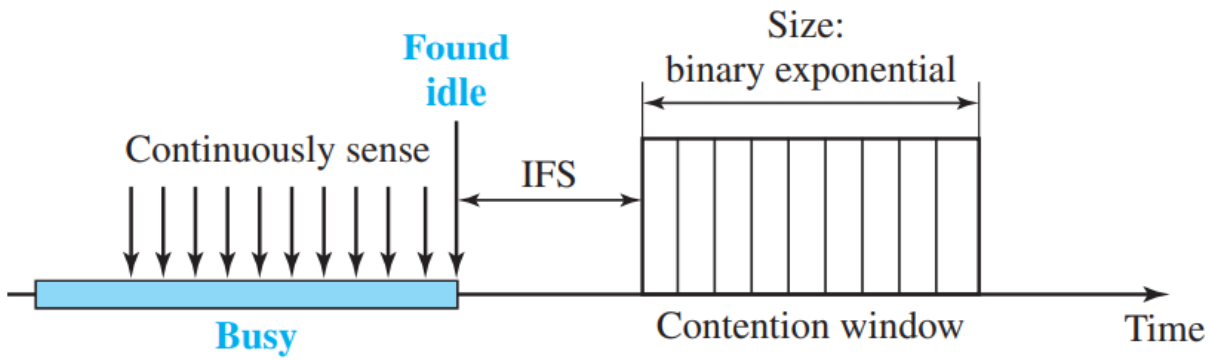


Figure 12.16 *Contention window*



Acknowledgment.

With all these precautions, there still may be a collision resulting in destroyed data/data corruption during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

Frame Exchange Time Line

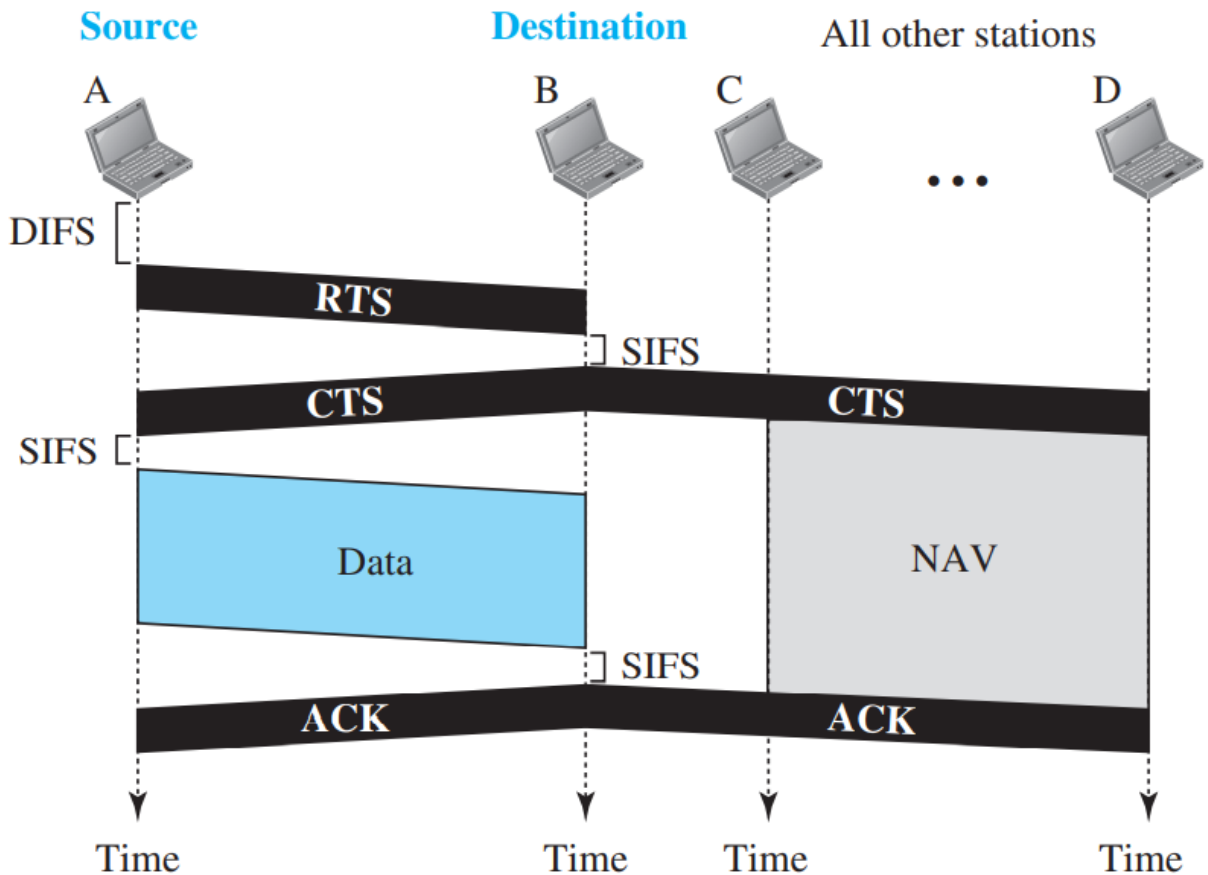
1. Before sending a frame, the source station senses the medium by checking the energy level at the carrier frequency.

a. The channel uses a persistence strategy with backoff until the channel is idle.

b. After the station is found to be idle, the station waits for a period of time called the DCF interframe space (DIFS); then the station sends a control frame called the request to send (RTS).

2. After receiving the RTS and waiting a period of time called the short interframe space (SIFS), the destination station sends a control frame, called the clear to send (CTS), to the source station. This control frame indicates that the destination station is ready to receive data.

Figure 12.17 *CSMA/CA and NAV*



Network Allocation Vector:

- RTS attaches the time required for its comm
- When other stations listen to this they start their NAV and set their timers when they listen to CTS.
- After the timer goes down to 0, they check the channel again

Collision During Handshaking

If two RTS frames collide, CTS is absent. So others as well source get to know of the collision

Backoff strategy applied and tried again

The throughput is a measure of how fast we can actually send data through a network.

Bandwidth is a potential measurement of a link; Throughput is an actual measurement of how fast we can send data.

For example, we may have a link with a bandwidth of 1 Mbps, but the devices connected to the end of the link may handle only 200 kbps. This means that we cannot send more than 200 kbps through this link.

Imagine a highway designed to transmit 1000 cars per minute from one point to another. However, if there is congestion on the road, this figure may be reduced to 100 cars per minute. The bandwidth is 1000 cars per minute; the throughput is 100 cars per minute.

Latency (Delay):

$$\text{Latency} = \text{propagation time} + \text{transmission time} + \text{queuing time} + \text{processing delay}$$

Propagation time : Distance / (Propagation Speed)

Transmission time = (Message size) / Bandwidth

The bandwidth-delay product defines the number of bits that can fill the link.

Example 3.44

A network with bandwidth of 10 Mbps can pass only an average of 12,000 frames per minute with each frame carrying an average of 10,000 bits. What is the throughput of this network?

Solution

We can calculate the throughput as

$$\textbf{Throughput} = (12{,}000 \times 10{,}000) / 60 = 2 \textbf{ Mbps}$$

The throughput is almost one-fifth of the bandwidth in this case.

Example 12.1

The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at 3×10^8 m/s, we find $T_p = (600 \times 10^3) / (3 \times 10^8) = 2$ ms. For $K = 2$, the range of R is $\{0, 1, 2, 3\}$. This means that T_B can be 0, 2, 4, or 6 ms, based on the outcome of the random variable R.

Vulnerable time- Pure Aloha:

Let us find the vulnerable time, the length of time in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking Tfr seconds to send.

we see that the vulnerable time during which a collision may occur in pure ALOHA is 2 times the frame transmission time.

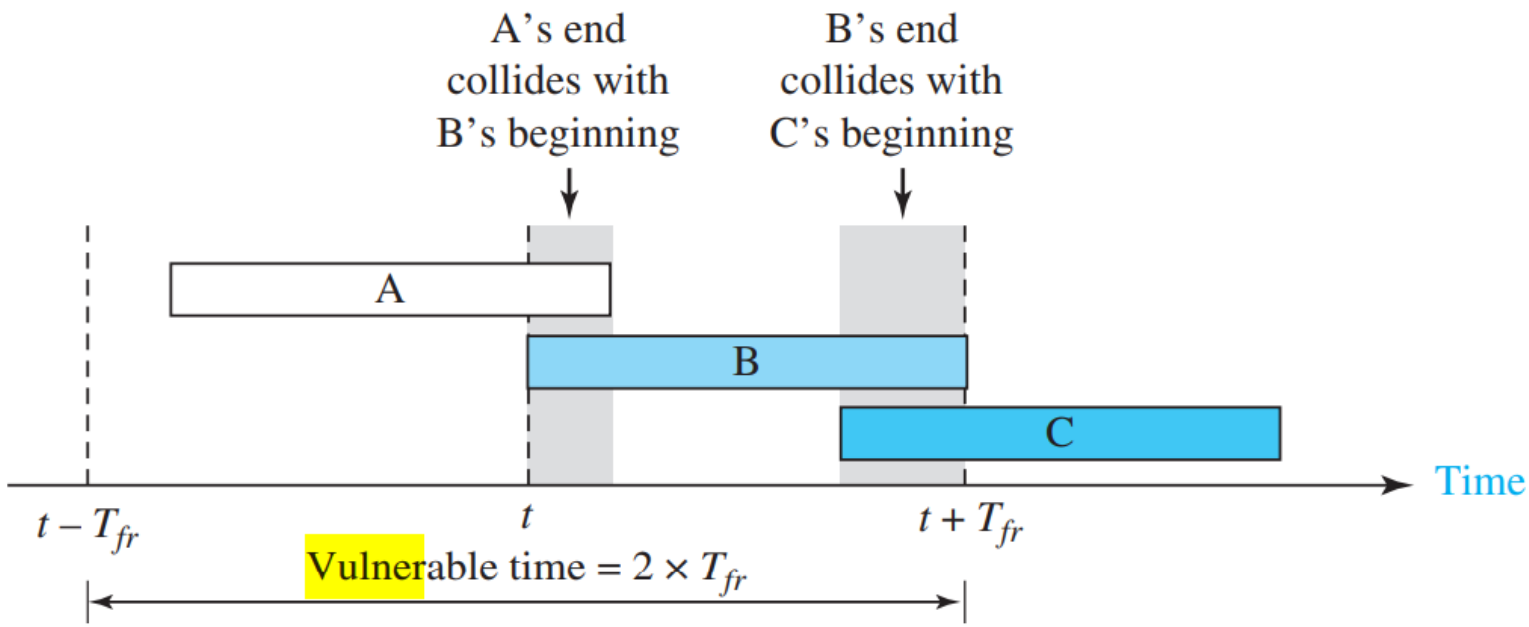
Pure ALOHA vulnerable time = 2 X Tfr

Example 12.2

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?

Solution: Average frame transmission time Tfr is 200 bits/200 kbps or 1 ms. The vulnerable time is 2 × 1 ms = 2 ms. This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the period (1 ms) that this station is sending

Figure 12.4 *Vulnerable time for pure ALOHA protocol*



Throughput :

Let G the average number of frames generated by the system during one frame transmission time.

Then it can be proven that the average number of successfully transmitted frames for pure ALOHA is $S = G \times e^{-2G}$.

The maximum throughput Smax is 0.184, for G = 1/2.

In other words, if one-half a frame is generated during one frame transmission time (one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully.

We expect G = 1/2 to produce the maximum throughput because the vulnerable time is 2 times the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other stations generate a frame during this time), the frame will reach its destination successfully.

The throughput for pure ALOHA is $S = G \times e^{-2G}$.
The maximum throughput $S_{max} = 1/(2e) = 0.184$ when $G = (1/2)$.

Example 12.3

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces

a. 1000 frames per second?
b. 500 frames per second?
c. 250 frames per second?

Solution

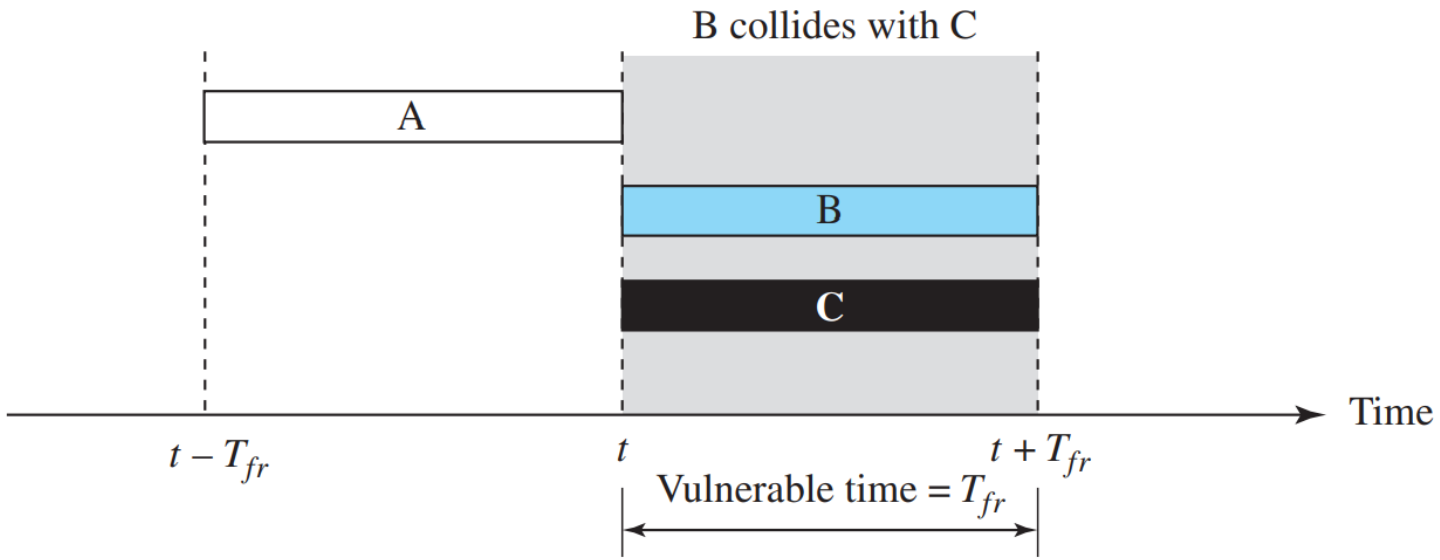
The frame transmission time is 200/200 kbps or 1 ms.

a. If the system creates 1000 frames per second, or 1 frame per millisecond, then $G = 1$. In this case $S = G \times e^{-2G} = 0.135$ (13.5 percent). This means that the throughput is $1000 \times 0.135 = 135$ frames. Only 135 frames out of 1000 will probably survive.

b. If the system creates 500 frames per second, or 1/2 frames per millisecond, then $G = 1/2$. In this case $S = G \times e^{-2G} = 0.184$ (18.4 percent). This means that the throughput is $500 \times 0.184 = 92$ and that only 92 frames out of 500 will probably survive. Note that this is the *maximum* throughput case, percentagewise.

c. If the system creates 250 frames per second, or 1/4 frames per millisecond, then $G = 1/4$. In this case $S = G \times e^{-2G} = 0.152$ (15.2 percent). This means that the throughput is $250 \times 0.152 = 38$. Only 38 frames out of 250 will probably survive.

Figure 12.6 *Vulnerable time for slotted ALOHA protocol*



The throughput for slotted ALOHA is $S = G \times e^{-G}$.
The maximum throughput $S_{max} = 0.368$ when $G = 1$.

Example 12.4

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces

a. 1000 frames per second.

b. 500 frames per second.

c. 250 frames per second.

Solution

This situation is similar to the previous exercise except that the network is using slotted ALOHA instead of pure ALOHA. The frame transmission time is 200/200 kbps or 1 ms.

a. In this case G is 1. So $S = G \times e^{-G} = 0.368$ (36.8 percent). This means that the throughput is $1000 \times 0.0368 = 368$ frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.

b. Here G is 1/2. In this case $S = G \times e^{-G} = 0.303$ (30.3 percent). This means that the throughput is $500 \times 0.0303 = 151$. Only 151 frames out of 500 will probably survive.

c. Now G is 1/4. In this case $S = G \times e^{-G} = 0.195$ (19.5 percent). This means that the throughput is $250 \times 0.195 = 49$. Only 49 frames out of 250 will probably survive.

Minimum Frame Size

For CSMA/CD to work, we need a restriction on the frame size.
once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the frame transmission time Tfr must be at least two times the maximum propagation time Tp. To understand the reason, let us think about the worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time Tp to reach the second, and the effect of the collision takes another time TP to reach the first. So the requirement is that the first station must still be transmitting after 2Tp.

Example 12.5

A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6 μs, what is the minimum size of the frame?

Solution

The minimum frame transmission time is $T_{fr} = 2 \times T_p = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet, as we will see later in the chapter.

Throughput:

The throughput of CSMA/CD is greater than that of pure or slotted ALOHA. The maximum throughput occurs at a different value of G and is based on the persistence method and the value of p in the p-persistent approach. For the 1-persistent method, the maximum throughput is around 50 percent when G = 1. For the nonpersistent method, the maximum throughput can go up to 90 percent when G is between 3 and 8.

Traditional Ethernet One of the LAN protocols that used CSMA/CD is the traditional Ethernet with the data rate of 10 Mbps.

12-2 CONTROLLED ACCESS

In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss three popular controlled-access methods.

Topics discussed in this section:

Reservation
Polling
Token Passing

Figure 12.18 *Reservation access method*

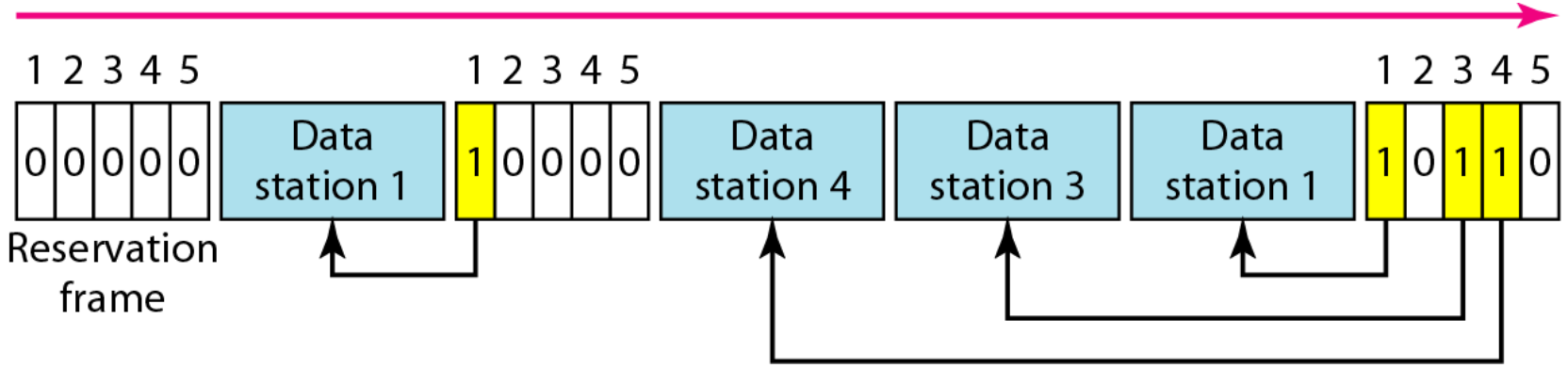


Figure 12.19 *Select and poll functions in polling access method*

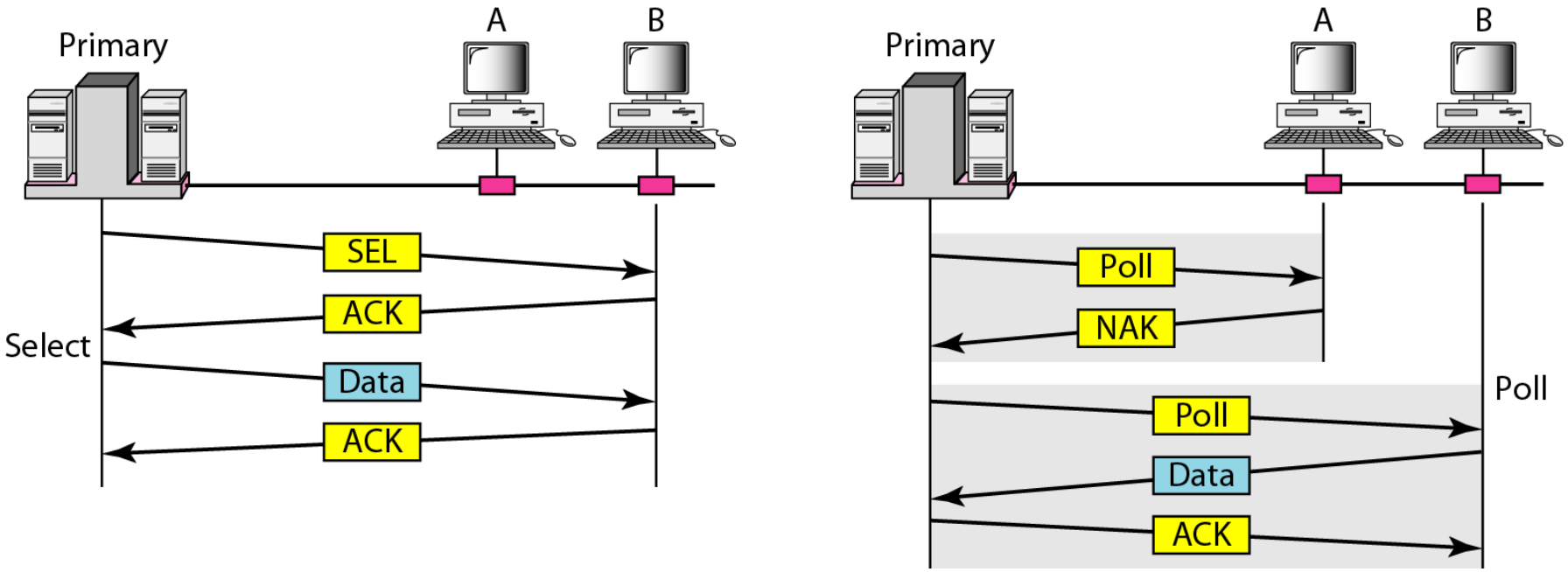
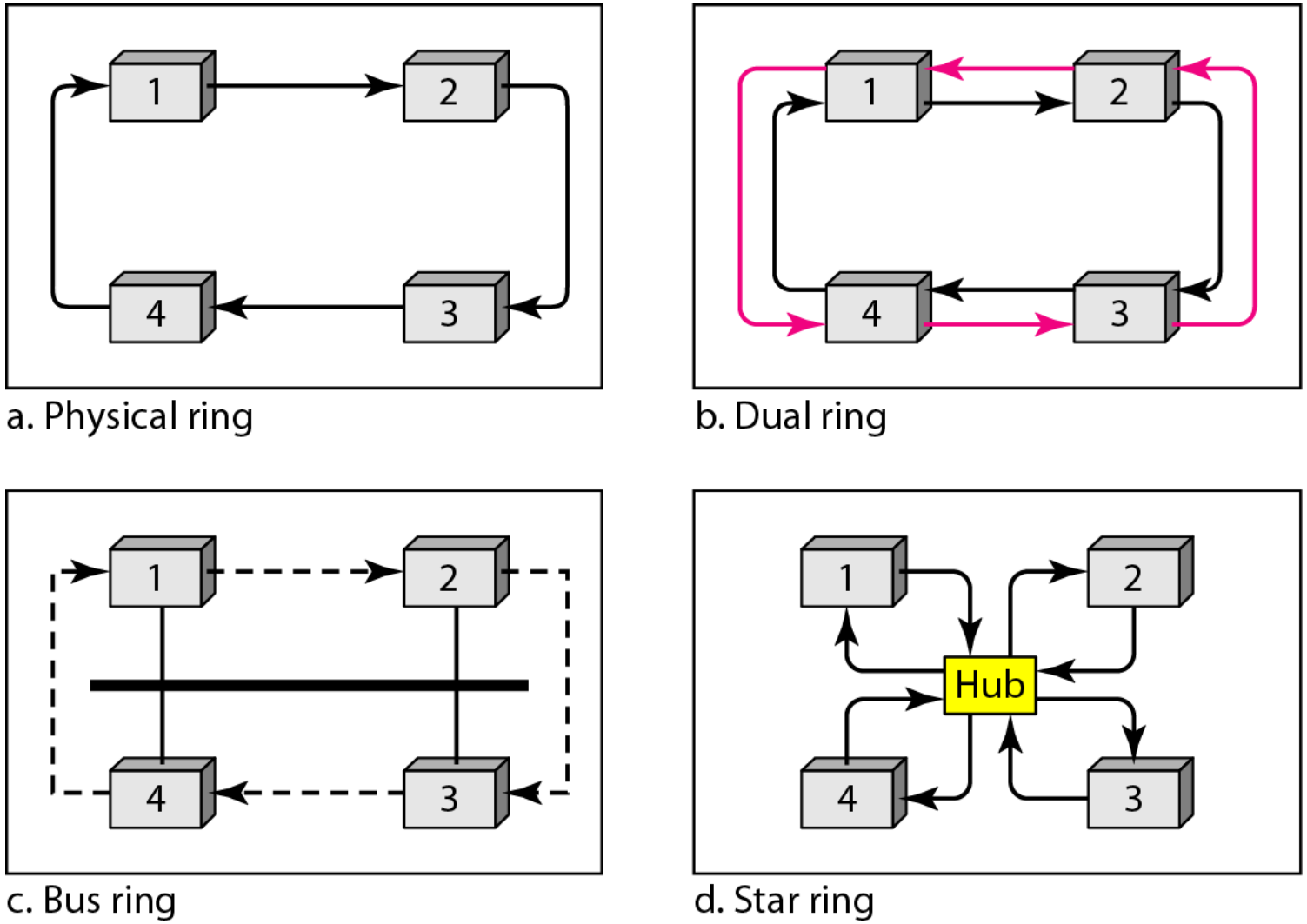


Figure 12.20 *Logical ring and physical topology in token-passing access method*



Token management is needed for this access method.

Stations must be limited in the time they can have possession of the token.

The token must be monitored to ensure it has not been lost or destroyed.

For example, if a station that is holding the token fails, the token will disappear from the network.

Another function of token management is to assign priorities to the stations and to the types of data being transmitted.

And finally, token management is needed to make low-priority stations release the token to high-priority stations.

CHANNELIZATION

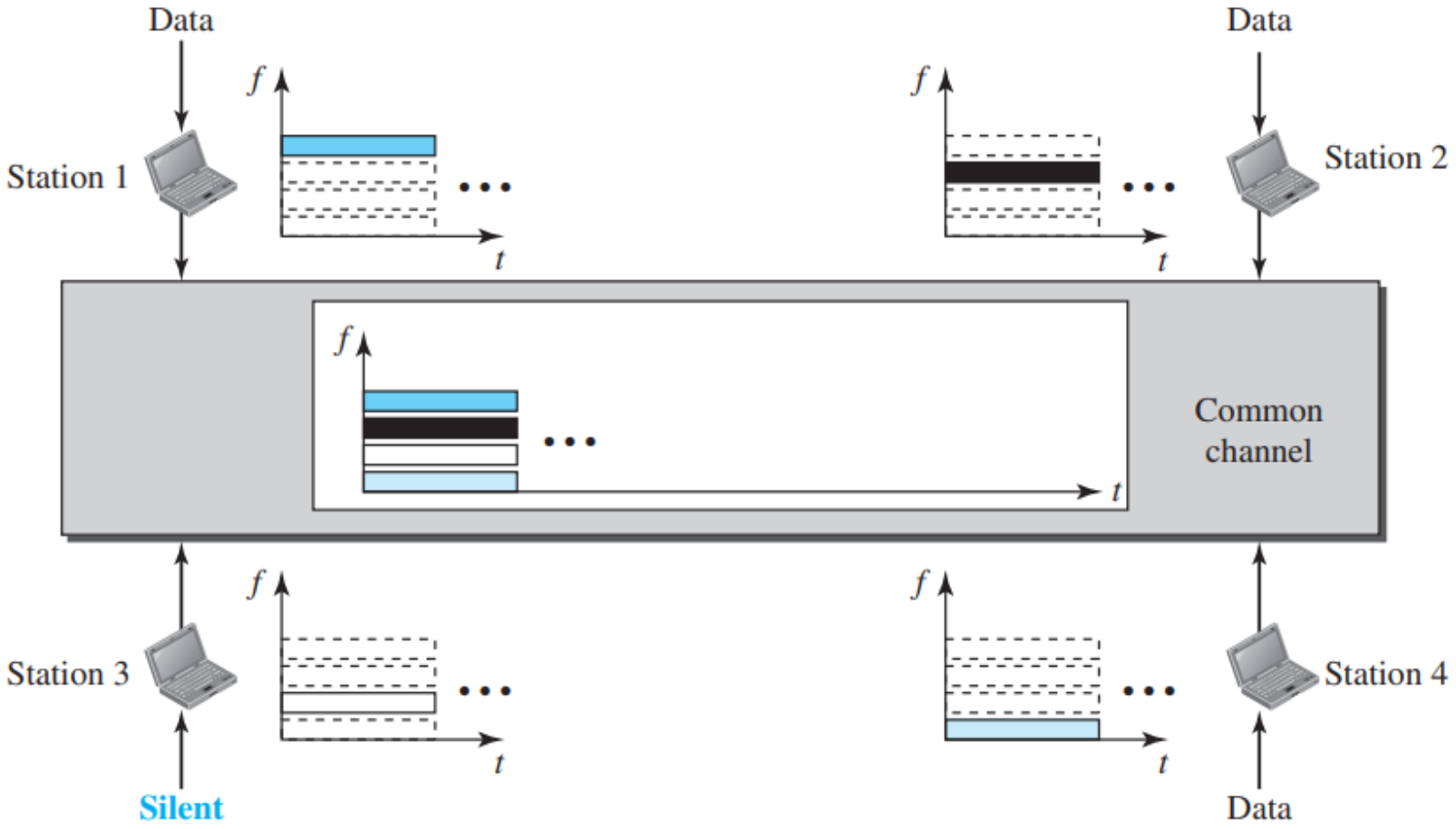
Multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations.

Three methods:

- **FDMA**
- **TDMA**
- **CDMA**

- FDMA

Figure 12.21 *Frequency-division multiple access (FDMA)*

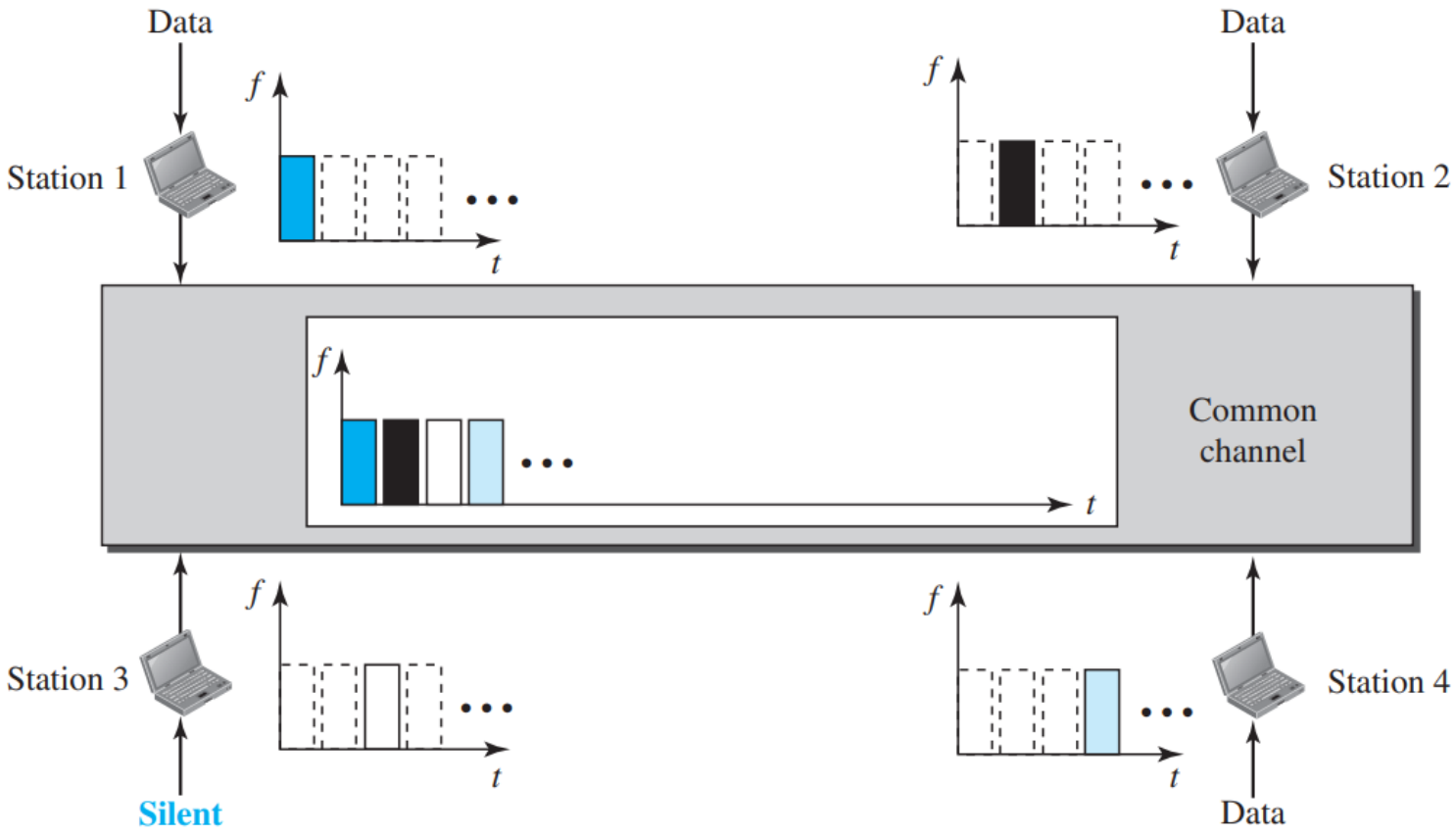


- FDM
  - is a physical layer technique that combines the loads from low bandwidth channels and transmits them by using a high-bandwidth channel.
  - The channels that are combined are low-pass.
  - The multiplexer modulates the signals, combines them, and creates a bandpass signal. The bandwidth of each channel is shifted by the multiplexer.

- FDMA
  - is an access method in the data-link layer.
  - The datalink layer in each station tells its physical layer to make a bandpass signal from the data passed to it.
  - The signal must be created in the allocated band. There is no physical multiplexer at the physical layer.
  - The signals created at each station are automatically bandpass-filtered. They are mixed when they are sent to the common channel.

- TDMA

Figure 12.22 *Time-division multiple access (TDMA)*



- The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot.
- To compensate for the delays, we can insert guard times.

- Diff – same as FDMA

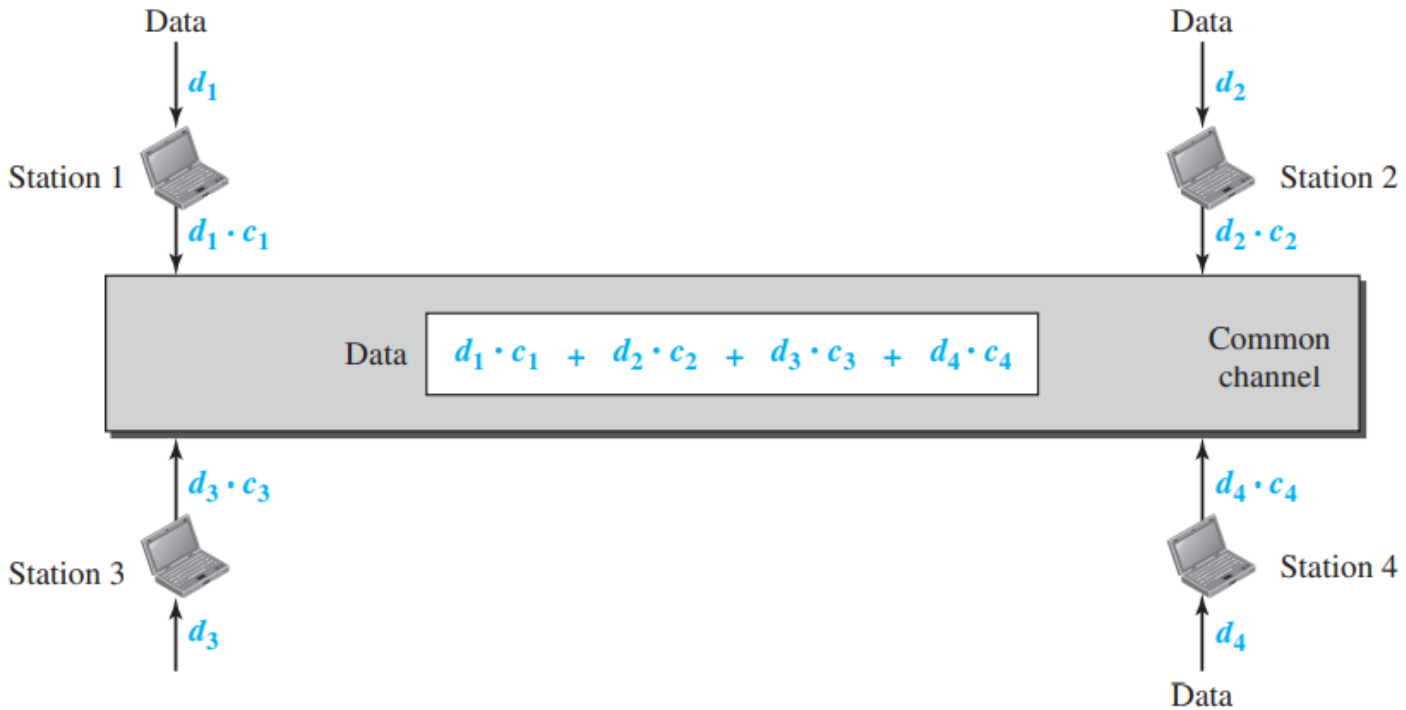
- Code-division multiple access (CDMA)
  - CDMA differs from FDMA in that only one channel occupies the entire bandwidth of the link. It differs from TDMA in that all stations can send data simultaneously; there is no timesharing.
  - CDMA simply means communication with different codes.
  - Example: People in a room talking with their other peer in a different language- say French, latin etc.

- Idea :
- Let us assume we have four stations, 1, 2, 3, and 4, connected to the same channel. The data from station 1 are d1, from station 2 are d2, and so on. The code assigned to the first station is c1, to the second is c2, and so on.

- We assume that the assigned codes have two properties.
- 1. If we multiply each code by another, we get 0. 2. If we multiply each code by itself, we get 4 (the number of stations).
- Ie., c1 X c2 =0 c1 x c1 =4

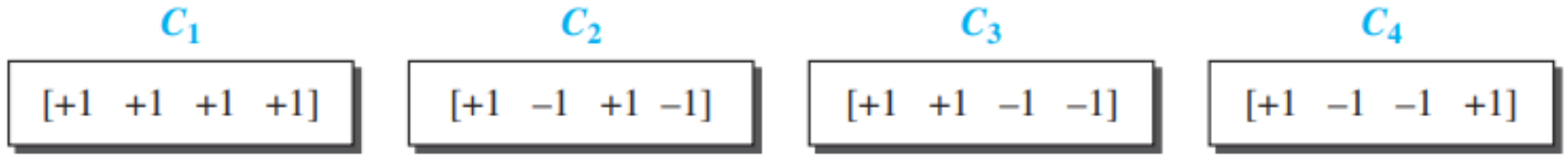
- Station 1 multiplies (a special kind of multiplication, as we will see) its data by its code to get $d1 \cdot c1$.
- Station 2 multiplies its data by its code to get $d2 \cdot c2$, and so on.
- to receive data from one of the other three multiplies the data on the channel by the code of the sender.
- Chips
  - CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called chips

Figure 12.23 *Simple idea of communication with code*



$$\begin{aligned}\text{data} &= (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1 \\ &= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 = 4 \times d_1\end{aligned}$$

Figure 12.24 *Chip sequences*



1. Each sequence is made of N elements, where N is the number of stations.
2. If we multiply a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar. For example,

$$2 \cdot [+1\ +1\ -1\ -1] = [+2\ +2\ -2\ -2]$$

3. If we multiply two equal sequences, element by element, and add the results, we get N, where N is the number of elements in each sequence. This is called the ***inner product*** of two equal sequences. For example,

$$[+1\ +1\ -1\ -1] \cdot [+1\ +1\ -1\ -1] = 1 + 1 + 1 + 1 = 4$$

4. If we multiply two different sequences, element by element, and add the results, we get 0. This is called the *inner product* of two different sequences. For example,

$$[+1\ +1\ -1\ -1] \cdot [+1\ +1\ +1\ +1] = 1 + 1 - 1 - 1 = 0$$

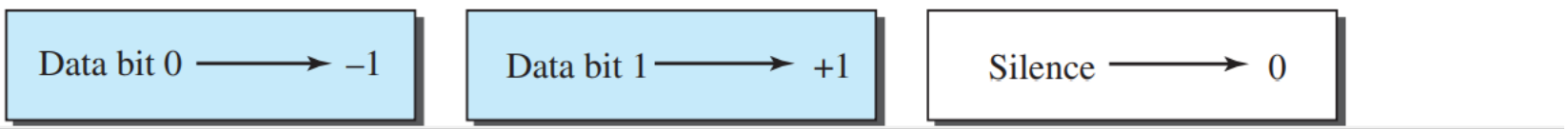
5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

$$[+1\ +1\ -1\ -1] + [+1\ +1\ +1\ +1] = [+2\ +2\ \ 0\ \ 0]$$

Data Representation

We follow these rules for encoding: If a station needs to send a 0 bit, it encodes it as –1; if it needs to send a 1 bit, it encodes it as +1. When a station is idle, it sends no signal, which is interpreted as a 0. These are shown in Figure 12.25.

Figure 12.25 *Data representation in CDMA*



- We assume that stations
  - 1 and 2 are sending a 0 bit and
  - 4 is sending a 1 bit.
  - Station 3 is silent.
  - The data at the sender site are translated to −1, −1, 0, and +1.

Encoding and Decoding

$$[-1\ -1\ -3\ +1] \bullet [+1\ -1\ +1\ -1] = -4/4 = -1 \quad \rightarrow \quad \textbf{bit 1}$$

Signal Level

Figure 12.26 *Sharing channel in CDMA*

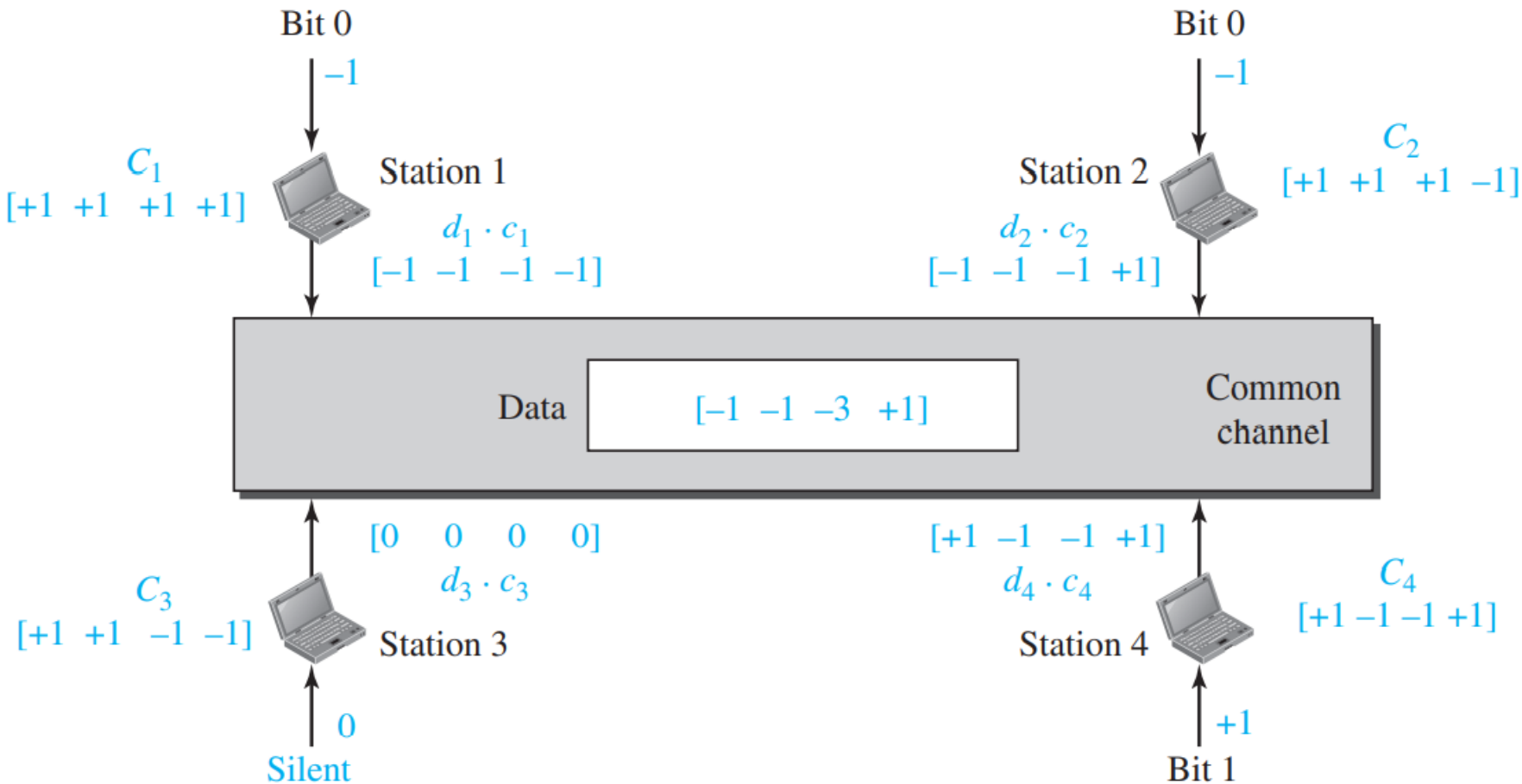


Figure 12.27 *Digital signal created by four stations in CDMA*

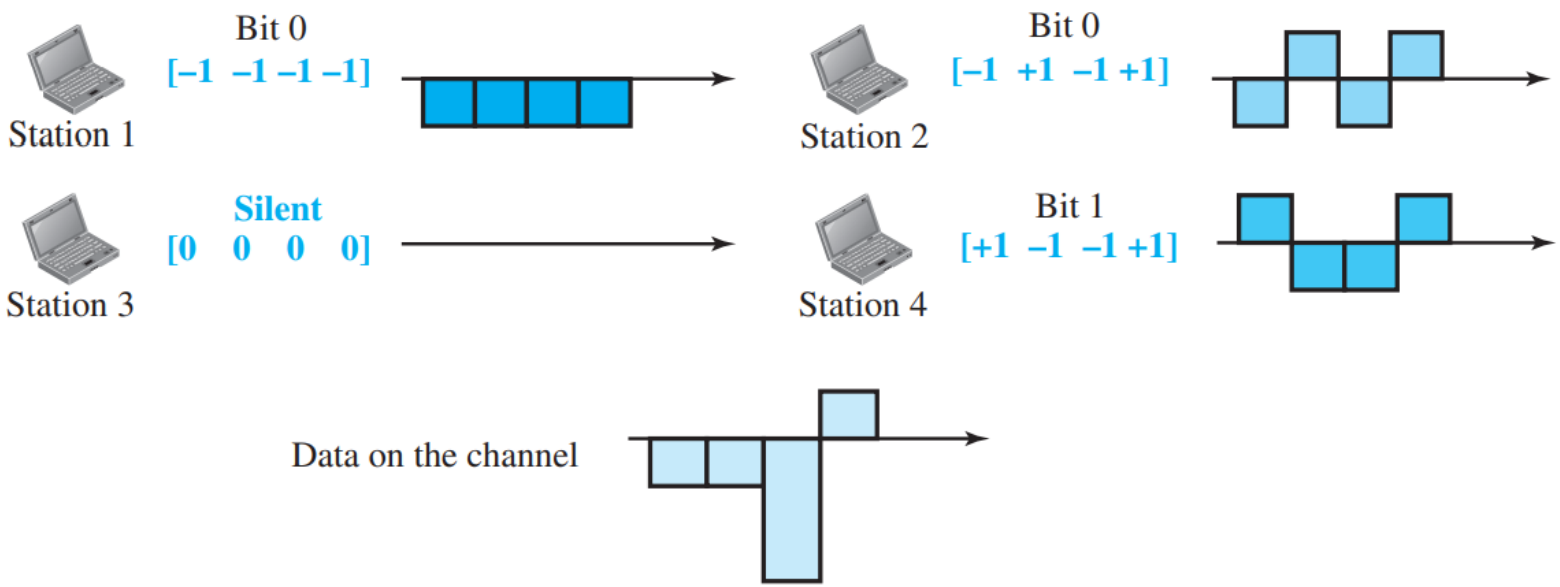
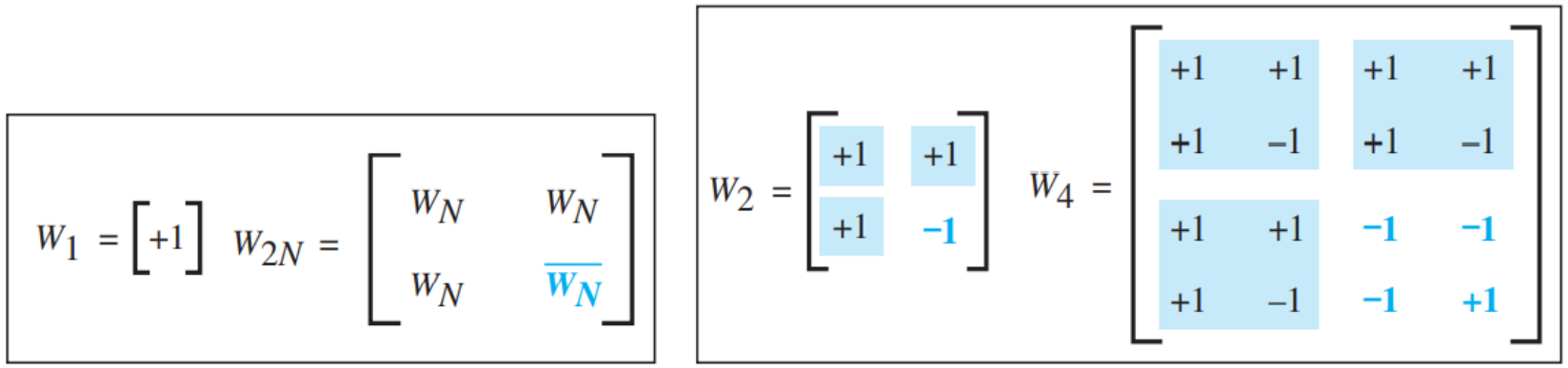


Figure 12.29 *General rule and examples of creating Walsh tables*

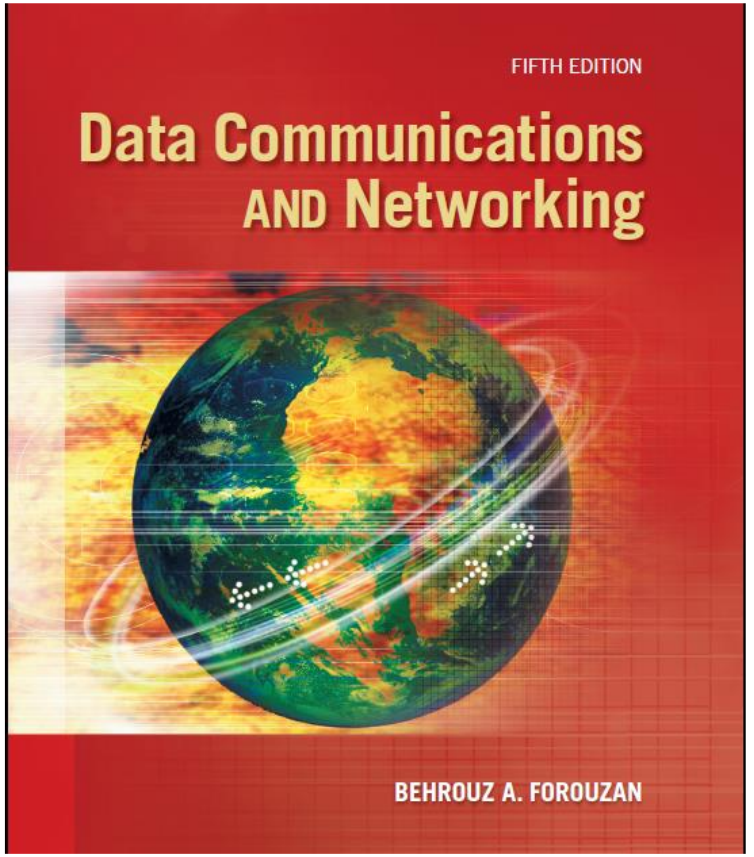


a. Two basic rules

b. Generation of W_2 and W_4

The number of sequences in a Walsh table needs to be 2^m.

TOPICS AFTER THIS SLIDE ARE NOT INCLUDED FOR TEST 2 BUT INCLUDED FOR SEE



Chapter 13

Wired LANs: Ethernet

13–1 IEEE STANDARDS

In 1985, the Computer Society of the IEEE started a project, called project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers.. Project 802 is a way of specifying functions of the physical layer and the data link layer of major LAN protocols.

Topics discussed in this section:

Data Link Layer
Physical Layer

Ethernet

Ethernet protocol was designed so that it could evolve with the demand for higher transmission rates.

In the 1980s and 1990s several different types of LANs were used. All of these LANs used a media-access method to solve the problem of sharing the media.

The Ethernet used the CSMA/CD approach.
The Token Ring, Token Bus, and FDDI (Fiber Distribution Data Interface) used the token-passing approach. ATM LAN- deployed the high speed WAN technology (ATM)

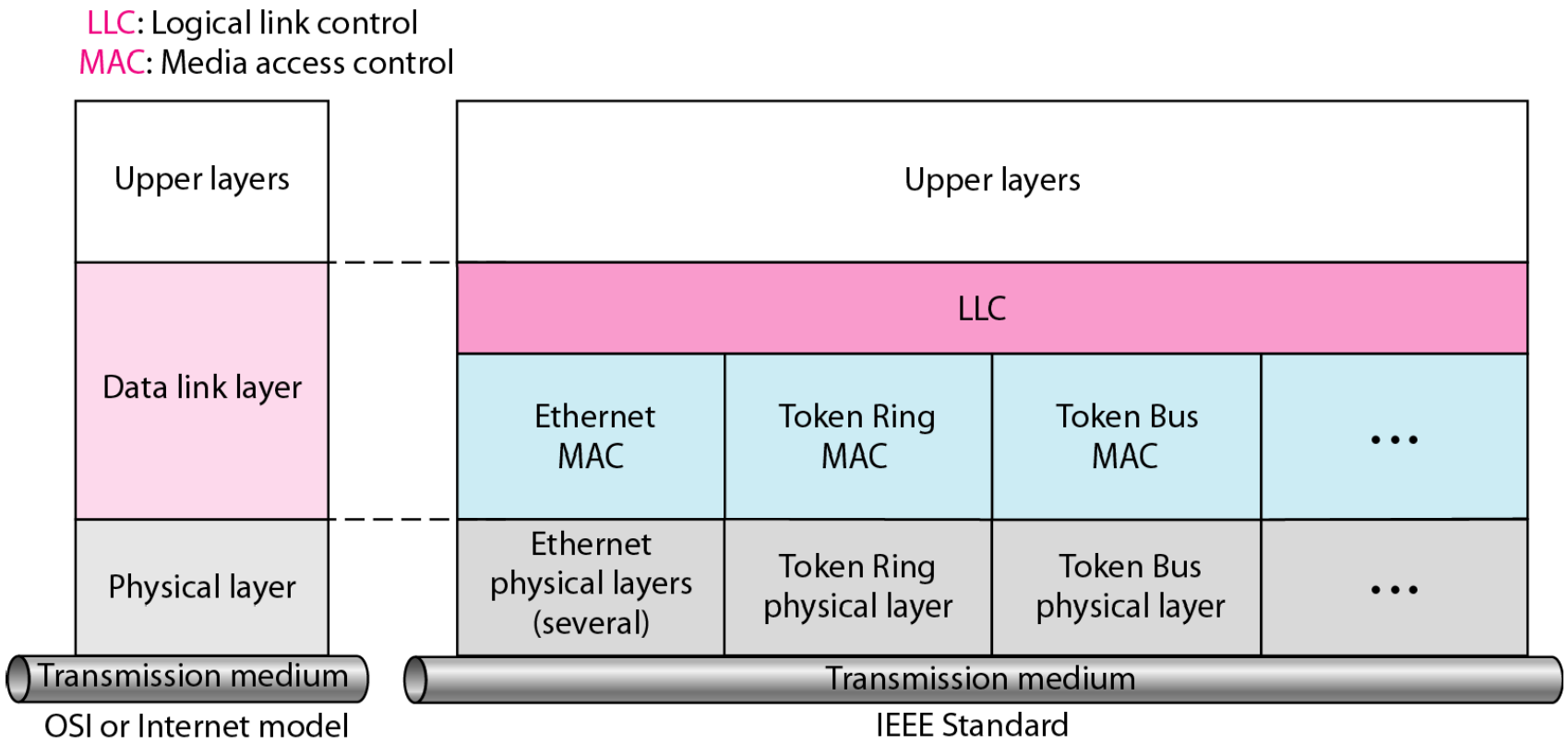
Almost every LAN except Ethernet has disappeared from the marketplace because Ethernet was able to update itself to meet the needs of the time. Organization with Ethernet upgrade to higher versions in place of change of technology.

Project 802

to set standards to enable intercommunication among equipment from a variety of vendors

The IEEE has subdivided the data-link layer into two sublayers: logical link control (LLC) and media access control (MAC).

Figure 13.1 *IEEE standard for LANs*



Sublayers

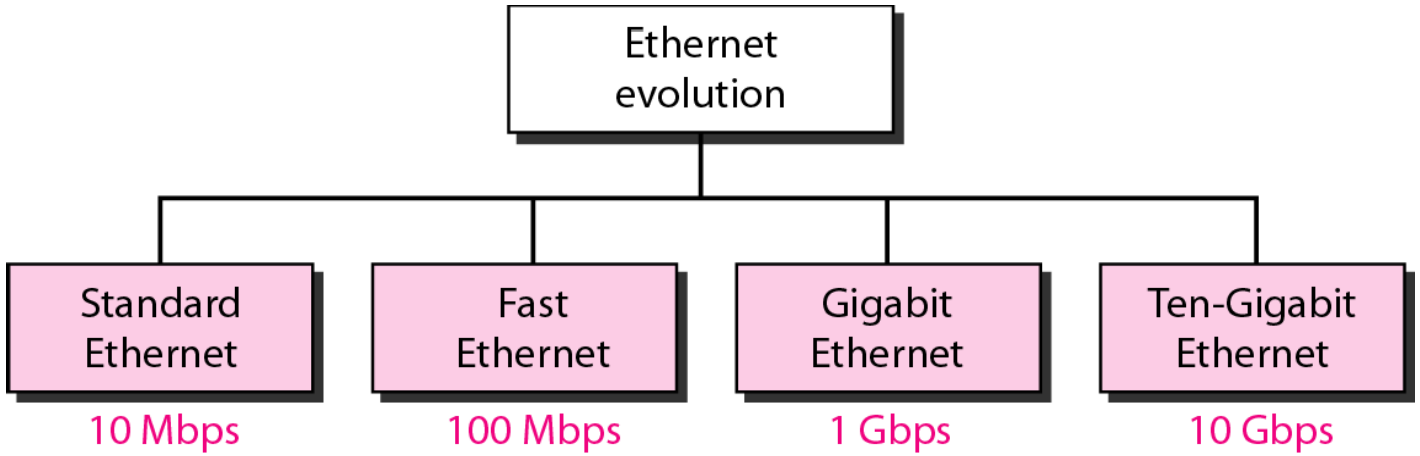
Logical Link Control (LLC)

In IEEE Project 802, flow control, error control, and part of the framing duties are collected into one sublayer called the logical link control (LLC). Framing is handled in both the LLC sublayer and the MAC sublayer.
The LLC provides a single link-layer control protocol for all IEEE LANs. This means LLC protocol can provide interconnectivity between different LANs because it makes the MAC sublayer transparent.

Media Access Control (MAC)

General - multiple access methods including random access, controlled access, and channelization.
IEEE Project 802 - defines the specific access method for each LAN. For example, it defines CSMA/CD as the media access method for Ethernet LANs and defines the token-passing method for Token Ring and Token Bus LANs. Part of the framing function is also handled by the MAC layer.

Figure 13.3 *Ethernet evolution through four generations*



13–2 STANDARD ETHERNET

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations. We briefly discuss the Standard (or traditional) Ethernet in this section.

Topics discussed in this section:

MAC Sublayer
Physical Layer

Characteristics of Ethernet

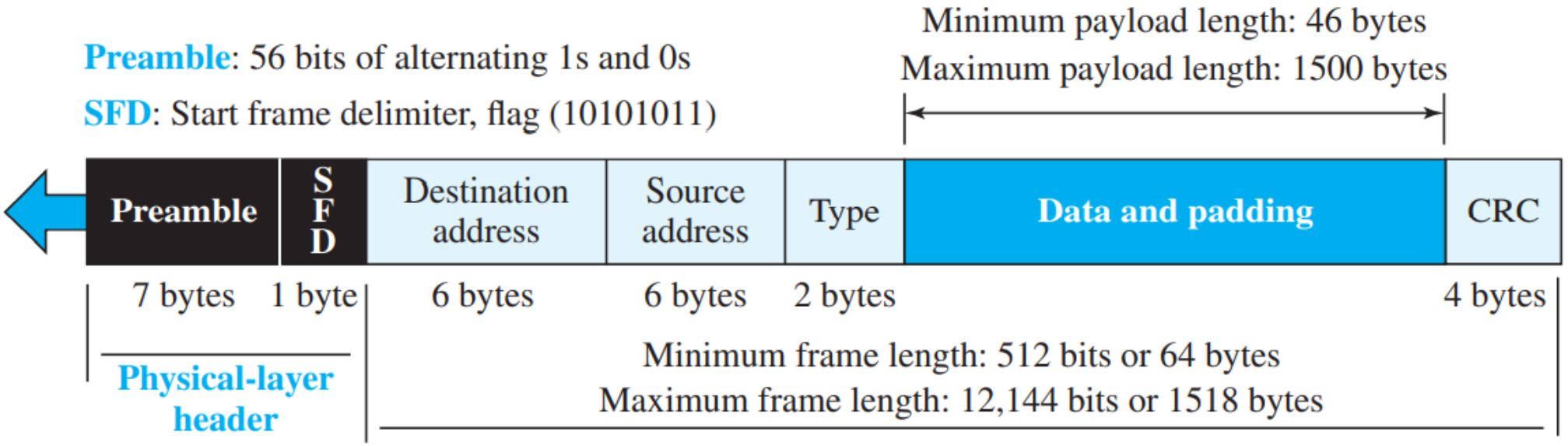
Connectionless and Unreliable Service

- each frame sent is independent of the previous or next frame.
- Ethernet has no connection establishment or connection termination phases.
- The sender sends a frame whenever it has it; the receiver may or may not be ready for it.
- Since IP, which is using the service of Ethernet, is also connectionless, it will not know about it either.
- If the transport layer is also a connectionless protocol, such as UDP, the frame is lost and salvation may only come from the application layer.
- Ethernet is also unreliable like IP and UDP.
- If a frame is corrupted during transmission and the receiver finds out about the corruption, which has a high level of probability of happening because of the **CRC-32**, the receiver drops the frame silently.

Frame Format

The Ethernet frame contains seven fields, as shown in Figure 13.3.

Figure 13.3 *Ethernet frame*



Frame structure

Preamble.

contains 7 bytes (56 bits) of alternating 0s and 1s that alert the receiving system to the coming frame and enable it to synchronize its clock if it's out of synchronization.

added at the physical layer and is not (formally) part of the frame.

Start frame delimiter (SFD). This field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization.

Ethernet frame is a variable-length frame. It needs a flag to define the beginning of the frame. The SFD field is also added at the physical layer

Destination address (DA), Source address (SA) – MAC address of Destination and source

Type:

defines the upper-layer protocol

Data:

it is a minimum of 46 and a maximum of 1500 bytes.

more than 1500 bytes, it should be fragmented and encapsulated in more than one frame.

If it is less than 46 bytes, it needs to be padded with extra 0s.

Removing/adding padding done by upper layer

Frame structure

CRC (Cyclic Redundancy Check)

CRC-32 used.

calculated over the addresses, types, and data field

if found corrupted, frame dropped

Frame Length

maximum length of a frame (without preamble and SFD field) as 1518 bytes.

If we subtract the 18 bytes of header and trailer, the maximum length of the payload is 1500 bytes.

Reasons for the maximum length restriction:

First, memory was very expensive when Ethernet was designed; a maximum length restriction helped to reduce the size of the buffer.

Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Minimum frame length: 64 bytes	Maximum frame length: 1518 bytes
Minimum data length: 46 bytes	Maximum data length: 1500 bytes

Note

Frame length:

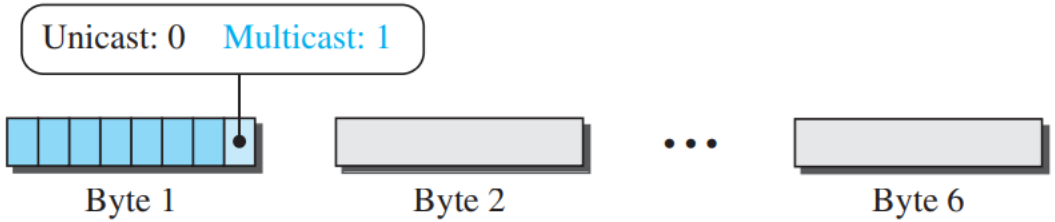
Minimum: 64 bytes (512 bits)

Maximum: 1518 bytes (12,144 bits)

Addressing

Each station has its own network interface card (NIC).
The NIC fits inside the station and provides the station with a link-layer address.
The Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between the bytes.
For example, **4A:30:10:21:10:1A**

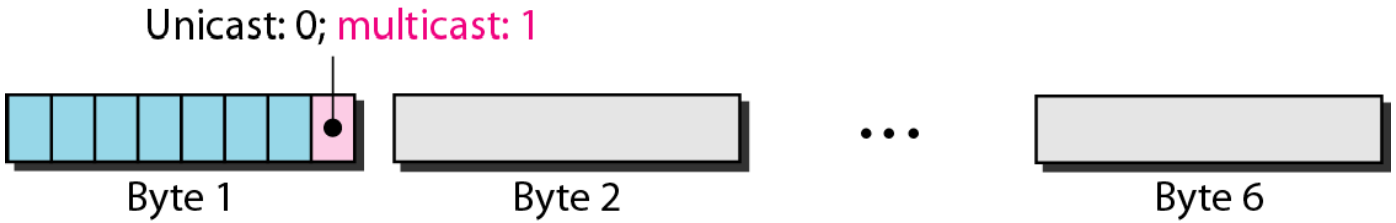
Figure 13.4 *Unicast and multicast addresses*



multicast address: the recipients are all the stations on the LAN. A broadcast destination address is forty-eight 1s.

If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.
the way the bits are transmitted, the unicast/multicast bit is the first bit which is transmitted/received.

Figure 13.7 *Unicast and multicast addresses*



Transmission of Address Bits

Example 13.1

Show how the address 47:20:1B:2E:08:EE is sent out online.

Solution

The address is sent left to right, byte by byte; for each byte, it is sent right to left, bit by bit, as shown below:

Hexadecimal	47	20	1B	2E	08	EE
Binary	01000111	00100000	00011011	00101110	00001000	11101110
Transmitted ←	11100010	00000100	11011000	01110100	00010000	01110111

This means that the bit that defines an address as unicast or multicast arrives first at the receiver. This helps the receiver to immediately know if the packet is unicast or multicast.

Note

The least significant bit of the first byte defines the type of address.
If the bit is 0, the address is unicast; otherwise, it is multicast.

Note

The broadcast destination address is a special case of the multicast address in which all bits are 1s.

Example 13.1

Define the type of the following destination addresses:

a ***4A:30:10:21:10:1*** ***b*** ***47:20:1B:2E:08:EE***

c. ***FF:FF:FF:FF:FF:FF***

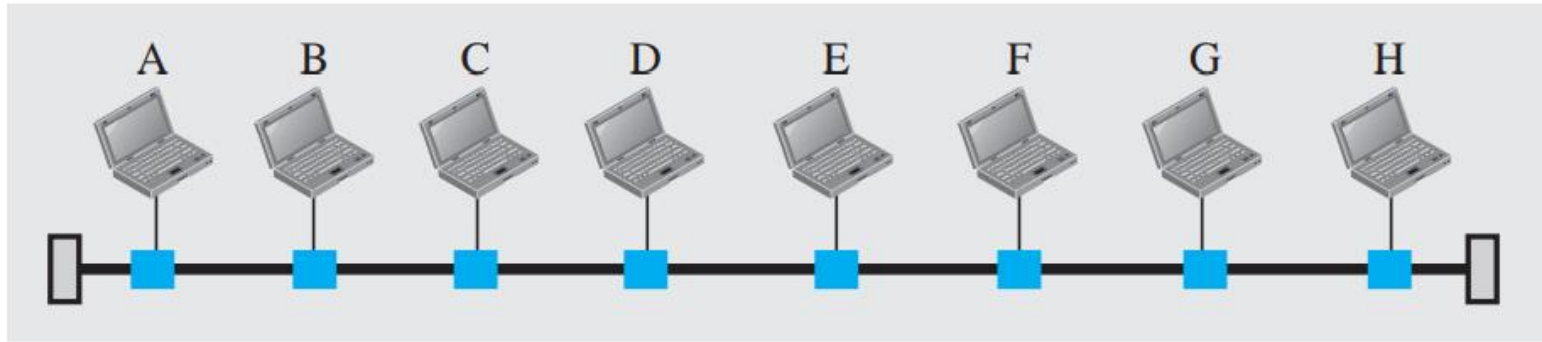
Solution

- *To find the type of the address, we need to look at the* second hexadecimal digit from the left. If it is even, the address is unicast. If it is odd, the address is multicast. If all digits are F's, the address is broadcast. Therefore, we have the following:

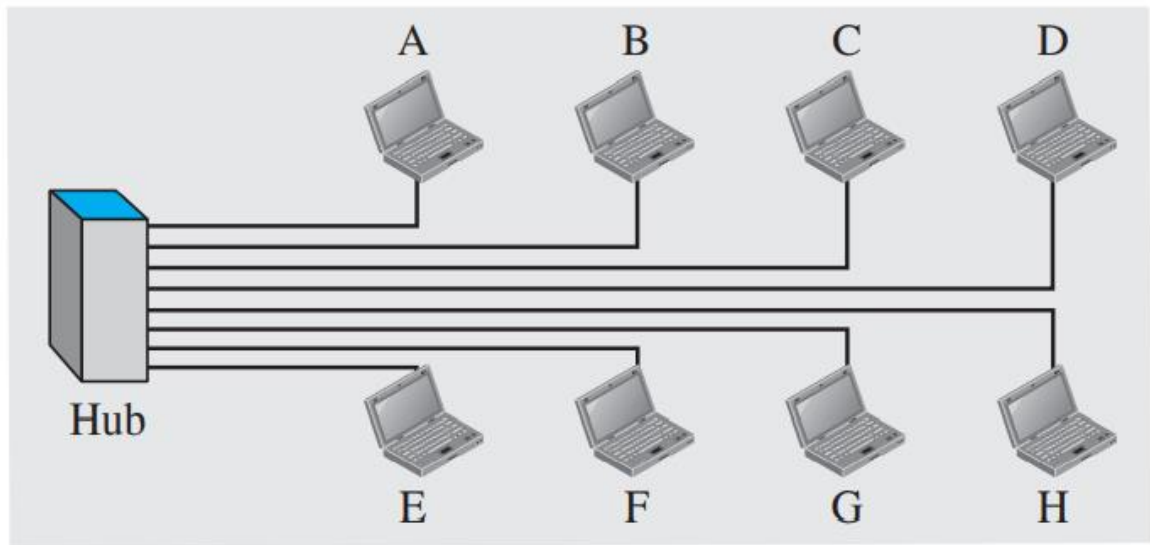
a. This is a unicast address because A in binary is 1010.

b. This is a multicast address because 7 in binary is 0111.

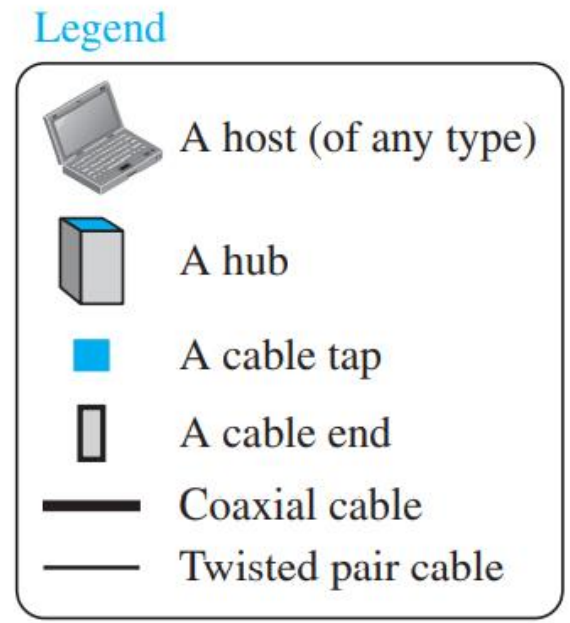
c. This is a broadcast address because all digits are F's.



a. A LAN with a bus topology using a coaxial cable



b. A LAN with a star topology using a hub



Transmission

Transmission in the standard Ethernet is always broadcast, no matter if the intention is unicast, multicast, or broadcast.
how the actual unicast, multicast, and broadcast transmissions are distinguished from each other.

In a unicast transmission, all stations will receive the frame, the intended recipient keeps and handles the frame; the rest discard it.

In a multicast transmission, all stations will receive the frame, the stations that are members of the group keep and handle it; the rest discard it

In a broadcast transmission, all stations (except the sender) will receive the frame and all stations (except the sender) keep and handle it.