

Chapter 2

Data Link Layer

Contents

- **Framing**
- **Error Detection and Correction:** Introduction Block Coding, Cyclic Codes.
- **Data link control:** DLC Services: Framing, Flow Control, Error Control, Connectionless and Connection Oriented, Data link layer protocols, High Level Data Link Control.
- **Media Access Control:** Random Access, Controlled Access. Check Sum and Point to Point Protocol

Introduction

- The primary service provided by the Data Link Layer is **framing**.
- At each node, the **Data Link Layer takes the packet** (datagram) **received from the Network Layer** and **encapsulates** it into a **frame** before transmitting it to the next node.
- Upon **receiving a frame from a logical channel**, it **decapsulates the datagram** for further processing.
- While a **frame always contains a header**, some formats also include a **trailer**.
- The **structure and format of frames vary depending on the specific Data Link Layer protocol** being used.

Introduction

- In any data communication system, transmitting **data across a channel can often result in errors due to various reasons.**
- These errors lead to the **alteration of the original data, which affects the integrity and reliability of the transmitted message.**
- Therefore, **error detection and correction mechanisms** are essential in ensuring that any such issues are identified and addressed.

Introduction

Why Do Errors Occur?

- **Noise:** Unwanted electrical signals that can distort data transmission.
- **Attenuation:** The weakening of signal strength as it travels over distance.
- **Interference:** Disruptions from external signals or other communication systems.

Introduction

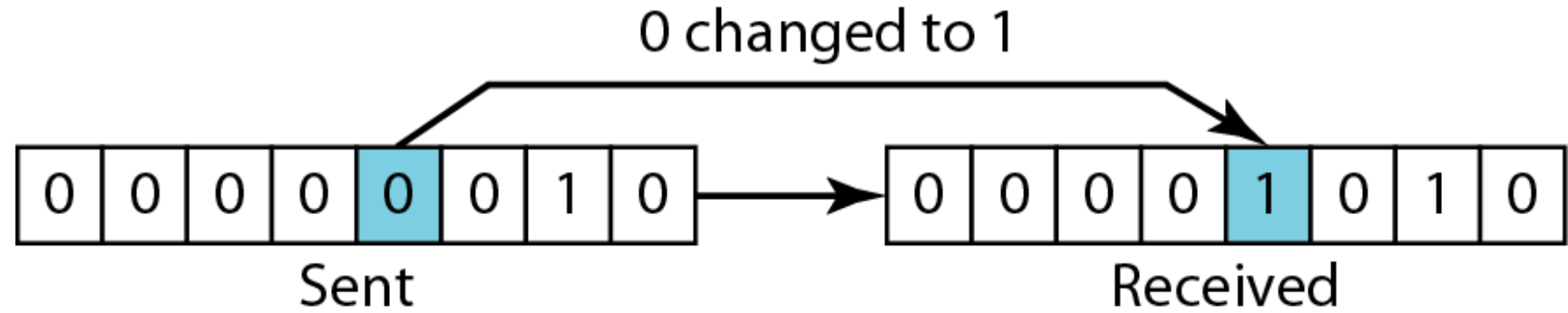
Types of Errors

- **Single-bit Error:** This occurs when **only one bit** within a data unit changes during transmission (e.g., a 1 turns into a 0 or vice versa).
- **Burst Error:** This happens when **multiple bits** within the data unit are altered, often caused by noise or interference in the communication channel.

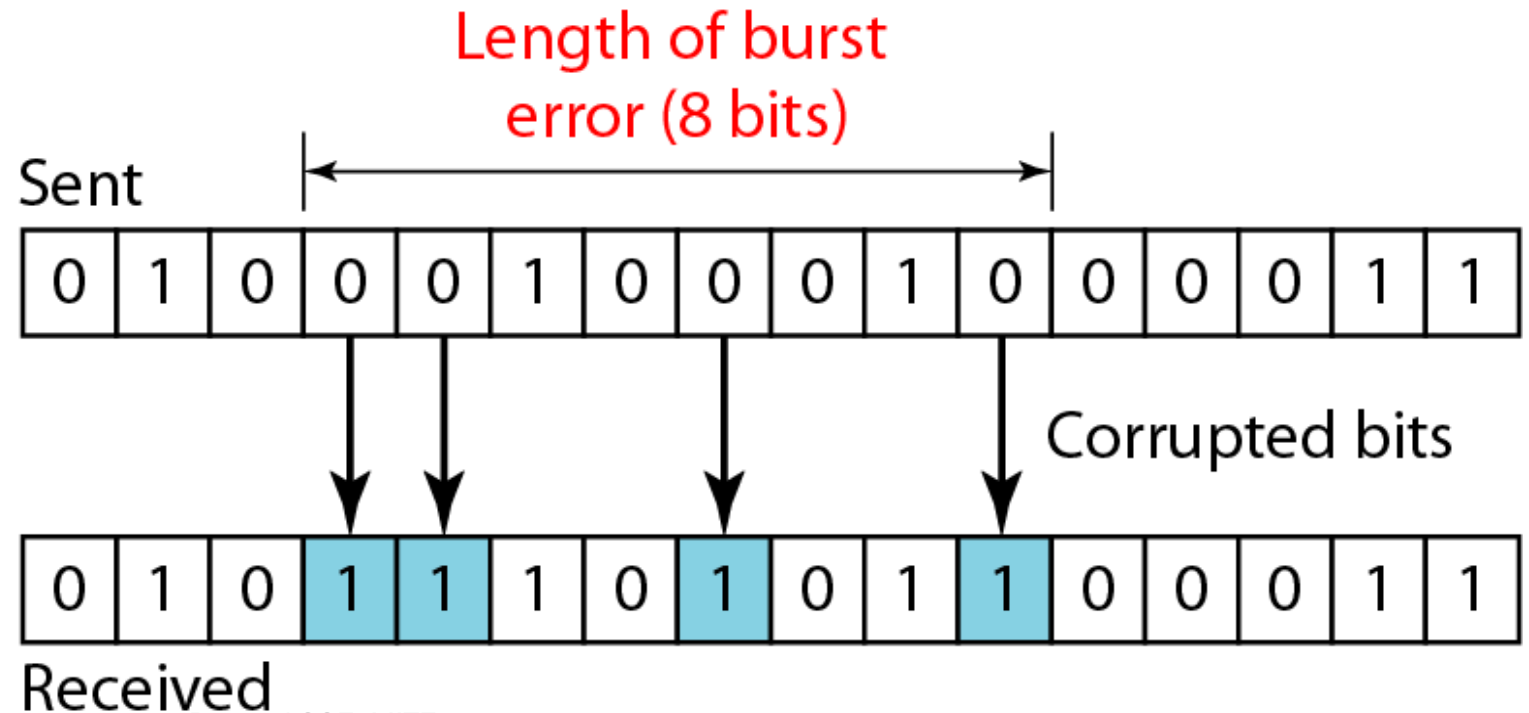
Introduction

Types of Errors

• **Single-bit Error:**



• **Burst Error:**



Introduction

Redundancy

Redundancy refers to the **addition of extra bits** to the **original message to help detect or correct errors**.

These redundant bits **provide additional information about the message**, enabling the receiver to detect and sometimes correct any errors in transmission.

Example: In a 4-bit message, adding 3 redundant bits makes it a 7-bit message, where the extra bits help in detecting or correcting errors.

Introduction

Detection Versus Correction

Error Detection is simpler as it only involves **determining whether an error occurred.**

- Checks **“error occurs” or “Not”**.
- It is **not concerned with how many bits are corrupted or their exact locations.**
- A **single-bit error** is treated same as a **burst error** during **error detection.**

Error Correction is more complex because:

- We need to know the **exact number of bits** that are corrupted.
- More importantly, we must identify the **location** of the **corrupted bits** in the message.
- The **number of errors and the size of the data unit** play a crucial role in correction.

Introduction

Detection Versus Correction

Complexity of Error Correction

- To correct a single-bit error in an **8-bit** data unit, we must consider **8 possible error locations**.

- To correct **two errors** in an **8-bit data unit**, we must consider **28 combinations** (permutations of 8 taken 2 at a time). $8C_2 =$

$$\binom{8}{2} = \frac{8!}{2!(8-2)!} = \frac{8 \times 7}{2 \times 1} = 28$$

- The **difficulty increases significantly** with **larger data units and more errors**.

Introduction

Coding

In communication systems, **coding** is the **process of transforming the original message into a different form that allows error detection and correction.**

- **Block Coding:** Divides data into blocks and adds redundant bits to each block.
- **Convolutional Coding:** Instead of dividing data into blocks, it considers the input data as a continuous stream and adds redundancy by encoding the data with a sliding window of bits.

Introduction

Block Coding

Dividing the Message:

- The message is **split into fixed-size blocks of k bits**. Each of these blocks is referred to as a **dataword**.

- For example, if you have a **message of 16 bits**, and you **divide it into 4-bit blocks**, then you have **4 datawords** (each 4 bits long).

Introduction

Block Coding

Redundant Bits:

- To help with **error detection** (and potentially error correction), **redundant bits** (denoted as **r**) are **added to each dataword**.
- The **total length of each block**, called a **codeword**, is **$n = k + r$** , where **k** is the **number of data bits**, and **r** is the **number of redundant bits**.
- The purpose of these **redundant bits** is to **add information** that can be **used to check** if the **dataword has been corrupted during transmission**.

Introduction

Block Coding

Number of Datawords and Codewords:

- With **k bits** in each dataword, there are **2^k possible combinations** of datawords (since each bit can be either 0 or 1).
- With **n bits** in each codeword, there are **2^n possible combinations** of codewords.
- Since **$n > k$** , the number of possible codewords (2^n) is **larger** than the number of possible datawords (2^k).

Introduction

Block Coding

Number of Datawords and Codewords:

Suppose we have:

- $k = 3$ bits per dataword.
- $n = 5$ bits per codeword.

Step 1: Possible datawords

With $k = 3$, the number of possible datawords is:

$$2^k = 2^3 = 8 \text{ possible datawords.}$$

These datawords can be:

000, 001, 010, 011, 100, 101, 110, 111

Introduction

Block Coding

Number of Datawords and Codewords:

Step 2: Possible codewords

With $n = 5$, the number of possible codewords is:

$$2^n = 2^5 = 32 \text{ possible codewords.}$$

These codewords can be combinations of 5 bits, like:

00000, 00001, 00010, 00011, ..., 11111

Thus, there are 32 distinct codewords.

Since $n > k$, the number of possible codewords (32) is larger than the number of possible datawords (8).

Introduction

Block Coding

- **Invalid or Illegal Codewords:**

There are **more codewords than datawords** which are not assigned to any data word, are called **invalid** or **illegal codewords**.

- **Why is this important?**

- These invalid codewords act as a **detection mechanism** for errors. If a corrupted codeword matches one of the invalid codewords, the receiver can immediately tell that an error has occurred during transmission.

Introduction

Block Coding

Error Detection

If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid codewords.
2. The original codeword has changed to an invalid one.

Introduction

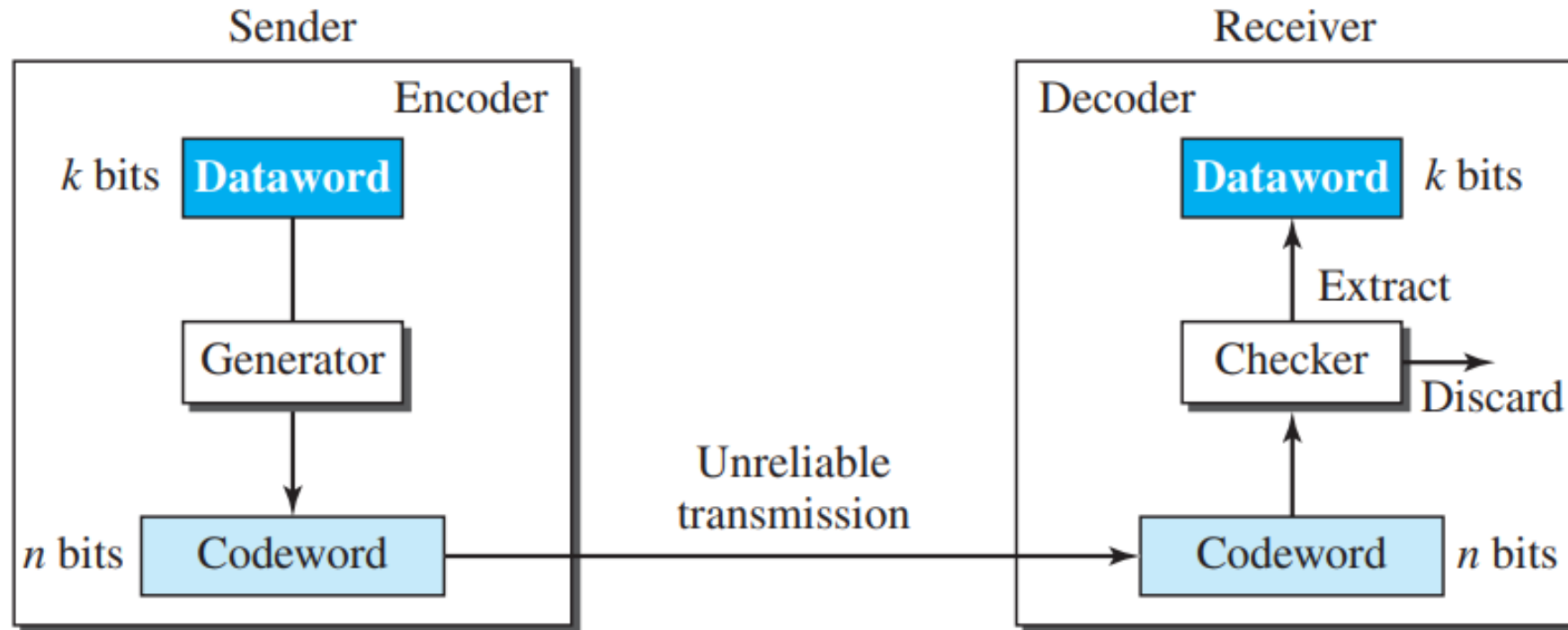
Block Coding

Error Detection-: One-to-One Mapping:

- In block coding, the **mapping is one-to-one**, meaning that a specific dataword is always encoded as the **same codeword**. There's a direct relationship between datawords and their corresponding codewords.
- the dataword 001 is always mapped to the codeword 0010
- This consistency is crucial for error detection, as any deviation from the expected codeword (due to corruption) indicates an error.

Introduction

Block Coding



Introduction

Block Coding

Let us assume that $k = 2$ and $n = 3$. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

<i>Dataword</i>	<i>Codeword</i>	<i>Dataword</i>	<i>Codeword</i>
00	000	10	101
01	011	11	110

Introduction

Block Coding

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

1. The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.
2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.
3. The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

Hamming Distance

- **What is Hamming Distance?**

Hamming distance is a measure of the difference between two binary numbers of the same length. It counts how many bits are different between them.

- **Why is it Important?**

In data transmission, errors can occur. The Hamming distance tells us how many bits have changed, helping detect and sometimes correct these errors.

Hamming Distance

How does Hamming Distance Help in Error Detection?

In a communication system, data is transmitted in the form of binary numbers (like 1010, 1101). Sometimes, due to noise or interference, some bits get corrupted (flipped from 0 to 1 or 1 to 0). The **Hamming Distance** between the **sent codeword** and the **received codeword** tells us how many bits got flipped (errors).

In many **simpler error detection schemes**, where the relationship $n = k + r$ (total bits = data bits + redundancy bits) is used to describe the overall structure of a codeword. In this case, the **r redundancy bits** are added to the **k data bits** to form an **n-bit codeword**.

Hamming Distance

- **Hamming distance = number of bit positions in which two codewords differ.**

To detect up to s errors, the minimum Hamming distance (**dmin**) must be:

$$\mathbf{dmin} \geq s + 1$$

Example: Detecting 1-bit errors

To detect **1-bit error**, set $s = 1$. So, **dmin ≥ 2**

- Any **two valid codewords must differ** by at least **2 bits**.
- If only **1 bit flips**, the new word will **not become another valid codeword**, so the receiver will detect an error.

If dmin = 1

- Only one bit is different between codewords.
- A **1-bit error can change one valid codeword into another**, so the error may go undetected.

Hamming Distance

How to Calculate Hamming Distance?

- Use the **XOR operation** on two binary numbers. The result shows which bits are different. Count the number of 1s in the result to get the Hamming distance.
- Example: For binary numbers 10101 and 11110, the XOR result is 01011, which has 3 ones. So, the Hamming distance is 3.

$$\begin{array}{r} 10101 \\ 11110 \\ \hline \text{XOR } 01011 \\ \hline \end{array}$$

• **Redundancy bits** are extra bits added to detect/correct errors.

• **Hamming code** is a method using redundancy bits to correct single-bit errors and detect two-bit errors.

Hamming Distance

Hamming Code: Redundancy Bits Not at the End

Redundancy bits are **inserted at specific positions within the codeword** that are **powers of 2** (1, 2, 4, 8, etc.). These positions are strategically chosen so that each redundancy bit checks different combinations of the data bits.

Example 1: For 4 Data Bits ($k = 4$)

Let's calculate how many redundancy bits are needed: $2^r \geq r + k + 1$

$$2^r \geq r + 4 + 1$$

$$2^r \geq r + 5$$

Now, try different values of r :

- For $r = 3$:

$$2^3 = 8 \geq 3 + 5 = 8$$

Dept of CSE, MITE

So, **3 redundancy bits** are required for 4 data bits.

Hamming Distance

Position: 1 2 3 4 5 6 7

Data: r1 r2 1 r4 0 1 1

Data word is 1011

r1 checks all positions where the **1st bit** in the binary representation of the position is **1**.

r2 checks all positions where the **2nd bit** in the binary representation of the position is **1**.

r3 checks all positions where the **3rd bit** in the binary representation of the position is **1**.

Position Binary

1 001

2 010

3 011

4 100

5 101

6 110

7 111

Position: 1 2 3 4 5 6 7

Content: 0 1 1 0 0 1 1

r1 r2 d1 r4 d2 d3 d4 = 7 bits

Hamming Distance

Steps to Calculate Redundancy Bits (r1, r2, r4) using even parity:

1. r1:

- Covers positions 1, 3, 5, 7.
- Checks D1 (position 3), D2 (position 5), D4 (position 7)
- Data bits: 1, 0, 1 → Sum = 2 (even).
- r1 = 0 (since the sum is even).

2. r2:

- Covers positions 2, 3, 6, 7.
- Checks D1 (position 3), D3 (position 6), D4 (position 7).
- Data bits: 1, 1, 1 → Sum = 3 (odd).
- r2 = 1 (since the sum is odd).

3. r4:

- Covers positions 4, 5, 6, 7.
- Checks D2 (position 5), D3 (position 6), D4 (position 7)
- Data bits: 0, 1, 1 → Sum = 2 (even).
- r4 = 0 (since the sum is even).

Hamming Distance

Compare Recalculated Redundancy Bits with Received Redundancy Bits:

- Now, compare the recalculated redundancy bits with the ones received as part of the codeword.

Two cases:

- If the recalculated redundancy bits **match** the received redundancy bits, then **no error** has occurred.
- If the recalculated redundancy bits **do not match** the received redundancy bits, an **error** has occurred.

Hamming Distance

Add Redundancy Bits:

- The sender adds redundancy bits to the dataword to form the **codeword**.

Transmit Codeword:

- The codeword is sent to the receiver.

Receive and Recalculate Redundancy:

- The receiver recalculates the redundancy bits based on the received codeword.

Calculate Hamming Distance:

- The receiver calculates the **Hamming distance** between the sent and received codeword to determine how many errors occurred.

Error Correction:

- If the error is within the correction limit, the system corrects the error using the redundancy bits.

Parity-Check Code

Parity-Check Code is a basic **error-detecting code** where a **k-bit dataword** is converted to an **n-bit codeword** by adding one extra bit called the **parity bit**.

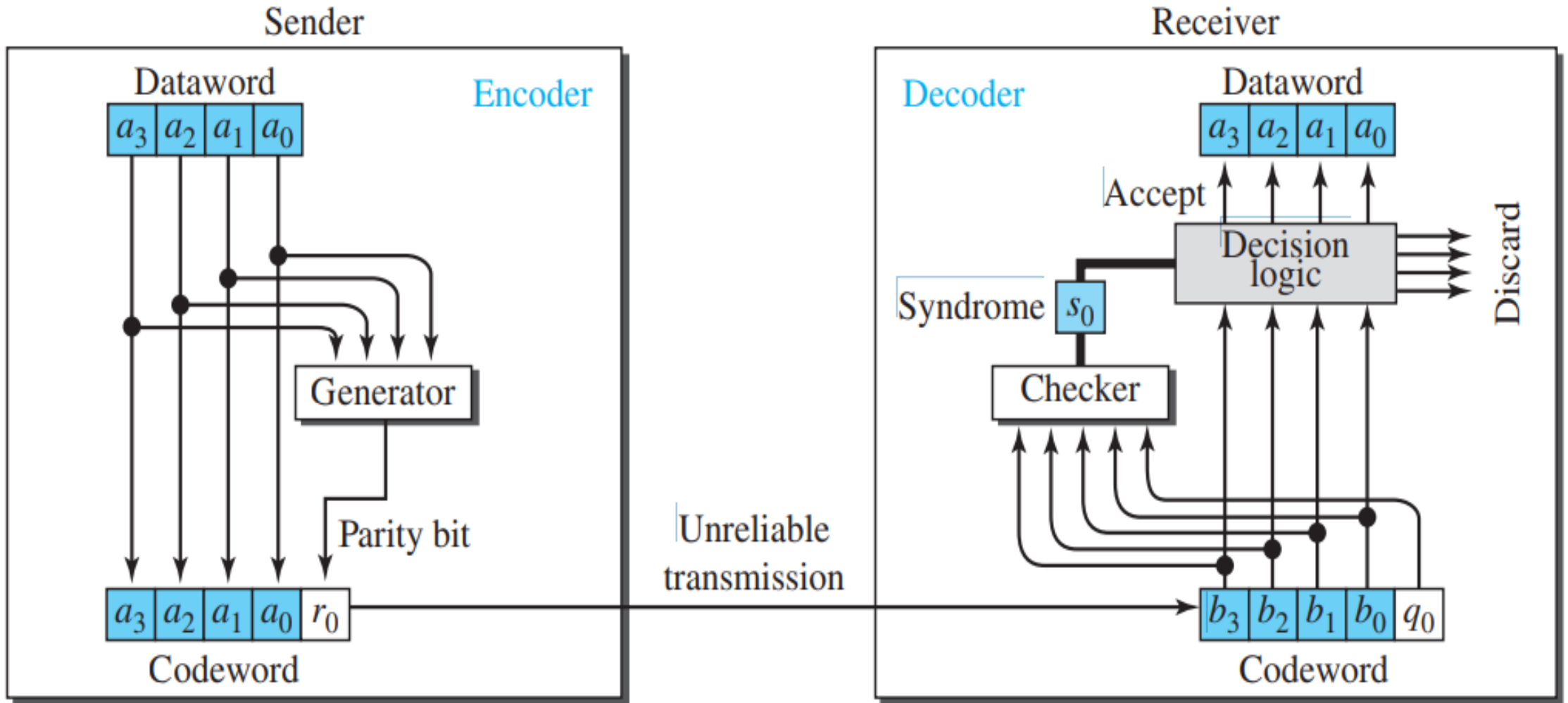
Parity bits are used for **simple error detection(single bit detection)**

This **bit ensures the total number of 1s in the codeword is even** (or odd in some cases).

The **encoder** generates the **parity bit by adding the data bits** using **modulo-2** arithmetic. At the receiver, a **syndrome** is calculated by adding all the bits in the received codeword.

If the syndrome is **0**, there is no detectable error, and the data is accepted. If the syndrome is **1**, an error is detected, and the dataword is discarded.

Parity-Check Code



Parity-Check Code

Sender (Left Side) - Encoder:

1. Dataword:

- The sender has a 4-bit **dataword** made up of bits a_3, a_2, a_1, a_0 .

2. Generator:

- The **generator** is responsible for calculating the **parity bit** r_0 .
- The parity bit is calculated by adding the 4 data bits using **modulo-2 (XOR)** operation. It ensures that the total number of 1s in the codeword is even (for **even parity**).
 - If the number of 1s in a_3, a_2, a_1, a_0 is **even**, $r_0 = 0$.
 - If the number of 1s is **odd**, $r_0 = 1$.

3. Codeword:

- The **codeword** is generated by combining the original data bits a_3, a_2, a_1, a_0 with the calculated parity bit r_0 , forming a 5-bit codeword.
- This 5-bit codeword is sent over the communication channel, which may introduce errors during **unreliable transmission**. Dept of CSE, MITE

Parity-Check Code

Receiver (Right Side) - Decoder:

1. Codeword Received:

- The receiver receives the 5-bit codeword: b_3, b_2, b_1, b_0, q_0 , where b_3, b_2, b_1, b_0 represent the data bits and q_0 is the parity bit received (which may or may not match r_0).

2. Checker:

- The **checker** recalculates the parity bit from the received data bits b_3, b_2, b_1, b_0 using the same method as the sender.
- The parity bit is compared with the received parity bit q_0 .
- The result of this comparison is called the **syndrome** s_0 , which is a single bit.
 - **If $s_0 = 0$:** No error is detected, meaning the codeword is considered valid.
 - **If $s_0 = 1$:** An error is detected, meaning the codeword is corrupted.

3. Decision Logic:

- The **syndrome** s_0 is passed to the **decision logic**, which decides whether to accept or discard the received data.
- If $s_0 = 0$, the data is accepted, and the data bits b_3, b_2, b_1, b_0 are considered correct.
- If $s_0 = 1$, an error is detected, and the codeword is discarded (no dataword is produced).

Parity-Check Code

<i>Dataword</i>	<i>Codeword</i>	<i>Dataword</i>	<i>Codeword</i>
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 10.4 shows a possible structure of an encoder (at the sender) and a decoder (at the receiver).

Cyclic Code

Cyclic codes are a class of **error-detecting and error-correcting codes** with a unique property that allows them to detect and correct errors in data transmission and storage. **multiple-bit errors detection**

- 1. Cyclic Redundancy Check (CRC):** CRC is an **error-detection method** used in data transmission to detect errors in messages or data blocks.
- 2. Polynomials (In Context of CRC):** Polynomials are a **mathematical tool** used within CRC.

Cyclic Redundancy Check (CRC)

Cyclic Redundancy Check (CRC) is an **error-detection technique** used in digital networks and storage devices to **detect accidental changes to raw data**. It is widely used in systems like **LANs, WANs**, and storage devices to ensure data integrity during transmission.

How CRC Works:

1.Dataword:

1. The **dataword** is the original set of bits (data) that needs to be transmitted.

2.Divisor (Polynomial):

1. CRC works by **treating** both the **dataword** and a **predefined divisor** as binary numbers (or polynomials). The divisor is agreed upon by both the sender and receiver before communication..

Cyclic Redundancy Check (CRC)

3. Augmentation:

- Before transmission, the dataword is **augmented** by appending **$n - k$ zeros** (where n is the length of the codeword, and k is the **length of the dataword**). This ensures that there is space for the **remainder** (called **CRC bits**) generated in the next step.

4. Modulo-2 Division:

- The augmented dataword is divided by the divisor using **modulo-2 binary division** (similar to regular division, but with XOR used for subtraction).
- The result of this division is a **remainder** (CRC bits), which is appended to the original dataword to create the final **codeword**.

Cyclic Redundancy Check (CRC)

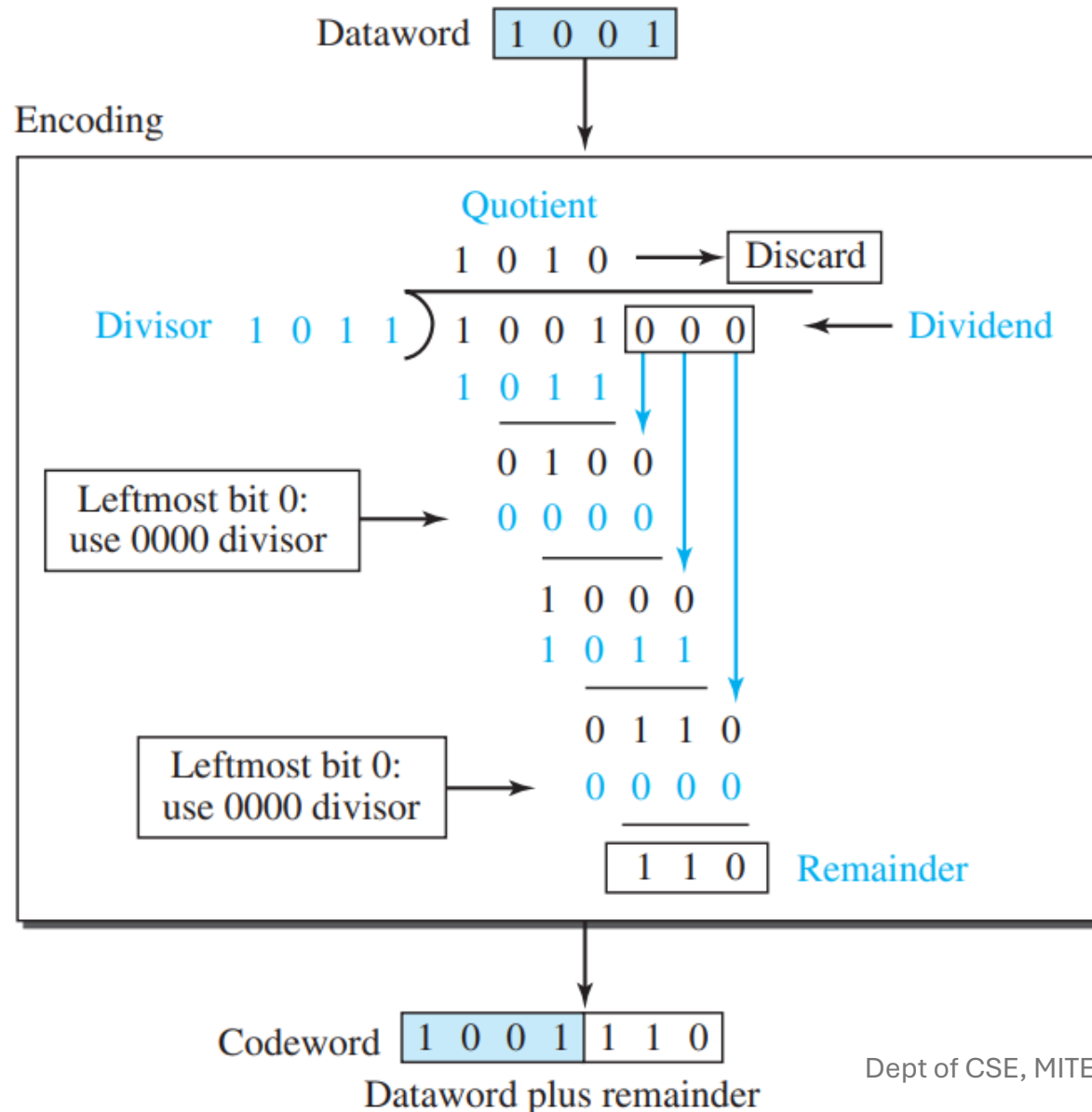
5. Transmission:

- The final **codeword** (dataword + CRC bits) is **transmitted to the receiver**. If any bit changes during transmission (due to noise or interference), the error can be detected.

6. Error Detection:

- At the receiver, **the same division process is applied**. The **received codeword is divided by the same divisor**. The remainder is called the **syndrome**.
- If the syndrome is all **zeros**, no errors are detected, and the data is accepted.
- If the syndrome is **non-zero**, an error is detected, and the codeword is discarded.

Cyclic Redundancy Check (CRC)



Note:

Multiply: AND
Subtract: XOR

The dataword is $k = 4$ bits.

The codeword is $n = 7$ bits.

The divisor length is given as $n - k + 1$.

In this case, $n - k + 1 = 7 - 4 + 1 = 4$,

Cyclic Redundancy Check (CRC)

CRC Encoding Steps:

1.Dataword: The dataword is 1001.

2.Augmentation: Add three zeros to make it 1001000 (since the divisor is 4 bits).

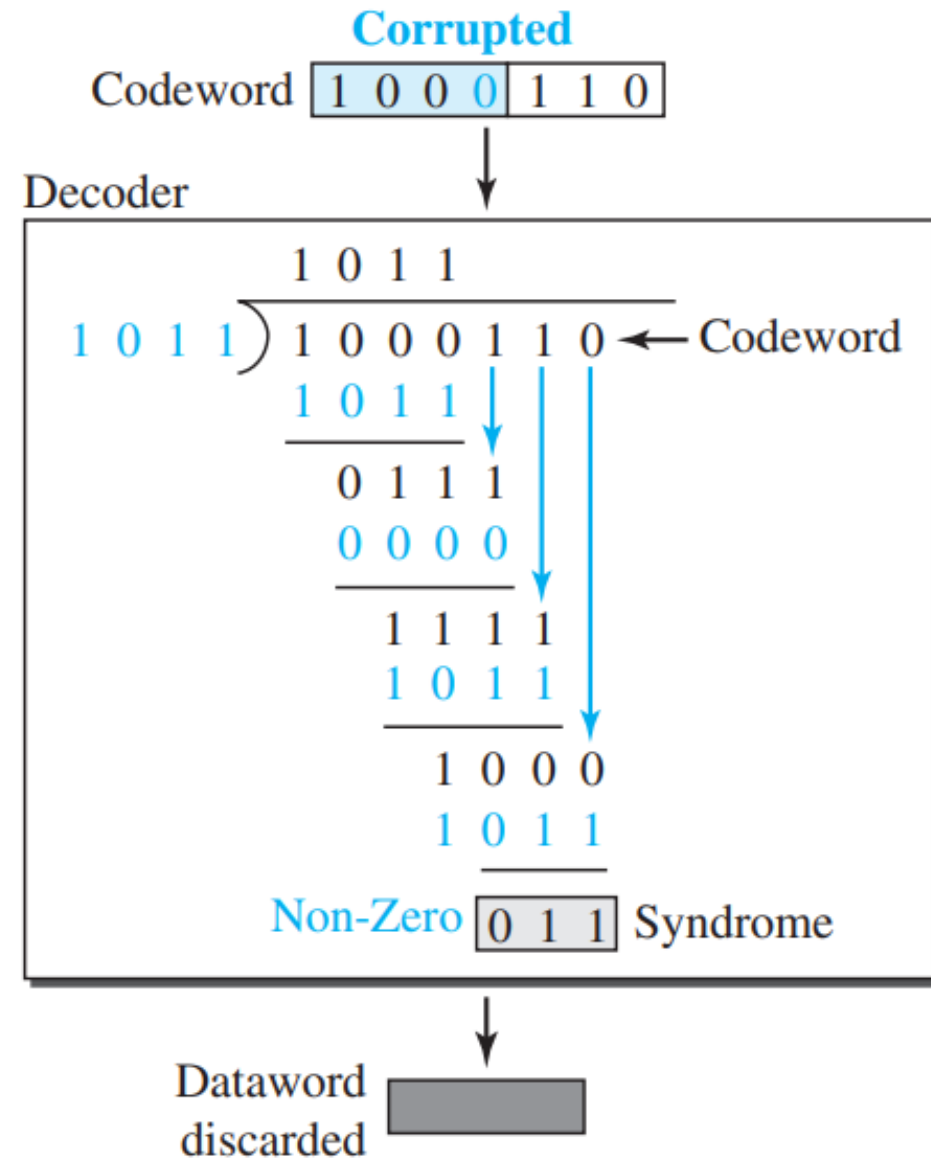
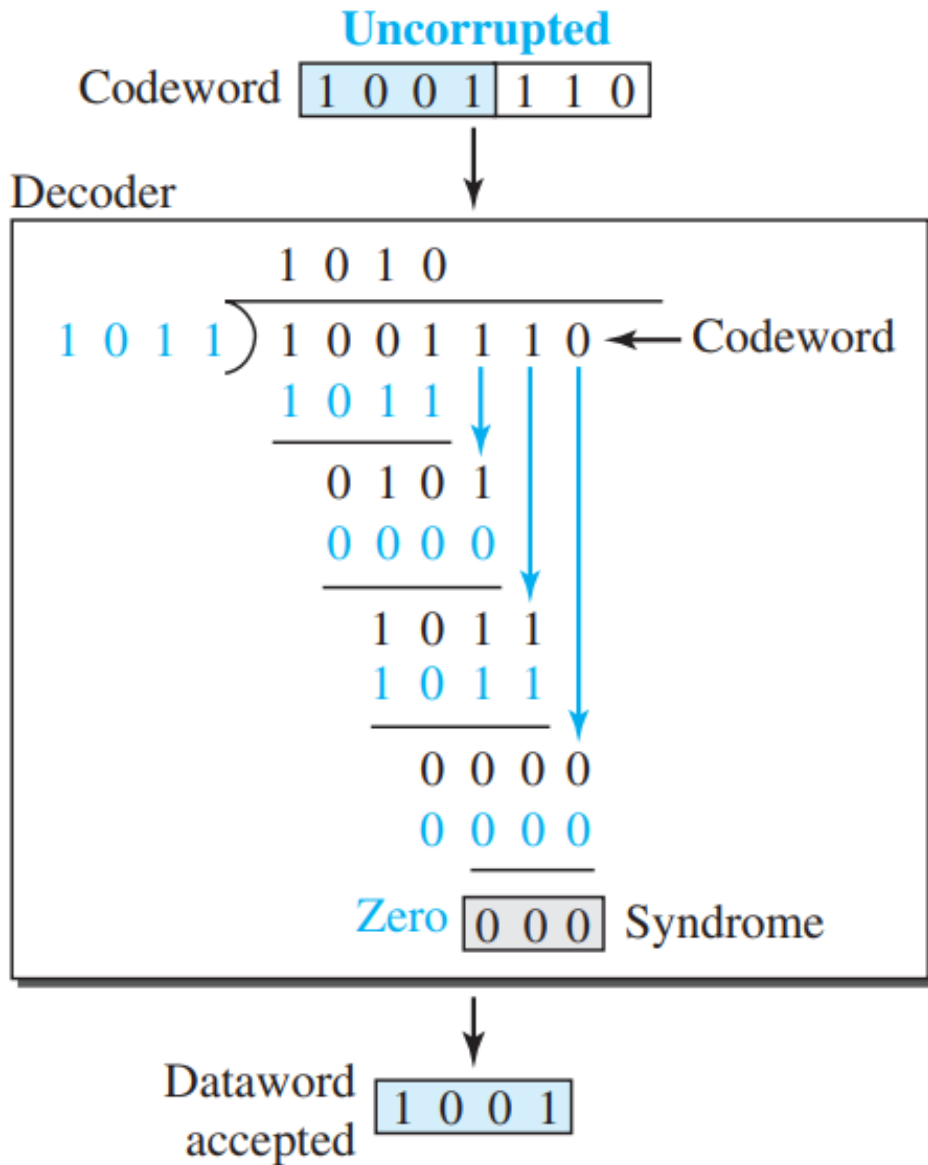
3.Divisor: Use the predefined divisor 1011.

4.Modulo-2 Division: Perform bit-by-bit XOR division between the augmented dataword and divisor.

5.Remainder: The remainder after division is 110.

6.Codeword: Append the remainder to the original dataword, forming the final codeword 1001110.

Cyclic Redundancy Check (CRC)



Cyclic Redundancy Check (CRC)

CRC Decoding Steps:

1. Uncorrupted Codeword (1001110):

- Perform the same division using the divisor.
- **Syndrome** = 000 (no error detected), so the dataword 1001 is **accepted**.

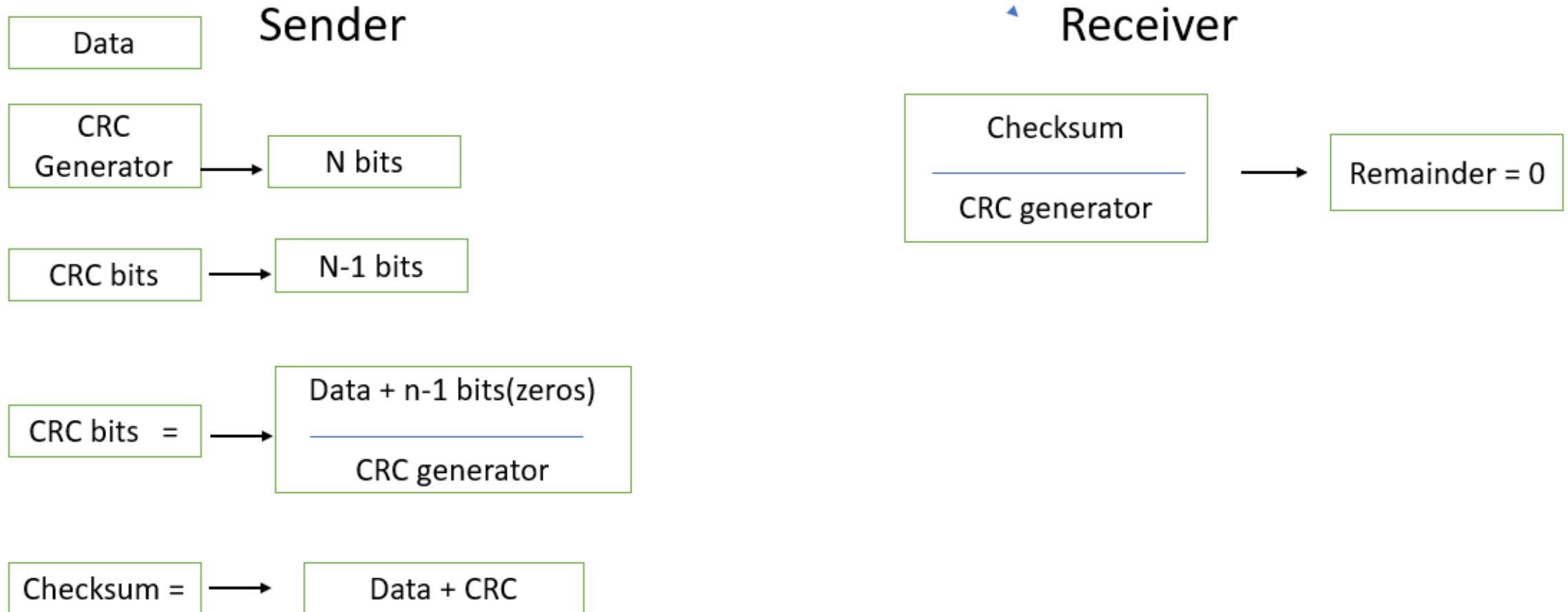
2. Corrupted Codeword (1000110):

- Perform division with the divisor.
- **Syndrome** = 011 (error detected), so the dataword is **discarded**.

• **Encoding:** Dataword + remainder = codeword.

• **Decoding:** Recalculate the syndrome. If it's all zeros, accept the data; otherwise, discard it.

Cyclic Redundancy Check (CRC)

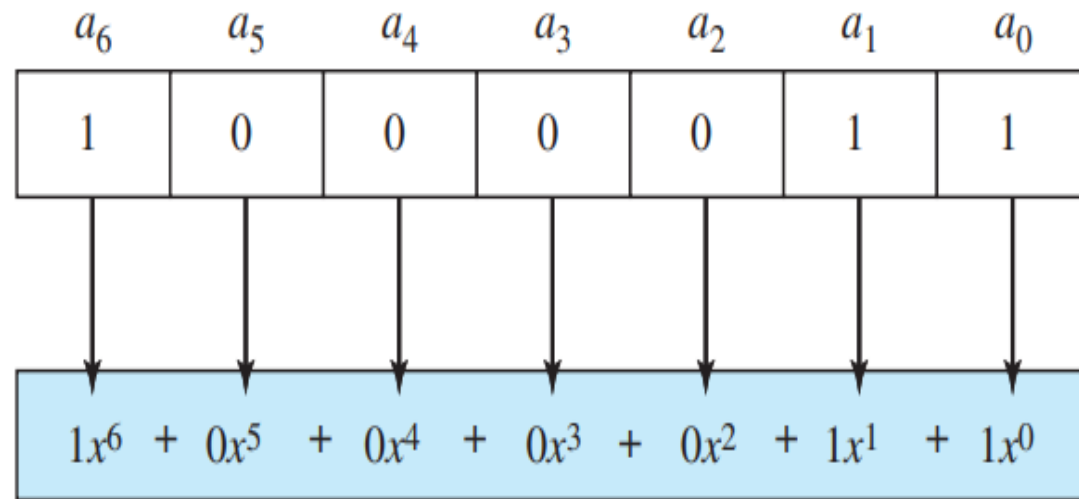


Polynomials

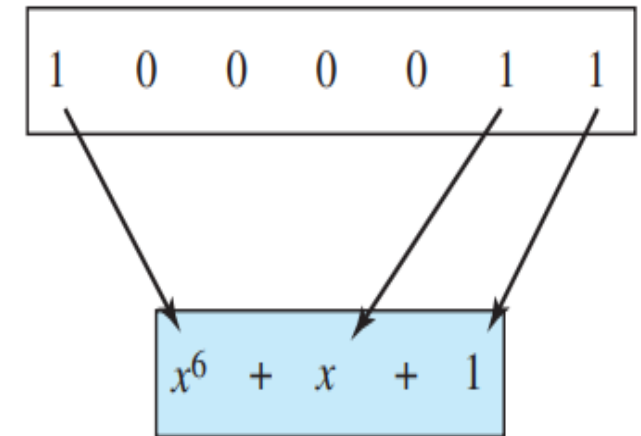
Cyclic codes is to represented as polynomials.

A binary pattern of 0s and 1s can be expressed as a polynomial, where each term's power indicates the bit's position, and the coefficient shows its value.

For example, we see how to convert a binary pattern to a polynomial and simplify it by removing terms with zero coefficients.



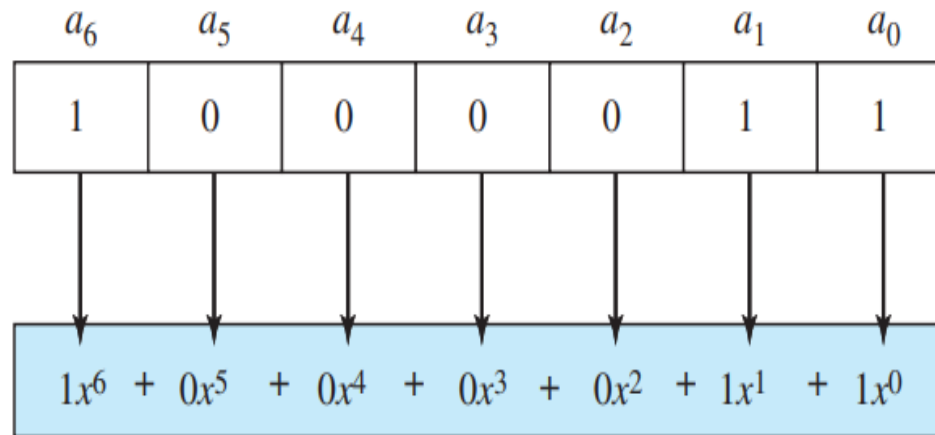
a. Binary pattern and polynomial



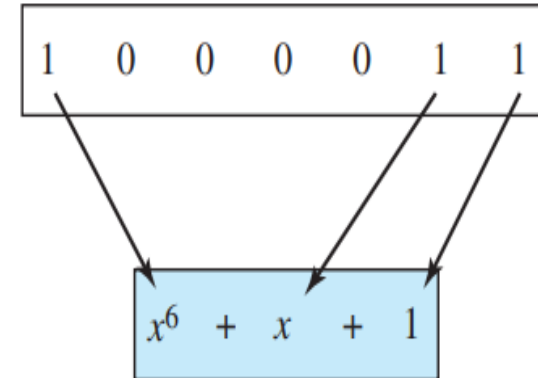
b. Short form

Polynomials

This method can significantly reduce the representation size: **a 7-bit pattern can become just three terms in a polynomial form, making analysis simpler.**



a. Binary pattern and polynomial



b. Short form

Degree of a Polynomial

The degree of a polynomial is the **highest power in the polynomial.**

For example, the degree of the polynomial $x^6 + x + 1$ is **6**. Note that the degree of a polynomial is 1 less than the number of bits in the pattern. The bit pattern in this case has 7 bits.

Polynomials

Multiplying or Dividing Terms

In this arithmetic, **multiplying a term** by another term is very simple; we just **add the powers**. For example, $x^3 \times x^4$ is x^7 . For dividing, we just subtract the power of the second term from the power of the first. For example, x^5/x^2 is x^3 .

Multiplying Two Polynomials: Multiplying a polynomial by another is **done term by term**. Each term of the first polynomial must be multiplied by all terms of the second.

$$\begin{aligned}(x^5 + x^3 + x^2 + x)(x^2 + x + 1) &= x^7 + x^6 + x^5 + x^5 + x^4 + x^3 + x^4 + x^3 + x^2 + x^3 + x^2 + x \\ &= x^7 + x^6 + x^3 + x\end{aligned}$$

Polynomials

Dividing One Polynomial by Another.

Polynomial division is similar to **binary division used in encoding**. We divide the first term of the dividend by the first term of the divisor to get the first term of the quotient, then multiply and subtract from the dividend.

$$\begin{array}{r} \text{Divisor} \\ x^3 + x + 1 \end{array} \overline{) \begin{array}{r} x^3 + x \\ x^6 + + x^3 \\ \underline{x^6 + x^4 + x^3} \\ x^4 \\ x^4 + x^2 + x \\ \underline{ + x^2 + x} \\ \boxed{x^2 + x} \end{array}} \quad \begin{array}{l} \text{Dividend:} \\ \text{augmented} \\ \text{dataword} \end{array} \quad \begin{array}{l} \text{Remainder} \end{array}$$

Polynomials

Shifting

Left Shifting:

- Used in cyclic codes to add extra zeros to the dataword. This creates space for check bits (remainder) after division, allowing error detection in the final codeword.

Right Shifting:

- Mostly used to remove extra bits or adjust data alignment in general binary operations, but it's not commonly used in the main cyclic encoding process.

Shifting left 3 bits: 10011 becomes 10011000

$x^4 + x + 1$ becomes $x^7 + x^4 + x^3$

Shifting right 3 bits: 10011 becomes 10

$x^4 + x + 1$ becomes x

Cyclic Code Encoder Using Polynomials

A **Cyclic Code Encoder Using Polynomials** is a method to **encode data for error detection in digital communication**. In this method, both the data and a special "generator" code are represented as polynomials.

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

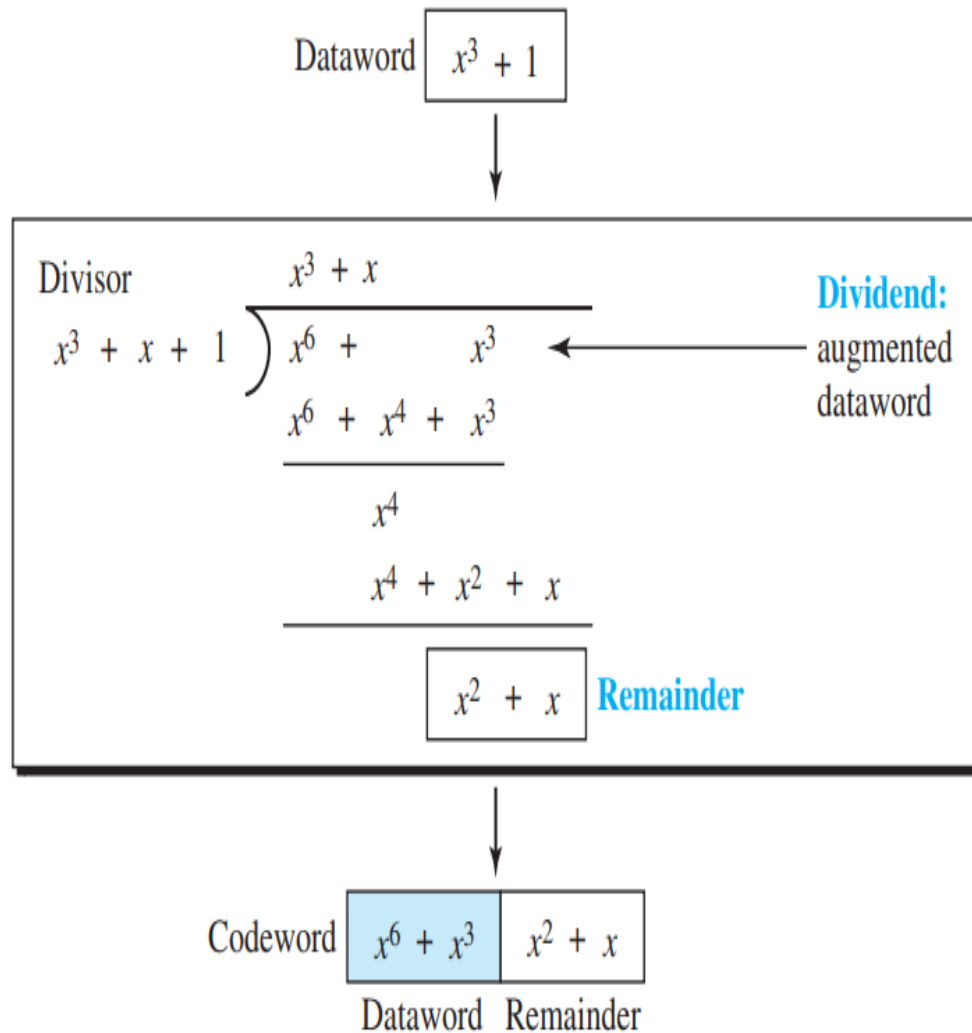
Example:

- The dataword 1001 is represented as $x^3 + 1$.
- The divisor 1011 is represented as $x^3 + x + 1$.

Cyclic Code Encoder Using Polynomials

- To find the **augmented dataword**, we have **left-shifted the dataword 3 bits** (multiplying by x^3). **1001+000**
- The result is $x^6 + x^3$.
- Divide the first term of the dividend, x^6 , by the first term of the divisor, x^3 .
- The first term of the quotient is then x^6/x^3 , or x^3 .
- Then we multiply x^3 by the divisor and subtract (according to our previous definition of subtraction) the result from the dividend.
- The result is x^4 , with a degree greater than the divisor's degree; we continue to divide until the degree of the remainder is less than the degree of the divisor.

Cyclic Code Encoder Using Polynomials



At the Receiver's End:

1. Receiver's Division:

1. The receiver divides the received codeword by the same divisor used by the sender.

2. Check for Remainder:

1. If the remainder is **zero**, it means no error is detected in the codeword, and the data is assumed to be correct.

2. If there is a **non-zero remainder**, it indicates an

error in transmission, and codeword was corrupted.

Cyclic Code Analysis

Analyze cyclic codes to detect or correct transmission errors using polynomial operations.

- **Dataword ($d(x)$):** Original data.
- **Codeword ($c(x)$):** Encoded data with check bits.
- **Generator ($g(x)$):** Polynomial used for encoding and error detection.
- **Syndrome ($s(x)$):** Remainder after dividing the received codeword by $g(x)g(x)g(x)$.
- **Error ($e(x)$):** Represents any errors in the transmitted data.
- **Error Detection:**
 - If $s(x) \neq 0$, is detected.
 - If $s(x) = 0$, Either no error or undetectable error.

Cyclic Code Analysis

Analyze cyclic codes to detect or correct transmission errors using polynomial operations.

- **Dataword ($d(x)$):** Original data.
- **Codeword ($c(x)$):** Encoded data with check bits.
- **Generator ($g(x)$):** Polynomial used for encoding and error detection.
- **Syndrome ($s(x)$):** Remainder after dividing the received codeword by $g(x)$.
- **Error ($e(x)$):** Represents any errors in the transmitted data.
- **Error Detection:**
 - If $s(x) \neq 0$, is detected.
 - If $s(x) = 0$, Either no error or undetectable error.

Cyclic Code Analysis

1. Single-Bit Error Analysis

- **Represent the Error:** A single-bit error at position i is represented as $e(x) = x^i$.
- **Divide by Generator Polynomial:** Divide $e(x)$ by the generator polynomial $g(x)$.
- **Check Remainder:**
 - If there's a non-zero remainder, the error is detected.
 - If the remainder is zero, the error goes undetected.
- **Typical Result:** A well-designed $g(x)$ with more than one term and a constant term of 1 generally detects all single-bit errors.

Cyclic Code Analysis

2. Burst Error Analysis

- **Define Burst Length:** Determine the length k of the burst error.
- **Represent the Burst Error:** For a burst of length k starting at position i , represent it as $e(x) = x^i + x^{i+1} + \dots + x^{i+k-1}$.
- **Divide by Generator Polynomial:** Perform polynomial division of $e(x)$ by $g(x)$.
- **Check Remainder:**
 - If there's a non-zero remainder, the burst error is detected.
 - If the remainder is zero, the burst error goes undetected.
- **Typical Result:** A generator polynomial of $k + 1$ bits can generally detect burst errors up to length k . Burst errors longer than k may or may not be detected, depending on $g(x)$.

Cyclic Code Analysis

3. Two Isolated Bit Errors

- **Represent Error:** Two isolated bit errors at positions i and j can be represented as $e(x) = x^i + x^j$.
- **Detection Condition:** The generator polynomial $g(x)$ must not divide $x^{j-i} + 1$ (the difference between the error positions) for the error to be detected.
- **Typical Result:** If $g(x)$ has more than one term and ends with 1, it can generally detect two isolated bit errors at most positions.

Example:

Errors at positions 3 and 7 are represented as x^3 and x^7 , respectively. The error polynomial is $e(x) = x^3 + x^7$. The difference between these positions is $7 - 3 = 4$, which is represented as $x^4 + 1$. This difference helps determine if the errors are detectable based on the generator polynomial.

Cyclic Code Analysis

4. Odd Number of Errors

- **Detection Condition:** If the generator polynomial $g(x)$ contains $x + 1$ as a factor, it can detect any error pattern with an odd number of bit errors.
- **Typical Result:** Most well-designed generator polynomials include $x + 1$ to ensure all odd-numbered errors (like 1, 3, 5, etc.) are detected.

Detecting odd-numbered errors(3 bit or 5 bit flipped) depends significantly on the chosen polynomial.

Advantages of Cyclic Codes

- Cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors.
- They can easily be implemented in hardware and software.
- The mathematical basis of cyclic codes allows for the use of polynomials, which simplifies the analysis of their properties and performance
- **Codeword: 1101011011 divisor $x^3 + x + 1$ i.e, 1011**

Hardware Implementation of Cyclic Codes

- **Advantages:** Cyclic code hardware implementation is easy, cheap, and improves the rate of check bit and syndrome bit calculation.

- **Divisor:**

1. XORed with part of the dividend.
2. Divisor has fixed bits ($n - k + 1$) like 1011 or 0000.
3. Only $n - k$ bits of the divisor are used for XOR.

- **Augmented Dataword:** Divisor bits shift left to align with the augmented dataword; no need to store the dataword.

- **Remainder:**

1. Use 3-bit registers for the remainder.
2. XOR remainder and augmented dataword bits for final result.

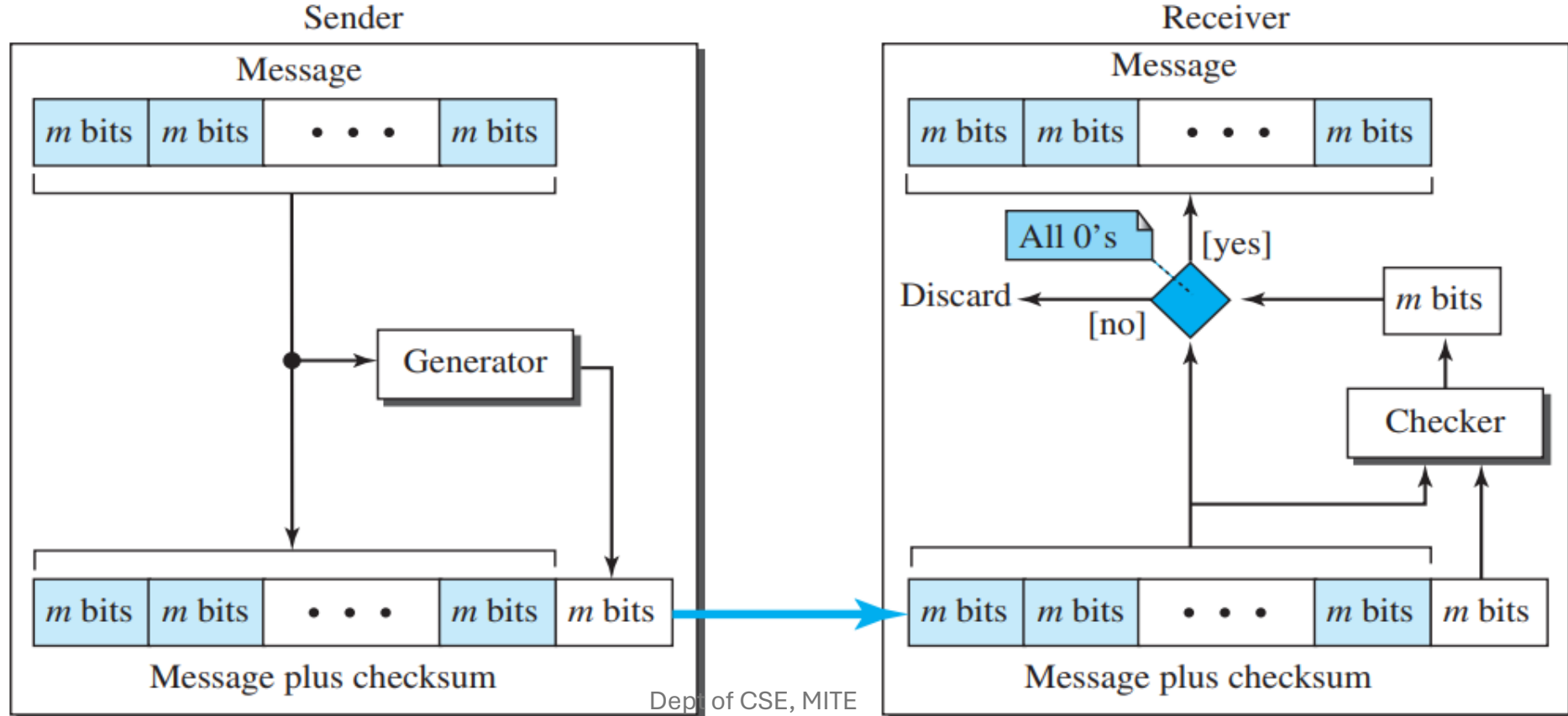
- **Simplifications:** Only need the final remainder bits, 3 registers, and 2 XOR devices for efficiency.

- **General Design:**

- Use $n - k$ shift registers and XOR devices in both encoder and decoder.
- After k steps, check bits are ready, but full codeword is required for the decoder.

CHECKSUM

Checksum is an **error-detecting technique** that can be **applied to a message of any length**. In the Internet, the **checksum technique is mostly used at the network and transport layer rather than the data-link layer**.



CHECKSUM

Communication between a sender and receiver.

Sender Side:

1.Message Segmentation: The message is divided into multiple units of *m bits*.

2.Checksum Generation: A generator creates an additional *m-bit* unit called the **checksum**.

3.Message Transmission: The original message and the checksum are combined and sent together as a package called "Message plus checksum."

CHECKSUM

Communication between a sender and receiver.

Receiver Side:

1.Message Reception: The receiver gets the message along with the checksum.

2.New Checksum Calculation: A checker at the receiver recomputes the checksum based on the received message and its checksum.

3.Comparison:

1. If the newly calculated checksum results in all 0s, the message is **accepted** as error-free.

2. If the **checksum does not result in all 0s**, the message is **discarded** as corrupted

CHECKSUM

The idea of the traditional checksum is simple.

The traditional checksum works by **sending the sum of the data along with the data itself.**

For example, **if we send five 4-bit numbers (7, 11, 12, 0, 6), we also send their sum, 36.**

The receiver adds the numbers and checks if the result matches the sent sum.

If they match, the data is accepted; if not, it's rejected. However, since we're dealing with 4-bit numbers, **36 exceeds the 4-bit limit, creating a problem. So, One's Complement Addition is used to convert it to 4 bits.**

CHECKSUM

One's Complement Addition

- **One's Complement** is a way to handle this issue.
- In **one's complement arithmetic**, if the sum has more bits than allowed (in this case, more than 4 bits), the extra leftmost bits are **wrapped around** and added to the rightmost bits.
- For example, the binary value **100100** is 6 bits long. The **extra** leftmost 2 bits (10) are added to the remaining 4 bits (0100), like this:
 - **100100** → **0100** (remaining 4 bits) + **10** (extra bits).
 - Adding 0100 and 10 gives 0110, which is 6.

So, instead of sending 36 as the sum, you can send **6**, which fits into 4 bits.

$$(10)_2 + (0100)_2 = (0110)_2 \rightarrow (6)_{10}$$

Dept. of CSE, MIT

CHECKSUM

Internet Checksum: Traditionally, the Internet has used a 16-bit checksum. The sender or the receiver uses five steps.

Internet Checksum calculation using the provided data
(0x12, 0x34, 0x56, 0x78): 00010010, 00110100, 01010110, 01111000

- Pair the Bytes:**

Combine 0x12 and 0x34 into 0x1234, and 0x56 and 0x78 into 0x5678.

- Add the 16-bit Words:**

Add 0x1234 and 0x5678 using one's complement addition:

CHECKSUM

$$0x1234 + 0x5678 = 0x68AC$$

- **Take the One's Complement (convert 0's to 1's and vice versa and change it to hexadecimal):** Invert the bits of the result: $\sim 0x68AC = 0x9753$

- **Final Checksum:**

The checksum is 0x9753. This value would be used to verify data integrity.

- **Verification upon Receiving Data:**

When the recipient gets the data, they perform a verification by summing the original data (in this case, 0x1234 and 0x5678) along with the checksum (0x975F).

- **Check the Sum:** The recipient computes: $0x1234 + 0x5678 + 0x9753 = 0xFFFF$

CHECKSUM

Interpretation:

If the sum equals **0xFFFF** (which is all bits set to 1 in a 16-bit value), it indicates that no errors occurred during transmission. After complementing it, the result becomes **0**.

<i>Sender</i>	<i>Receiver</i>
<ol style="list-style-type: none">1. The message is divided into 16-bit words.2. The value of the checksum word is initially set to zero.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.	<ol style="list-style-type: none">1. The message and the checksum are received.2. The message is divided into 16-bit words.3. All words are added using one's complement addition.4. The sum is complemented and becomes the new checksum.5. If the value of the checksum is 0, the message is accepted; otherwise, it is rejected.

CHECKSUM

- **Traditional Checksum:** While it is **fast and easy to implement**, it has **limited error detection capabilities**, particularly **against burst errors**. It is **suitable for simpler applications** but may not be sufficient for critical data integrity requirements.

- **CRC:** Offers **superior error detection capabilities**, especially **for burst errors**, making it ideal for applications where data integrity is crucial. The trade-off is slightly increased complexity and computational overhead, but this is manageable with modern hardware.

Chapter 2-

DATA LINK CONTROL

DLC SERVICES

- Data link Layer is **divided into Logical Link Control (DLC) and MAC.**
- The data link control (DLC) deals with **procedures for communication between two adjacent nodes i.e. node-to-node communication.**
- Data link control functions include
 - 1) Framing
 - 2) Flow control and
 - 3) Error control.

Framing

- A frame is **a group of bits**. Framing means organizing the bits into a **frame that are carried by the physical layer**.
- The data-link-layer needs to form frames, so that **each frame is distinguishable from another**.
- Framing separates a message from other messages by **adding sender-address & destination-address**.
- The destination-address defines where the packet is to go.
- The sender-address helps the recipient acknowledge the receipt.

Framing

- Q: Why the whole message is not packed in one frame?
- Ans: Large frame makes flow and error-control very inefficient.

Even a single-bit error requires the re-transmission of the whole message.

- When a message is divided into smaller frames, a single-bit error affects only that small frame.

Framing

- 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**.
- In addition, each **envelope defines the sender and receiver addresses** since the postal system is a many-to-many carrier facility

Framing

Frame Size

- Two types of frames:

1) Fixed Size Framing

- There is **no need for defining boundaries of frames**; the **size itself can be used as a delimiter**. For example: ATM WAN uses frames of fixed size called cells.

2) Variable Size Framing

- Variable-size frames **do not have a predefined length**.
- **We need to define the end of the frame and the beginning of the next frame.**

Framing

Two approaches are used in **Variable Size Framing** :

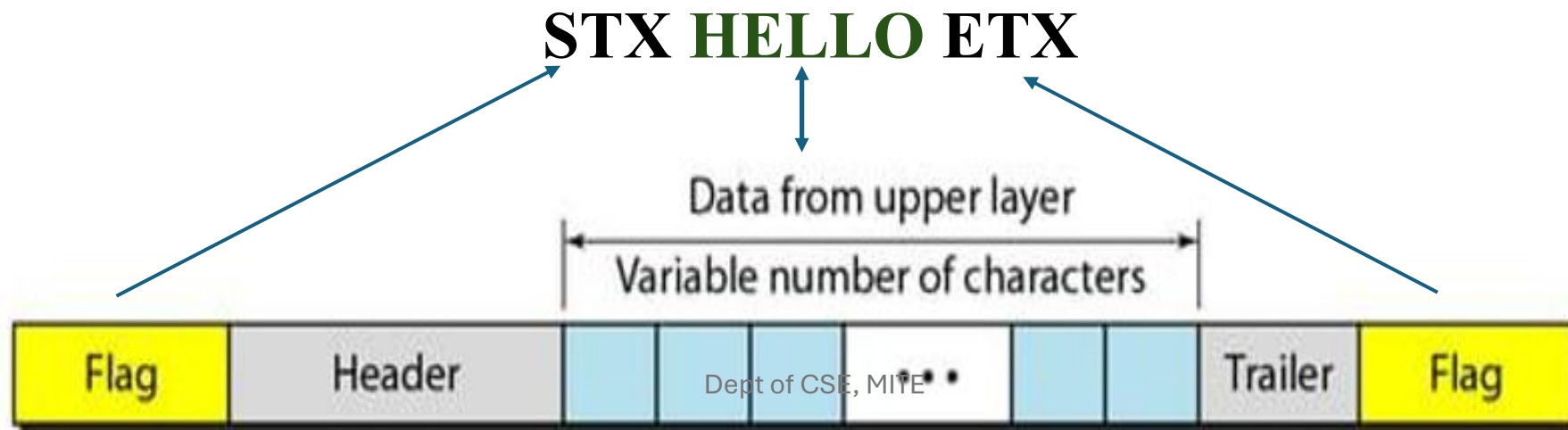
- 1) Character-oriented approach
- 2) Bit-oriented approach.

Character Oriented Framing

- Data to be carried are 8-bit characters from a coding system such as ASCII.
- The header and the trailer are also multiples of 8 bits.

Character Oriented Framing

- **Header carries the source and destination-addresses and other control information.**
- **Trailer carries error-detection or error-correction redundant bits.**
- To separate one frame from the next frame, an 8- bit (I-byte) **flag is added at the beginning and the end of a frame.**
- **The flag signals the start or end of a frame.**



Character Oriented Framing

- **Problem:**
- Character-oriented framing is suitable **when only text is exchanged by the data-link-layers**. However, if we send another type of information (say audio/video), then **any pattern used for the flag can also be part of the information**.
- If the **flag-pattern appears in the data-section, the receiver might think that it has reached the end of the frame**.
- **Solution: A byte-stuffing is used**

Character Oriented Framing

byte-stuffing or character stuffing :

- In byte stuffing, a **special byte is added to the data-section** of the frame when there is a character with the same pattern as the flag.
- The **data-section is stuffed with an extra byte**. This byte is called the **escape character (ESC)**, which has a predefined bit pattern.
- When a **receiver encounters the ESC character**, the receiver **removes ESC character from the data-section and treats the next character as data, not a delimiting flag**.

Character Oriented Framing

byte-stuffing or character stuffing :

1. Original Data:

- You send: STX Hello ESC ETX ETX .

2. What the Receiver Sees:

- The receiver sees the full message: STX Hello ESC ETX ETX .

3. Processing the Message:

- The receiver starts reading at STX, indicating the start of the frame.
- It reads Hello as the actual data.
- Upon encountering ESC, the receiver recognizes that the next character (which is ETX) should be treated as data and **not** as a flag.

4. Final Interpretation:

- The receiver interprets the message as:
 - **Data:** Hello ESC ETX
- The last ETX (not preceded by ESC) is treated as the **actual end of the frame**.

Character Oriented Framing

byte-stuffing or character stuffing :

- What happens if the **text contains one or more escape characters followed by a flag?**
- The receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame.

Solution:

- Escape characters part of the text must also be marked by another escape character .

Character Oriented Framing

byte-stuffing or character stuffing :

- **Scenario: Two Escape Characters**
 - **Message Sent:** STX Hello ESC ESC ETX ETX .

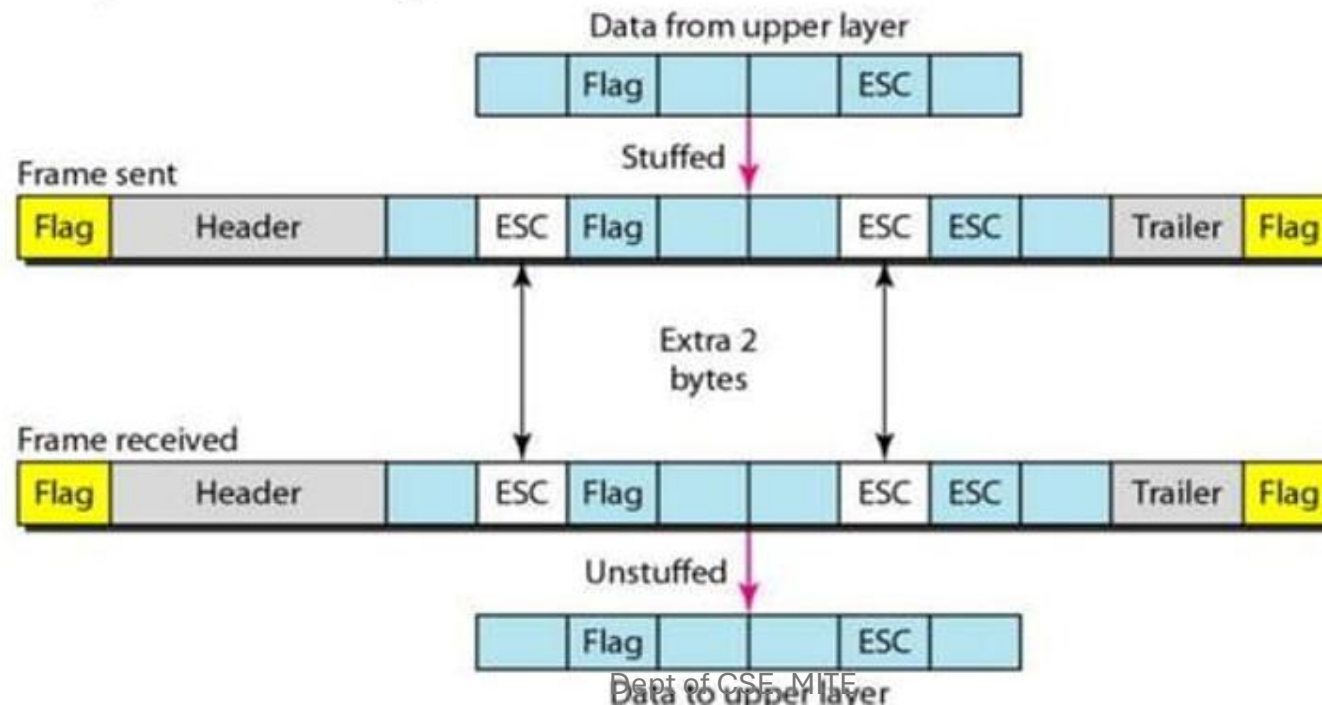
Interpretation:

1. **Start Frame:** STX indicates the start.
2. **Data:** The receiver reads Hello .
3. **First ESC :** Treats the next character as data (not a flag).
4. **Second ESC :** Again treats the next character as data.
5. **ETX :** The first ETX is treated as data (because of the preceding ESC).
6. **Last ETX :** Marks the actual end of the frame.

Character Oriented Framing

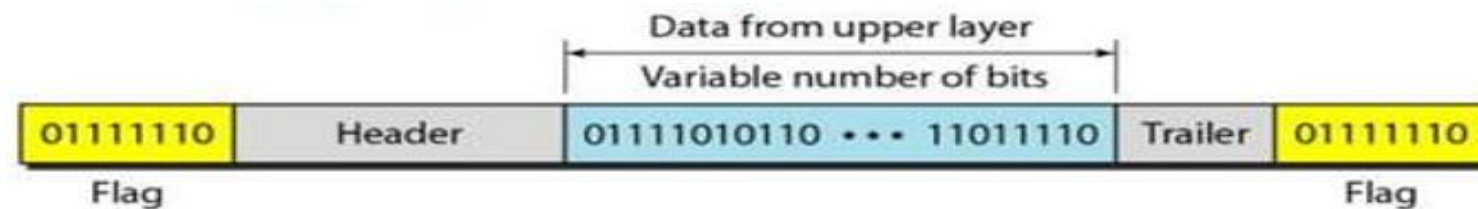
byte-stuffing or character stuffing :

- In short, byte stuffing is the **process of adding one extra byte whenever there is a flag or escape character in the text.**



Bit Oriented Framing

- The data-section of a frame is a **sequence of bits to be interpreted by the upper layer as text, audio, video, and so on.**
- However, in **addition to headers and trailers, we need a delimiter to separate one frame from the other.**
- Most protocols use a special 8-bit pattern **flag 01111110 as the delimiter** to define **the beginning and the end** of the frame



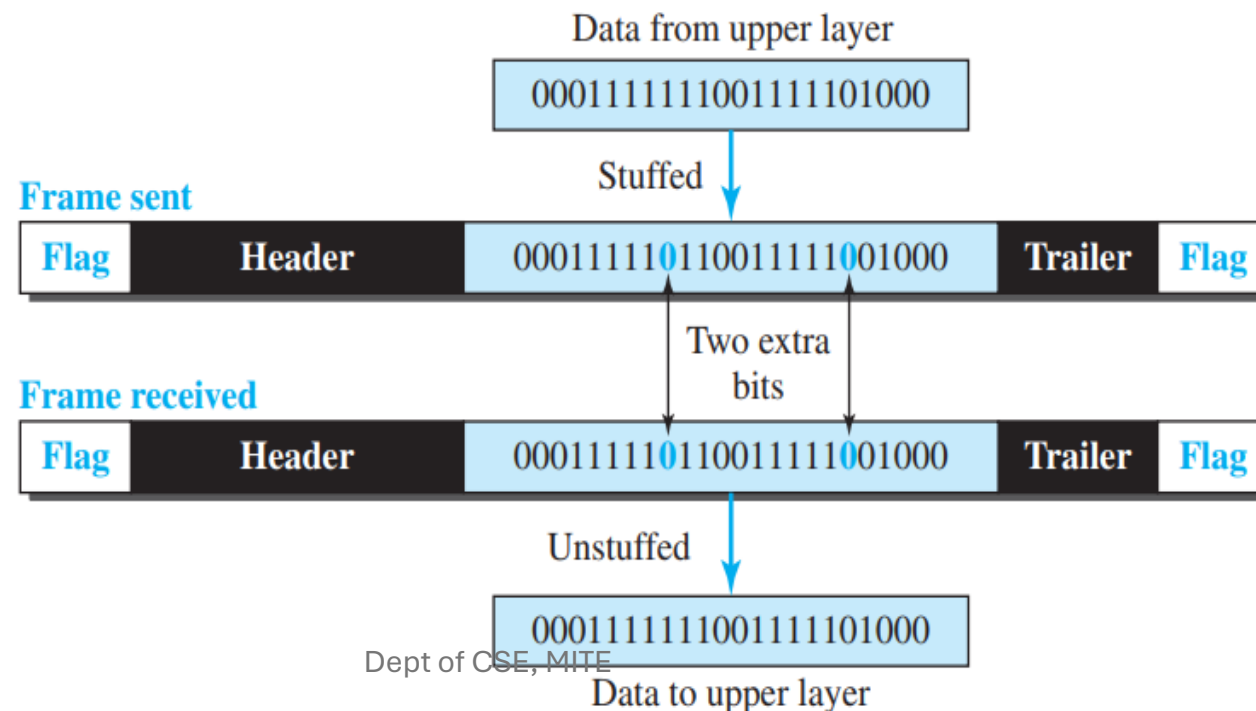
Bit Oriented Framing

- **Problem:**
- If the flag-pattern appears in the data-section, the receiver might think that it has reached the end of the frame.
- **Solution: A bit-stuffing is used.**
- In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver.

Bit Oriented Framing

Bit-stuffing

- This guarantees that the flag field sequence does not inadvertently appear in the frame.
- In short, 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 01111



Flow Control and Error Control

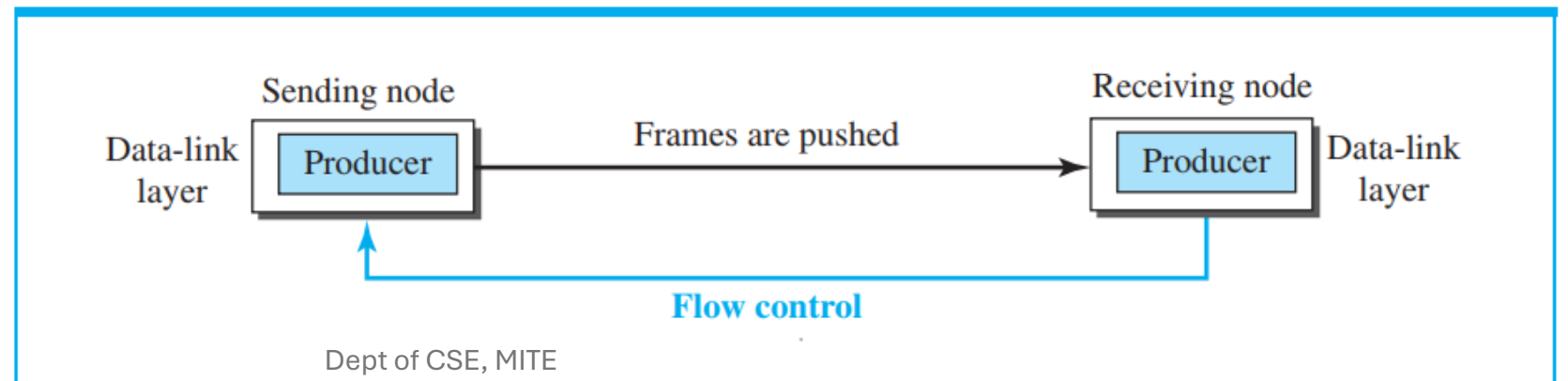
- One of the **responsibilities of the DLC sublayer is flow and error control at the data-link layer.**

Flow Control

- Whenever an **entity produces items and another entity consumes them, there should be a balance between production and consumption rates.**
- If the **items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items.**
- **We need to prevent losing the data items at the consumer site.**

Flow Control

- At the sending node, the **data-link layer** tries to **push frames toward the data-link layer at the receiving node**
- If the **receiving node cannot process and deliver the packet** to its network at the same rate that the frames arrive, it **becomes overwhelmed with frames**.
- Here, **flow control can be feedback from the receiving node to the sending node to stop or slow down pushing frames**.



Buffers

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.**
- Normally, two buffers can be used.
 - 1) First buffer at the sender.
 - 2) Second buffer at the receiver.
- **The flow control communication can occur by sending signals from the consumer to the producer.**
- **When the buffer of the receiver is full, it informs the sender to stop pushing frames.**

Error Control

- Error-control includes **both error-detection and error-correction.**
- Error-control **allows the receiver to inform the sender of any frames lost/damaged in transmission.**
- A CRC is **added to the frame header** by the sender and → checked by the receiver.

Error Control

- At the data-link layer, **error control** is normally **implemented using one of the following two methods.**
- 1) **First method:** If the **frame is corrupted, it is discarded;**
- If the **frame is not corrupted, the packet is delivered to the network layer.** This method is used **mostly in wired LANs such as Ethernet.**

Error Control

- 2) **Second method:** If the frame is corrupted, it is discarded;
- If the frame is not corrupted, an acknowledgment is sent to the sender.
- **Acknowledgment** is used for the purpose of both flow and error control.

Combination of Flow and Error Control

- **Flow and error control can be combined.**
- **The acknowledgment that is sent for flow control can also be used for error control to tell the sender the packet has arrived uncorrupted.**
- **The lack of acknowledgment means that there is a problem in the sent frame.**
- **A frame that carries an acknowledgment is normally called an ACK to distinguish it from the data frame.**

Original message: Data

Acknowledgement message: ACK

Connectionless and Connection-Oriented

Connectionless and connection-oriented protocols are **two fundamental approaches to data transmission in networking.**

Connectionless Protocol

- Connectionless protocols **allow data to be sent from one device to another without establishing a dedicated connection beforehand.**
- Each packet of data, often referred to as a datagram, is **sent independently and may take different paths to reach its destination.**
- There is **no need for a preliminary connection setup, making the process faster but less reliable.**

Connectionless and Connection-Oriented

Connectionless Protocol

- **Frames are sent from one node to the next without any relationship between the frames; each frame is independent.**
- **The frames are not numbered and there is no sense of ordering.**
- **Most of the data-link protocols for LANs are connectionless protocols.**

Connectionless and Connection-Oriented

Connection Oriented Protocol

- **Connection-oriented protocols require a connection to be established between two devices before any data can be transmitted.**
- **This approach ensures that all packets are delivered in order and provides mechanisms for error recovery.**
- **A handshake process is used to establish a connection before data transmission begins.**
- **The protocol maintains the state of the connection throughout the communication session, allowing for reliable data transfer.**

Connectionless and Connection-Oriented

Connection Oriented Protocol

- If the frames are not received in order, the receiver needs to wait until all frames belonging to the same set are received and then deliver them in order to the network layer.
- Connection oriented protocols are rare in wired LANs, but we can see them in some point-to-point protocols, some wireless LANs, and some WANs.

DATA LINK LAYER PROTOCOLS

- Traditionally **2 protocols** have been defined for the data-link layer to deal with **flow and error control**:

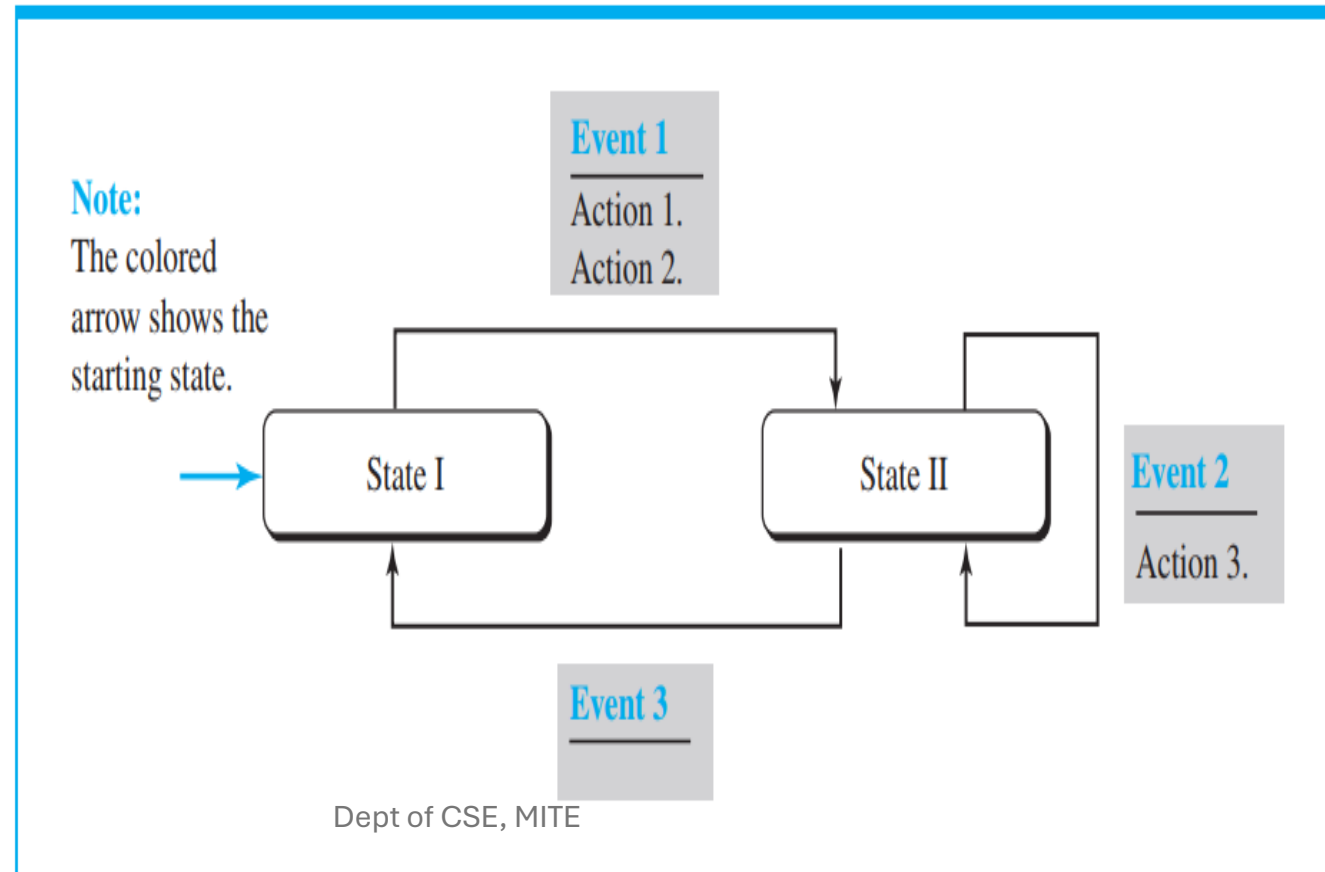
1) **Simple Protocol and**

2) **Stop-and-Wait Protocol.**

- The **behavior of a data-link-layer protocol** can be better shown as a **finite state machine (FSM)**.
- **FSM** helps break down a **system's behavior into manageable steps**.

DATA LINK LAYER PROTOCOLS

- The machine can be in **State I** or **State II**.
- When certain **events** happen, they trigger the machine to do specific **actions** and sometimes move to a different state.



DATA LINK LAYER PROTOCOLS

- how it works:

1. The machine starts in **State I** (shown by the arrow).

2.Event 1 happens in **State I**:

1. It causes the machine to do **Action 1** and **Action 2**.
2. Then, the machine moves to **State II**.

3.Event 2 happens in **State II**:

1. It triggers **Action 3**, but the machine stays in **State II**.

4.Event 3 could make the machine either go back to **State I** or stay in **State II**.

DATA LINK LAYER PROTOCOLS

- **Traffic light FSM** example:

States:

- **State I:** Green light (cars move).
- **State II:** Red light (cars stop).

Events:

- **Event 1:** Timer ends (green light duration).
- **Event 2:** Timer ends (red light duration).
- **Event 3:** Pedestrian button pressed.

Actions:

- **Action 1:** Turn light yellow (prepare to stop).
- **Action 2:** Turn light red (stop cars).
- **Action 3:** Turn light green (allow cars to move)

1) The light starts in **State I** (green light on).

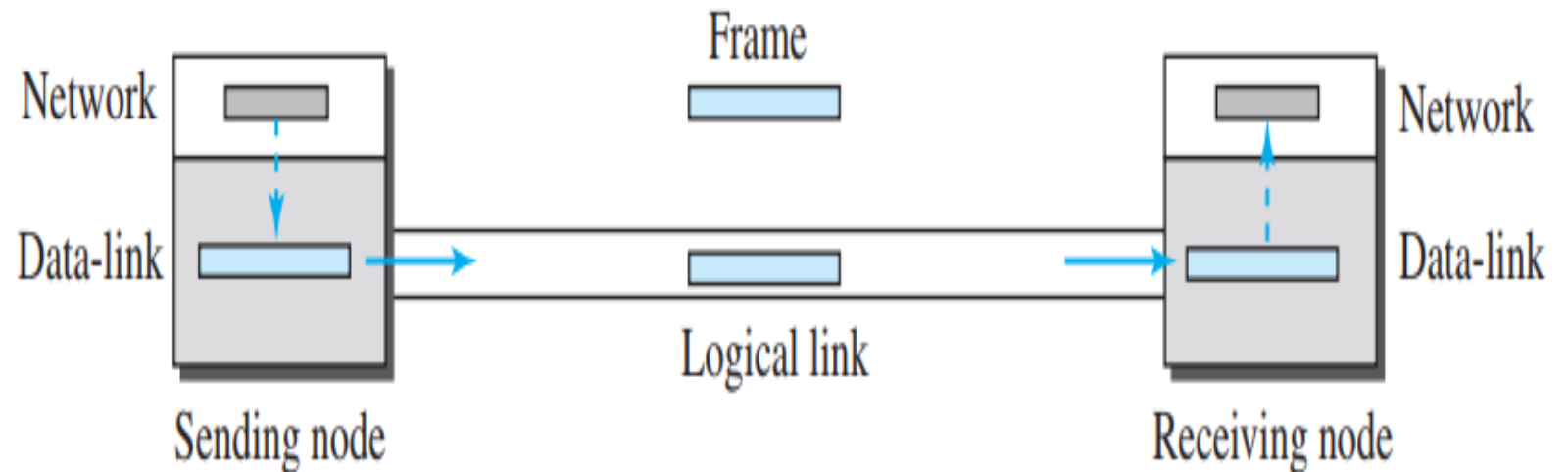
2) After a set time (**Event 1**), the light changes to yellow (**Action 1**) and then to red (**Action 2**), moving to **State II** (red light on).

3) After another set time (**Event 2**), the light changes back to green (**Action 3**), returning to **State I**.

4) If someone presses the pedestrian button (**Event 3**), the light might switch to red earlier to let people cross.

Simple Protocol

- This is the most **basic form of a data link layer protocol**. The protocol has **no flow-control or error-control**.
- The protocol is a **unidirectional protocol** (in which frames are traveling in only one direction). The receiver can immediately handle any frame it receives.



Simple Protocol

- How it works:**

The sender sends frames without waiting for an acknowledgment or checking whether the receiver can handle more data. The receiver simply accepts the data as it arrives.

- Use cases:**

This is typically used as a theoretical or conceptual starting point for understanding more complex protocols, not for practical data transmission in real-world networks.

Simple Protocol

- **Sender:**

Data-link-layers of sender & receiver provide transmission services for their network-layers.

- **Receiver:**

Data-link-layers use the services provided by their physical layers for the physical transmission of bits.

Simple Protocol

FSM (Finite state machine)s:

- Two main requirements:

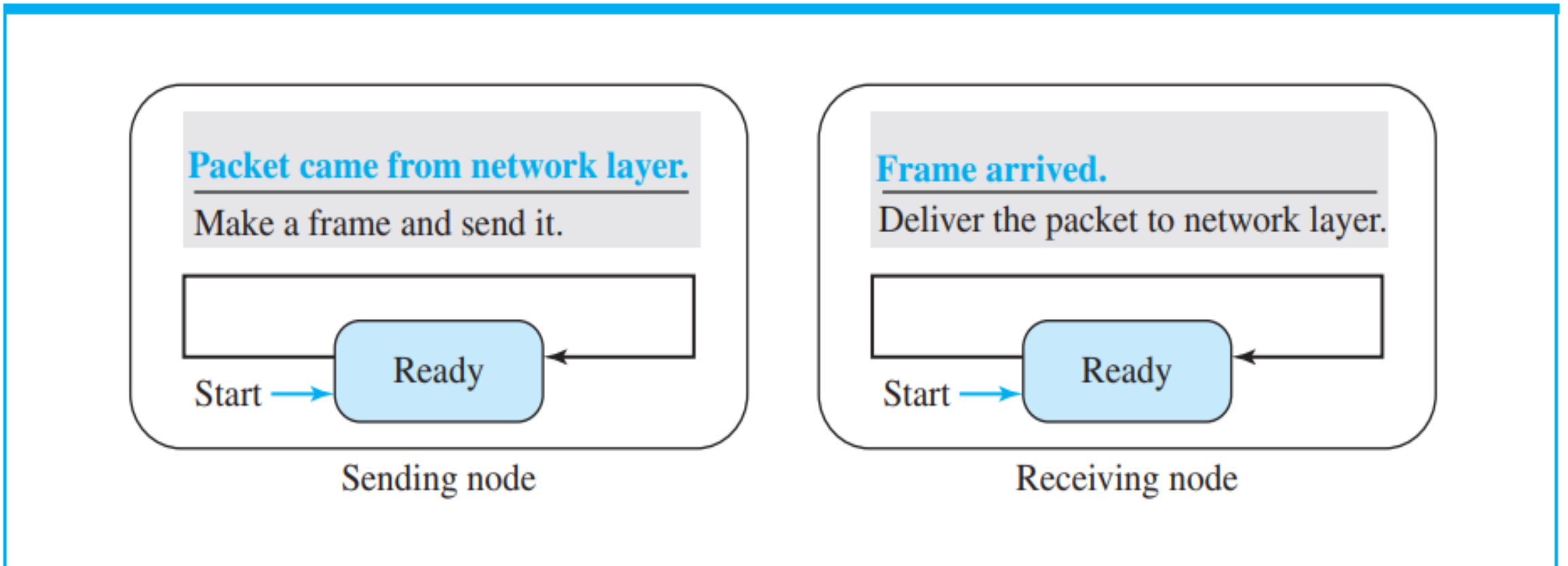
1) The **sender-site cannot send a frame until its network-layer has a data packet to send.**

2) The **receiver-site cannot deliver a data packet to its network-layer until a frame arrives.**

Simple Protocol

FSM (Finite state machine)s:

- Each FSM has only one state, the ready state.



Simple Protocol

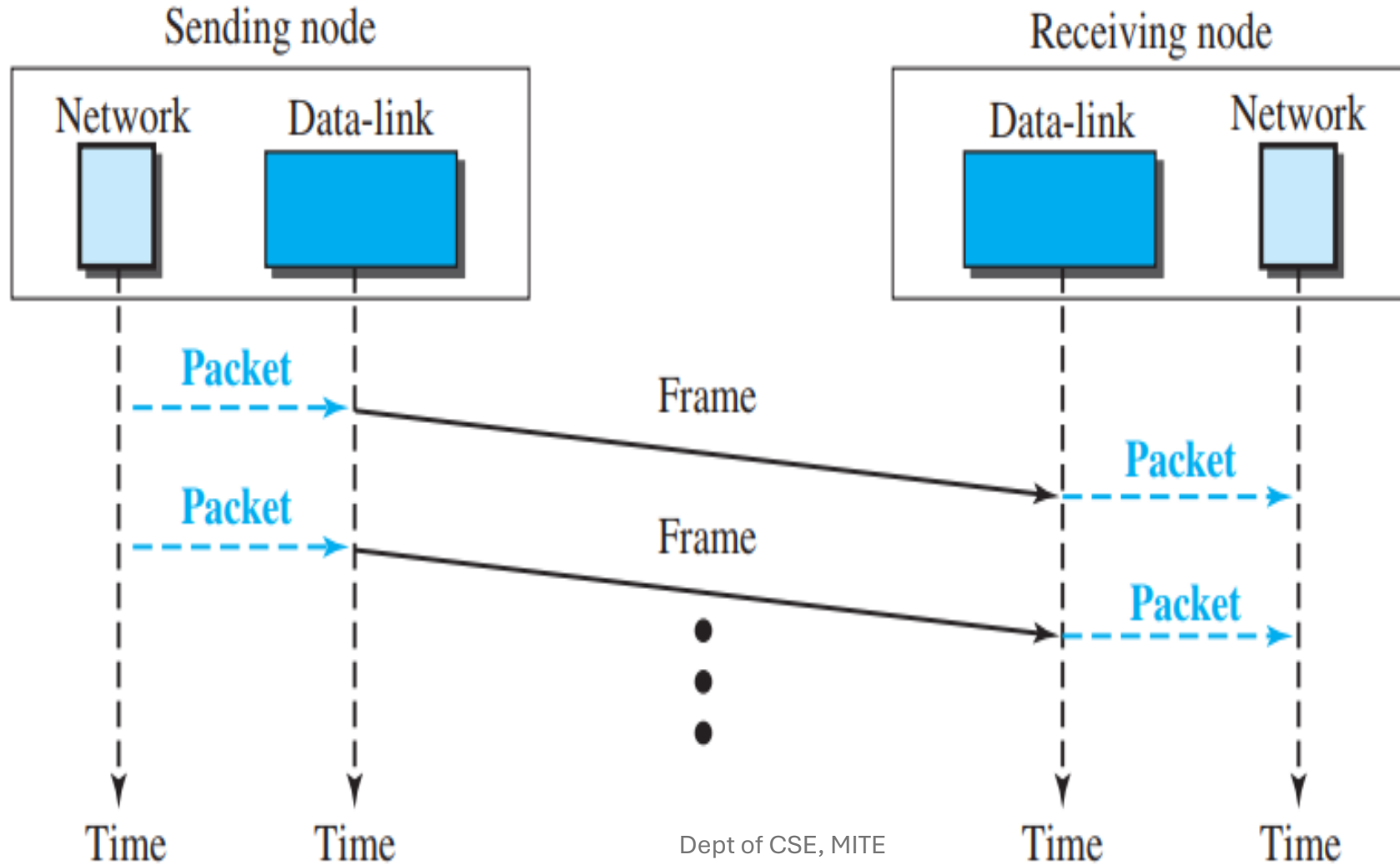
1) At Sending Machine

- The **sending machine remains in the ready state until a request comes from the process in the network layer.**
- When this event occurs, the **sending machine encapsulates the message in a frame and sends it to the receiving machine.**

2) At Receiving Machine

- The **receiving machine remains in the ready state until a frame arrives from the sending machine.**
- When this event occurs, the **receiving machine decapsulates the message out of the frame and delivers it to the process at the network layer.**

An example of communication using this protocol. It is very simple. The sender sends frames one after another without even thinking about the receiver.



Stop-and-Wait Protocol

- It is the **simplest flow control method**. In this, the **sender will transmit one frame at a time to the receiver**. The **sender will stop and wait for the acknowledgement from the receiver**.
- When the **sender gets the acknowledgement (ACK)**, it will **send the next data packet to the receiver and wait for the disclosure again**, and **this process will continue as long as the sender has the data to send**.

Stop-and-Wait Protocol

- **Packet Loss**

While sending the data from the sender to the receiver, the data flow needs to be controlled. If the sender is transmitting the data at a rate higher than the receiver can receive and process it, the data will get lost.

Using CRC

- Each data frame is assigned a Cyclic Redundancy Check (CRC) code to detect corruption. The receiver checks the frame using CRC.
- If the CRC indicates corruption, the frame is silently discarded by the receiver (no acknowledgment sent).

Stop-and-Wait Protocol

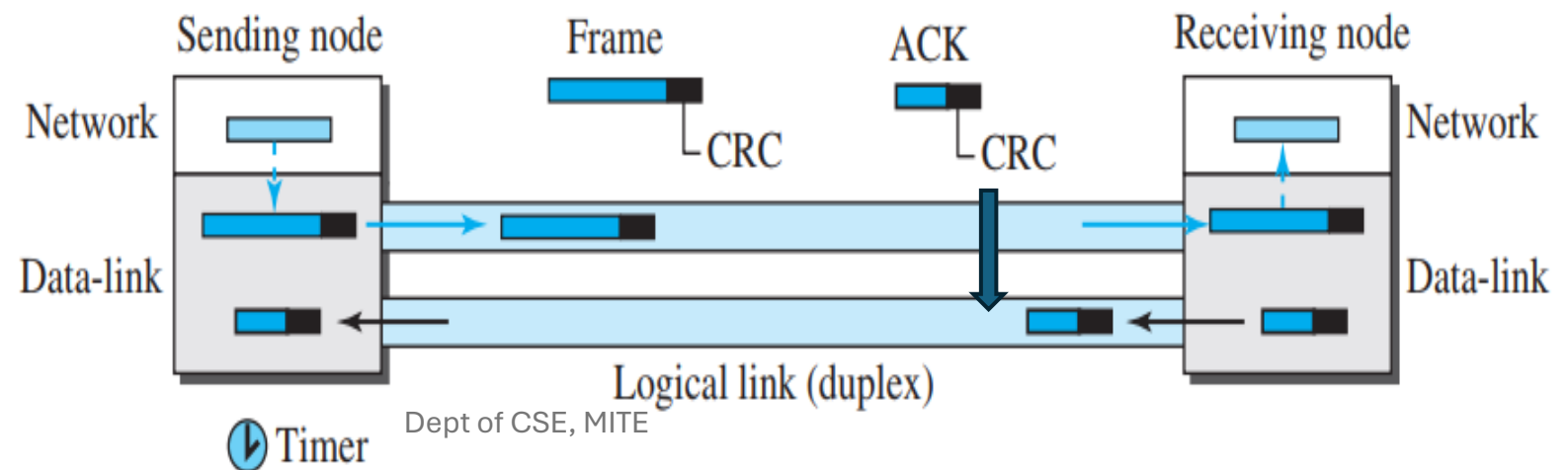
Timer

- **When the sender transmits a frame, it starts a timer to wait for acknowledgment (ACK).**
- **If the acknowledgment arrives before the timer expires, the sender stops the timer and sends the next frame.**
- **If the timer expires (due to missing ACK), the sender assumes the frame was lost or corrupted and retransmits it.**

Stop-and-Wait Protocol

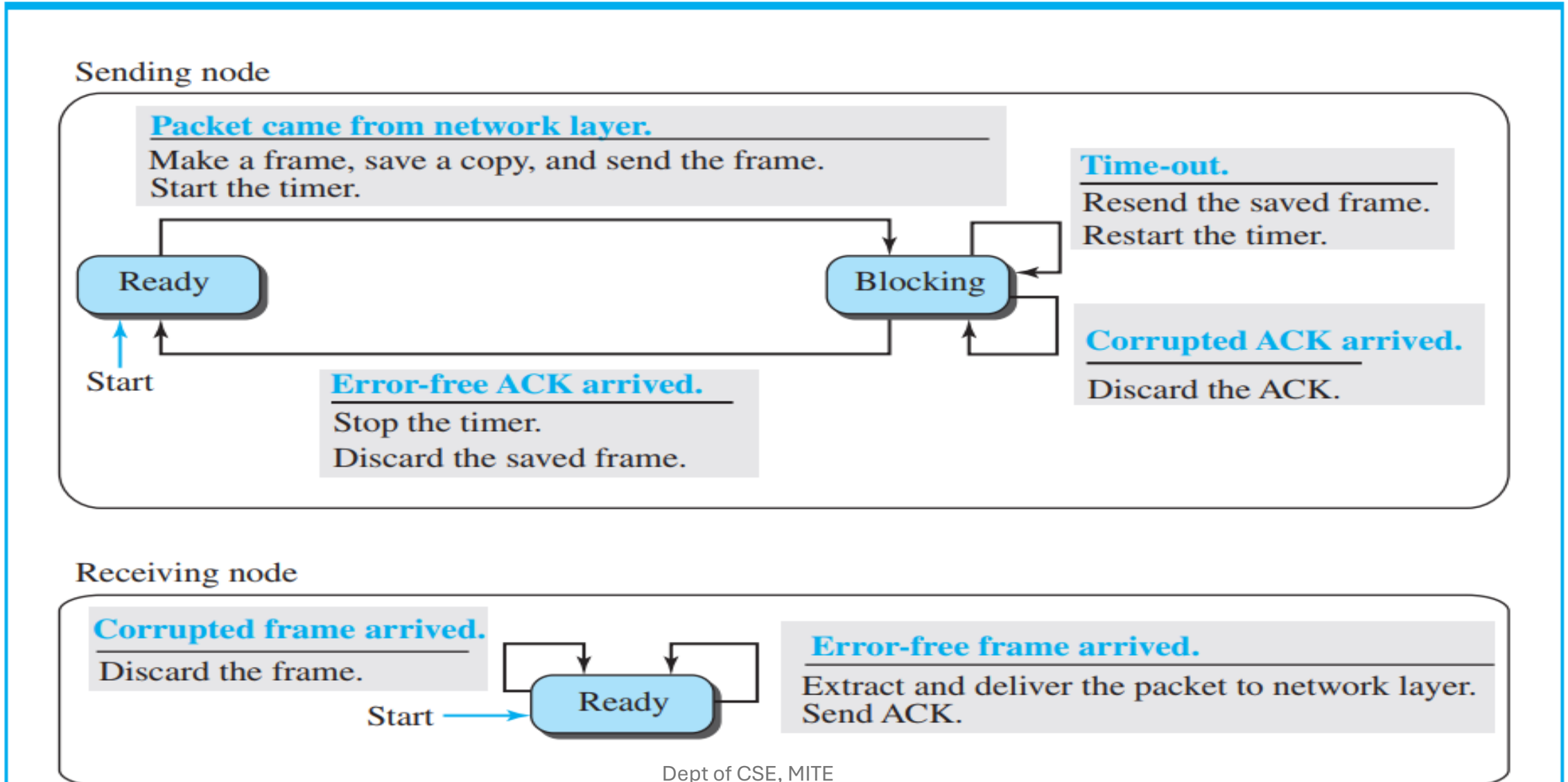
- The sender keeps a copy of the transmitted frame until it receives the corresponding acknowledgment.
- The protocol ensures that only one frame and one acknowledgment are in transit at any given moment (no concurrent frames).

Figure 11.10 Stop-and-Wait protocol



Stop-and-Wait Protocol

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



Stop-and-Wait protocol

- **Sender States**

- ✓ The sender is **initially in the ready state**, but it can move between the **ready and blocking state**.

Ready State.

- ✓ When the **sender is in this state**, it is **only waiting for a packet from the network layer**.
- ✓ If a packet comes from the network layer, the **sender creates a frame, saves a copy of the frame, starts the only timer and sends the frame**.
- ✓ The sender then **moves to the blocking state**.

Stop-and-Wait protocol

Blocking State. When the sender is in this state, **three events can occur:**

- a. If a time-out occurs,** the sender resends the saved copy of the frame and restarts the timer.
- b. If a corrupted ACK arrives,** it is discarded.
- c. If an error-free ACK arrives,** the sender stops the timer and discards the saved copy of the frame. It then moves to the ready state.

Stop-and-Wait protocol

Receiver

The **receiver is always in the ready state**. Two events may occur:

- a. **If an error-free frame arrives**, the message in the frame is delivered to the network layer and an ACK is sent.
- b. **If a corrupted frame arrives**, the frame is discarded.

Stop-and-Wait protocol

- **Sequence and Acknowledgment Numbers**

✓ If a **packet is duplicated**, the receiver might **treat it as a new packet**, leading to issues like placing duplicate orders (as in the online order example). **Duplicate packets need to be avoided just like corrupted ones.**

✓ Adding **Sequence Numbers**: To **avoid duplicates**, **sequence numbers are added to each data frame**. These **numbers alternate between 0 and 1**. So, the sequence would be: 0, 1, 0, 1, 0, 1...

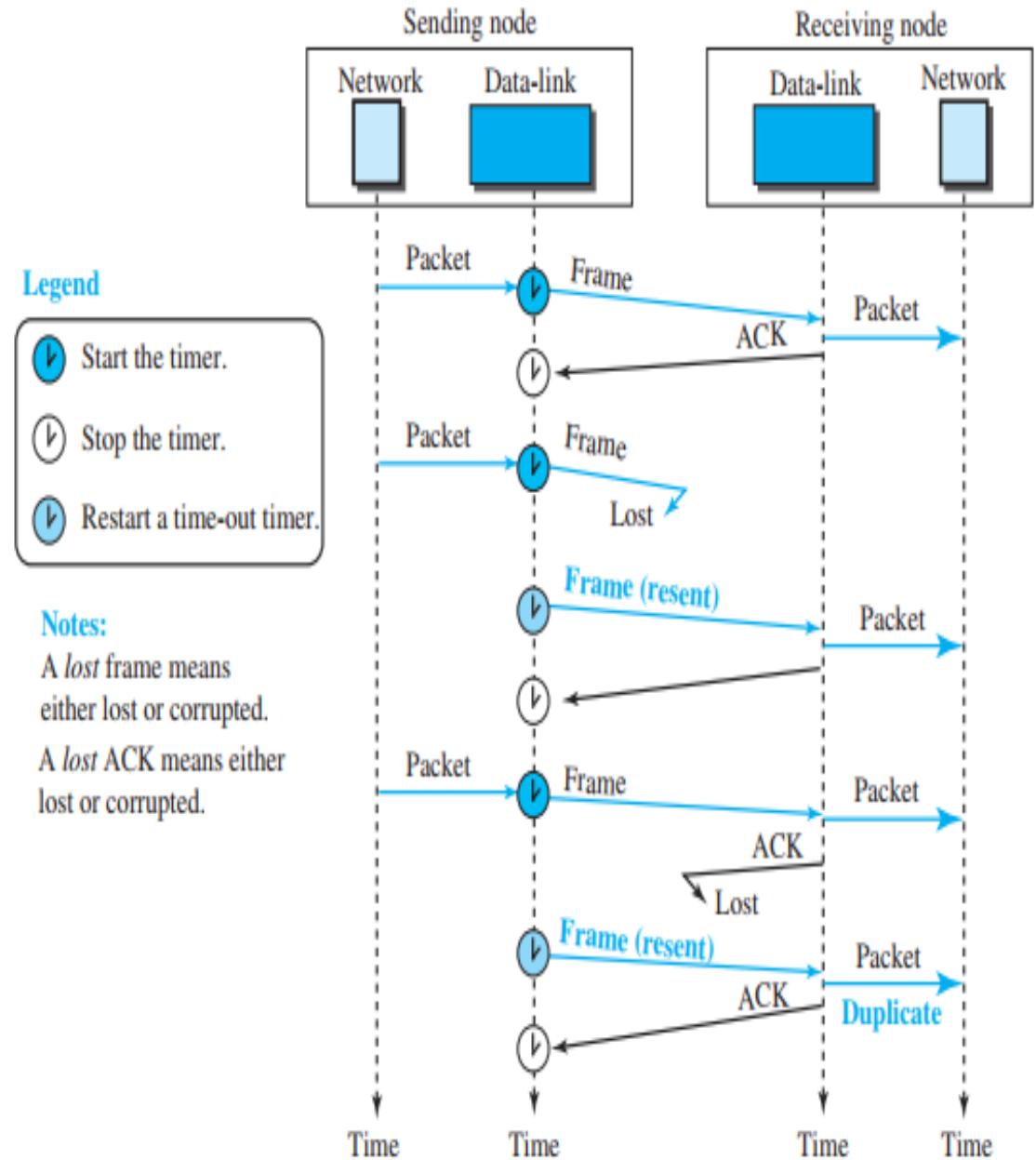
Stop-and-Wait protocol

- **Sequence and Acknowledgment Numbers**

- ✓ Similarly, **acknowledgment (ACK) frames also use alternating numbers, but they start with 1.** The **acknowledgment number in an ACK frame indicates the sequence number of the next frame expected from the sender.**
- ✓ **The sequence number in the data frame helps the receiver identify whether the frame is new or a retransmission.**
- ✓ **The acknowledgment number sent back to the sender indicates which frame should come next.**

Stop-and-Wait protocol

- Figure 11.12 shows an example.
- The **first frame** is sent and acknowledged.
- The **second frame** is sent, **but lost**. After time-out, it is resent.
- The **third frame** is sent and acknowledged, but the **acknowledgment is lost**. The frame is resent. However, there is a problem with this scheme. The network layer at the receiver site receives two copies of the third packet, which is not right.



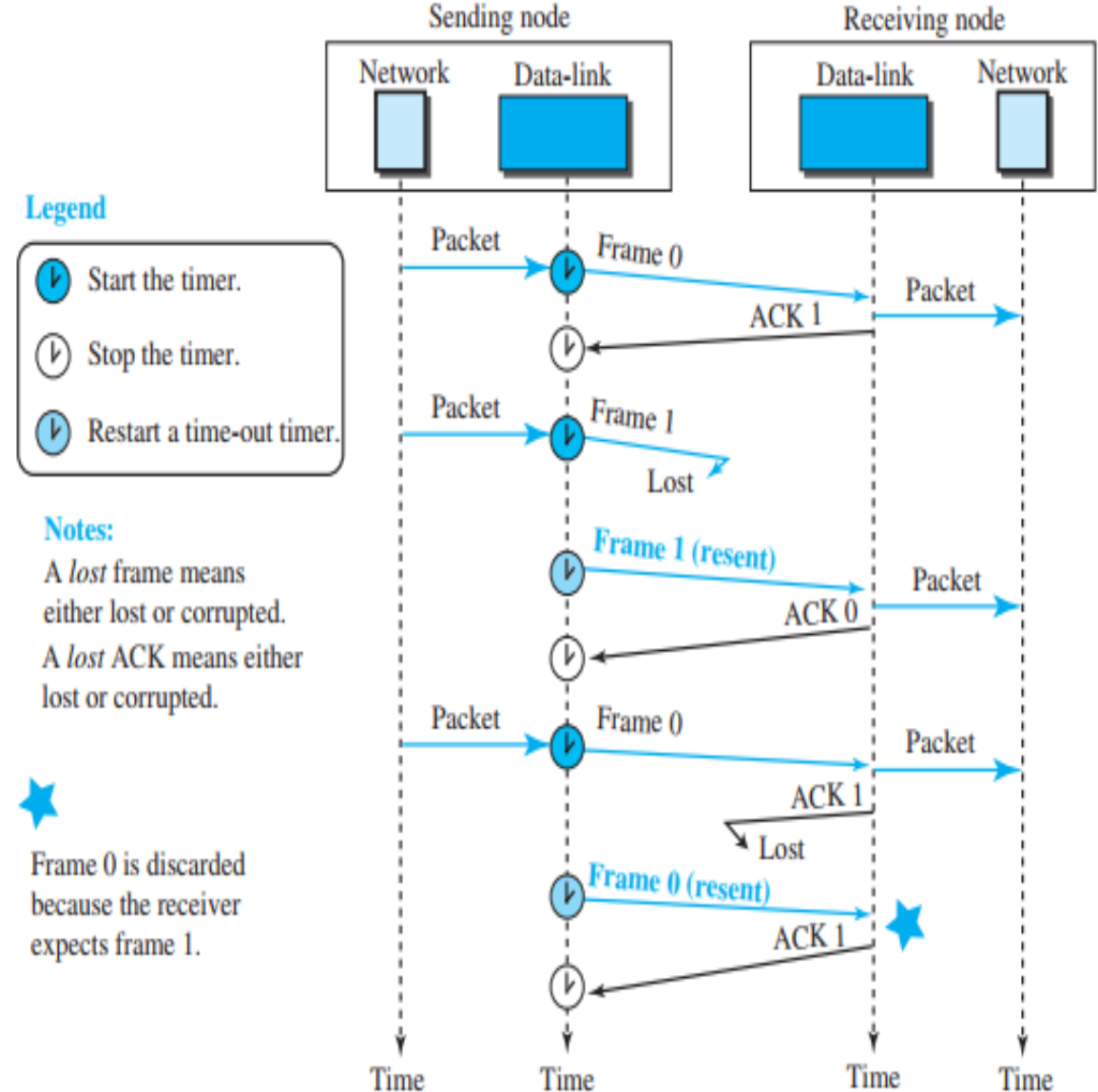
Stop-and-Wait protocol

Adding sequence numbers and acknowledgment numbers can prevent duplicates.

The first frame is sent and acknowledged. The second frame is sent, but lost. After time-out, it is resent.

The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent.

Figure 11.13 Flow diagram for Example 11.4



Piggybacking

A technique called piggybacking is used to **improve the efficiency of the bidirectional protocols.**

- The data in **one direction is piggybacked with the acknowledgment in the other direction.**
- In other words, when node A is sending data to node B, Node A also acknowledges the data received from node B.

High-level Data Link Control (HDLC)

- It is a type of **communication protocol** used for **transmitting data over networks**.
- Its main job is to **ensure reliable communication between devices, making sure that data is sent and received correctly**.
- HDLC is a **bit-oriented protocol** for communication **over point-to-point and multipoint links**.

High-level Data Link Control (HDLC)

Configurations and Transfer Modes

HDLC provides 2 common transfer modes that can be used in different configurations:

- 1) Normal response mode (NRM)
- 2) Asynchronous balanced mode (ABM).

High-level Data Link Control (HDLC)

1) Normal response mode (NRM)

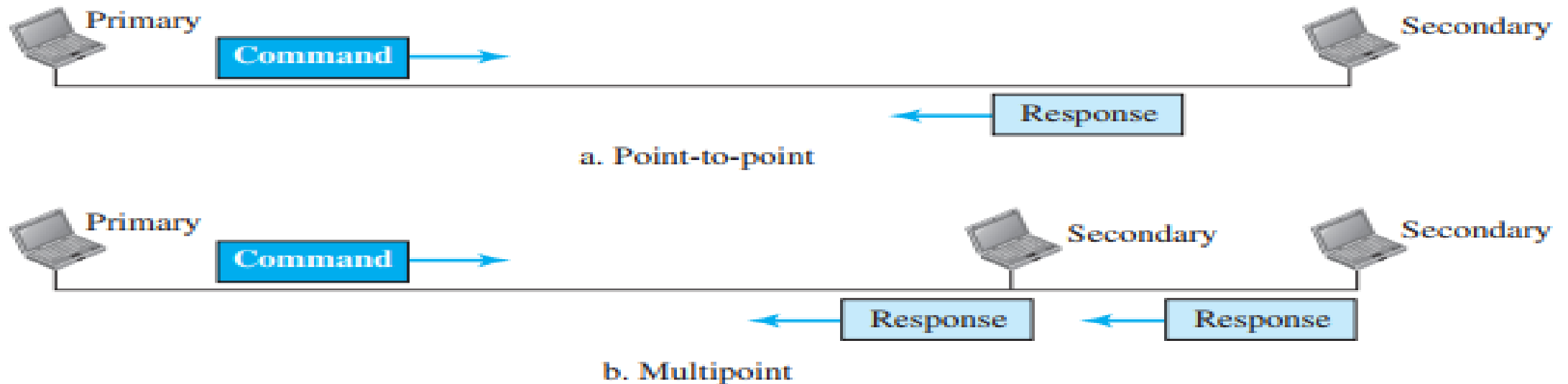
- It is a **communication protocol used in data link layer networking**, particularly in systems that **involve a primary station and one or more secondary stations**.
- We have one primary station and multiple secondary stations.
- **Sender** sends commands to the secondary stations.
- Initiates communication and requests responses.
- **Receiver** receives commands from the primary station and responds accordingly.
- Cannot initiate communication on its own.
- **NRM** can use a polling method, where the **primary station sends commands to multiple secondary stations**. This can reduce waiting time compared to **Stop-and-Wait**.

High-level Data Link Control (HDLC)

1) Normal response mode (NRM)

- The NRM is used for both point-to-point and multiple-point links.

Figure 11.14 *Normal response mode*



In this **unbalanced configuration**, the **primary station manages the communication process**, while **secondary stations respond only when prompted**.

High-level Data Link Control (HDLC)

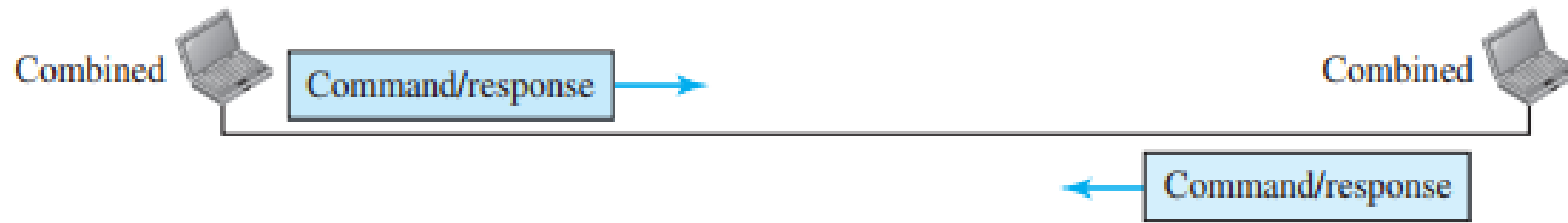
2) Asynchronous Balanced Mode (ABM) is a communication protocol used in data link layer networking that allows for flexible and efficient communication between multiple devices.

- In ABM, all stations (devices) have equal rights. There is no designated primary or secondary station.
- Each station can both send and receive messages independently, allowing for peer-to-peer communication.

High-level Data Link Control (HDLC)

2) Asynchronous Balanced Mode (ABM)

Figure 11.15 *Asynchronous balanced mode*



ABM allows all stations to send and receive messages independently, reducing the idle time inherent in Stop-and-Wait. It promotes efficient data transfer as multiple stations can communicate simultaneously.

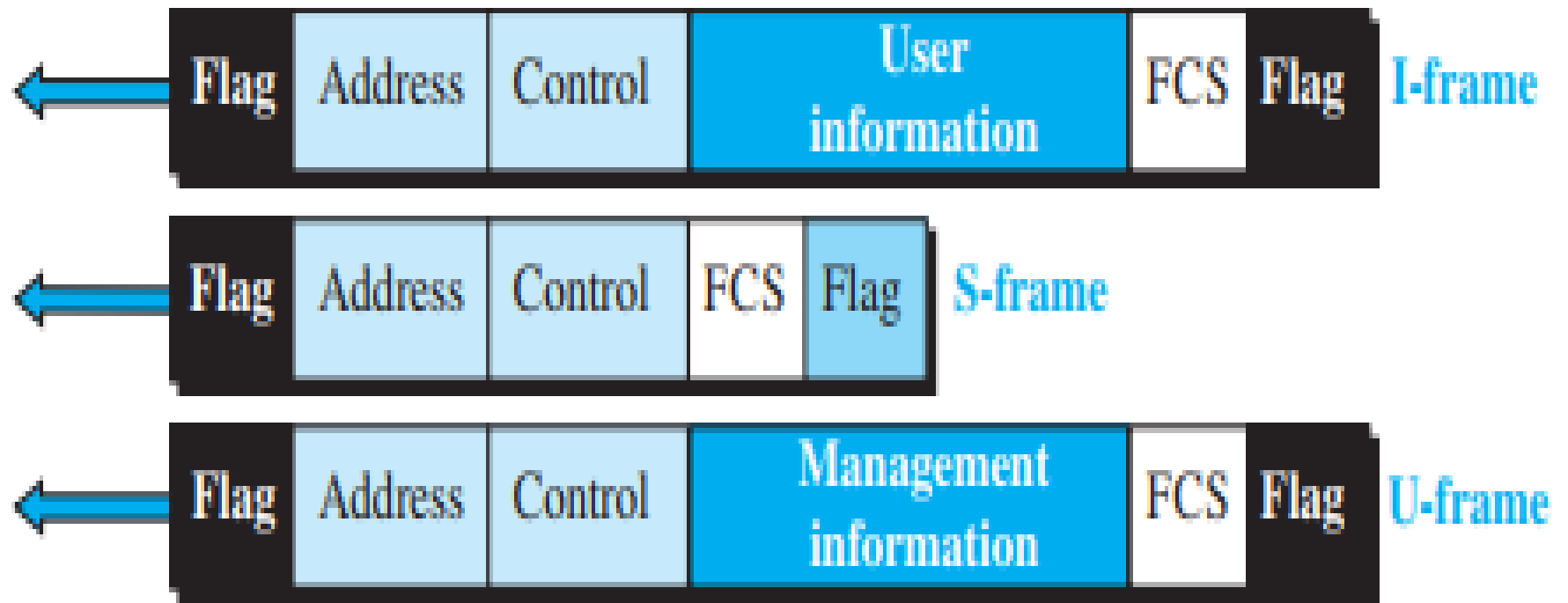
HDLC Framing

HDLC defines three types of frames:

- 1) Information frames (I-frames):** are used to transport user data and control information relating to user data (piggybacking).
 - 2) Supervisory frames (S-frames):** are used only to transport control information.
 - 3) Unnumbered frames (U-frames):** are reserved for system management. Information carried by U-frames is intended for managing the link itself.
- Each type of frame serves as an envelope for the transmission of a different type of message.

HDLC Framing

Figure 11.16 *HDLC frames*



HDLC Framing

- Various fields of HDLC frame are:

1) Flag Field

- This field has a **synchronization pattern 01111110**.
- This field **identifies both the beginning and the end of a frame**.

2) Address Field

- This field **contains the address of the secondary station**.
- If a **primary station created the frame, it contains a to-address**.
- If a **secondary creates the frame, it contains a from-address**.
- This **field can be 1 byte or several bytes long**, depending on the needs of the network.

HDLC Framing

3) Control Field

- **This field is one or two bytes used for flow and error control.**

4) Information Field

- **This field contains the user's data from the network-layer or management information. Its length can vary from one network to another.**

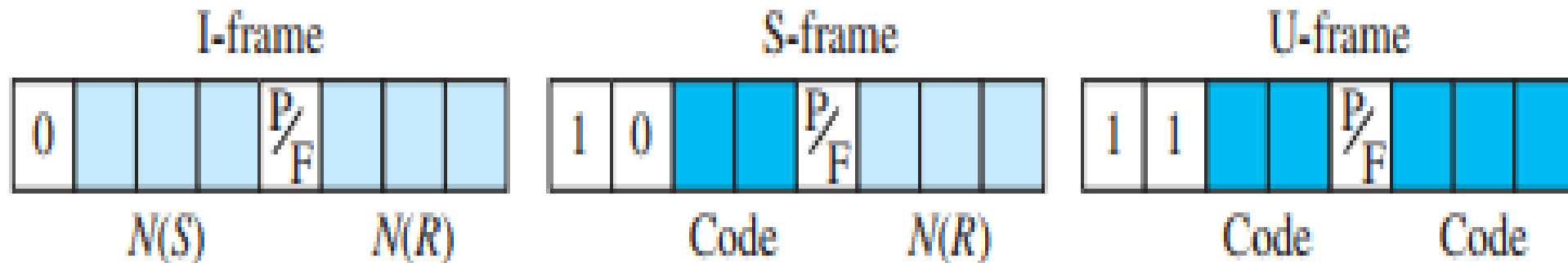
5) FCS Field

- **This field is the error-detection field. (FCS Frame Check Sequence).**
- **This field can contain either a 2- or 4-byte standard CRC**

Control Fields of HDLC Frames

The control field determines the type of frame and defines its functionality.

Figure 11.17 *Control field format for the different frame types*



Control Fields of HDLC Frames

1. I-Frames (Information Frames)

- **Purpose:** Carry user data and provide flow and error control.
- **Control Field Structure:**
 - **1st Bit:** Indicates frame type (0 = I-frame).
 - **N(S):** 3 bits for the sequence number (from 0 to 7).
 - **P/F Bit:** 1 bit indicating whether the frame is a poll (1) or final (1).
 - **N(R):** 3 bits for acknowledgment number when piggybacking is used

Control Fields of HDLC Frames

2. S-Frames (Supervisory Frames)

- **Purpose:** Used for flow control and error control without carrying user data.
- **Control Field Structure:**
 - **1st 2 Bits:** Indicate frame type (10 = S-frame).
 - **N(R):** 3 bits for acknowledgment number (ACK or NAK).
 - **Code:** 2 bits that define the specific type of S-frame:
 - **RR (Receive Ready):** Acknowledges received frames.
 - **RNR (Receive Not Ready):** Acknowledges received frames but indicates the receiver is busy.
 - **REJ (Reject):** Indicates a frame was lost or damaged.
 - **SREJ (Selective Reject):** Indicates a specific lost frame in Selective Repeat ARQ

Control Fields of HDLC Frames

3. U-Frames (Unnumbered Frames)

- **Purpose:** Used for session management and control information exchange.
- **Control Field Structure:**
 - **2-Bit Prefix:** Indicates frame type.
 - **P/F Bit:** Indicates poll or final status.
 - **3-Bit Suffix:** Provides additional information about the specific type of U-frame.
 - **Total Types:** U-frames can define up to 32 different types based on the combination of prefix and suffix bits.

POINT-TO-POINT PROTOCOL (PPP)

- **PPP (Point-to-Point Protocol)** is a common protocol for point-to-point connections.
- Many Internet users use PPP to connect their home computers to an ISP.
- Most use a **modem** with a telephone line.
- The telephone line works as the **physical layer**.
- PPP works at the **data link layer** to control and manage data transfer.
- It is the **most widely used** protocol for this purpose.

POINT-TO-POINT PROTOCOL (PPP)

Services

- PPP is designed with several services suitable for point-to-point communication while omitting some traditional services for simplicity.
- PPP defines the **format of the frame** exchanged between devices.
- It specifies how two devices can **negotiate link establishment** and exchange data.
- PPP can accept payloads from multiple network layer protocols, not just IP.
- Authentication** is supported in PPP, but it is optional.
- Multilink PPP** allows connections over multiple physical links.

POINT-TO-POINT PROTOCOL (PPP)

Services

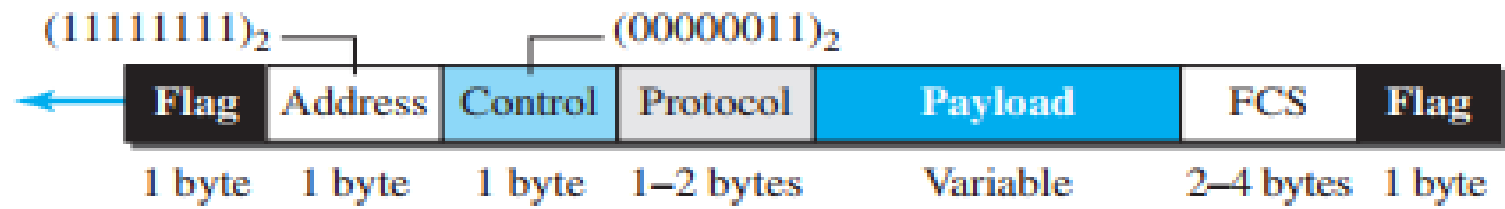
- PPP provides **network address configuration**, which is useful for temporary Internet connections (e.g., for home users).
- PPP does **not** provide **flow control**; the sender can transmit multiple frames without checking if the receiver is overwhelmed.
- Error control in PPP is minimal — it uses a **CRC field** for error detection.
- If a frame is corrupted, it is silently discarded, and the **upper-layer protocol** must handle the problem.
- Lack of error control and sequence numbering can cause packets to arrive out of order.
- PPP does not include a **sophisticated addressing mechanism** for multipoint configurations.

POINT-TO-POINT PROTOCOL (PPP)

Framing

- PPP uses a character-oriented (or byte-oriented) frame.

Figure 11.20 *PPP frame format*



POINT-TO-POINT PROTOCOL (PPP)

Various fields of PPP frame are:

1) **Flag**

- This field has a synchronization pattern 01111110.
- This field identifies both the beginning and the end of a frame.

2) **Address**

- This field is set to the constant value 11111111 (broadcast address).

3) **Control**

- This field is set to the constant value 00000011 (imitating unnumbered frames in HDLC).
- PPP does not provide any flow control.
- Error control is also limited to error detection.

POINT-TO-POINT PROTOCOL (PPP)

4) Protocol

- This field defines what is being carried in the payload field.
- Payload field carries either i) user data or ii) other control information.
By default, size of this field = 2 bytes.

5) Payload field

- This field carries either i) user data or ii) other control information.
- By default, maximum size of this field = 1500 bytes.

POINT-TO-POINT PROTOCOL (PPP)

Byte Stuffing

- Since PPP is a byte-oriented protocol, the flag in PPP is a byte that needs to be escaped whenever it appears in the data section of the frame.
- The escape byte is 0111101, which means that every time the flag like pattern appears in the data, this extra byte is stuffed to tell the receiver that the next byte is not a flag.
- Obviously, the escape byte itself should be stuffed with another escape byte.

POINT-TO-POINT PROTOCOL (PPP)

Transition Phases

- The transition diagram starts with the dead state.

1) Dead State

In dead state, there is no active carrier and the line is quiet.

2) Establish State

When 1 of the 2 nodes starts communication, the connection goes into the establish state. In establish state, options are negotiated between the two parties.

3) Authenticate State

If the 2 parties agree that they need authentication, Then the system needs to do authentication; Otherwise, the parties can simply start communication.

POINT-TO-POINT PROTOCOL (PPP)

Transition Phases

4) Open State

Data transfer takes place in the open state.

5) Terminate State

When 1 of the endpoints wants to terminate connection, the system goes to terminate state.

4) Open State

Data transfer takes place in the open state.

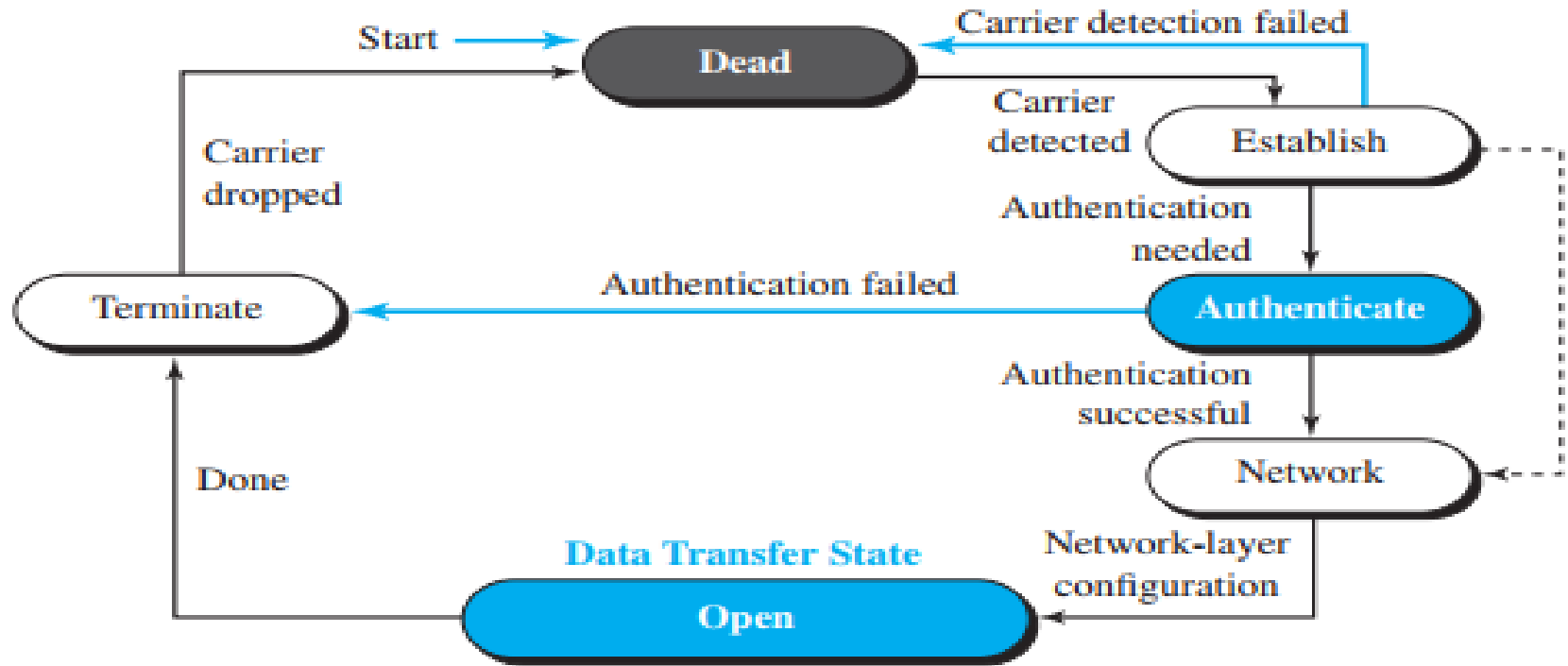
5) Terminate State

When 1 of the endpoints wants to terminate connection, the system goes to terminate state.

POINT-TO-POINT PROTOCOL (PPP)

Transition Phases

Figure 11.21 *Transition phases*



POINT-TO-POINT PROTOCOL (PPP)

Multiplexing

- **PPP** is a link-layer protocol but uses other protocols to establish the link, authenticate, and carry network-layer data.
- Three main sets of protocols make PPP powerful:
 - **Link Control Protocol (LCP)**
 - **Authentication Protocols (APs)**
 - **Network Control Protocols (NCPs)**
- At any time, a PPP packet can carry data from one of these protocols in its data field.
- Data may also come from different network layers.

POINT-TO-POINT PROTOCOL (PPP)

Multiplexing

Legend

LCP: Link control protocol
AP: Authentication protocol
NCP: Network control protocol

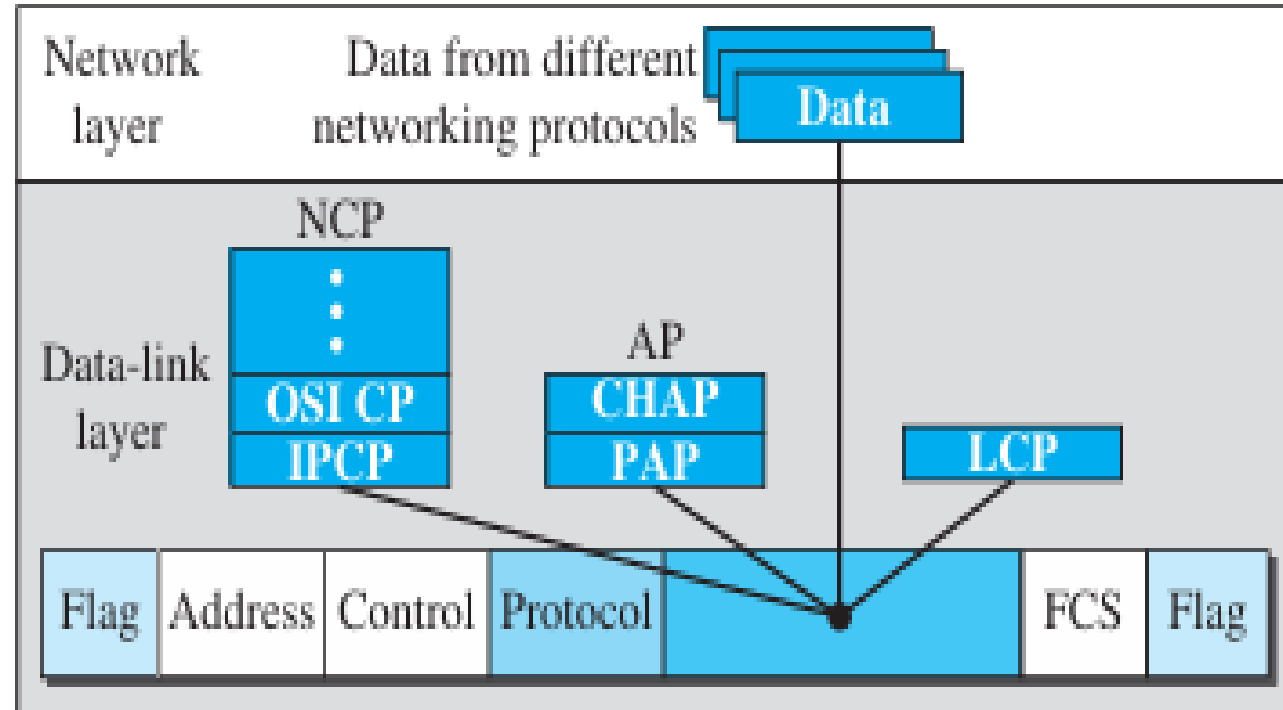
Protocol values:

LCP: 0xC021

AP: 0xC023 and 0xC223

NCP: 0x8021 and

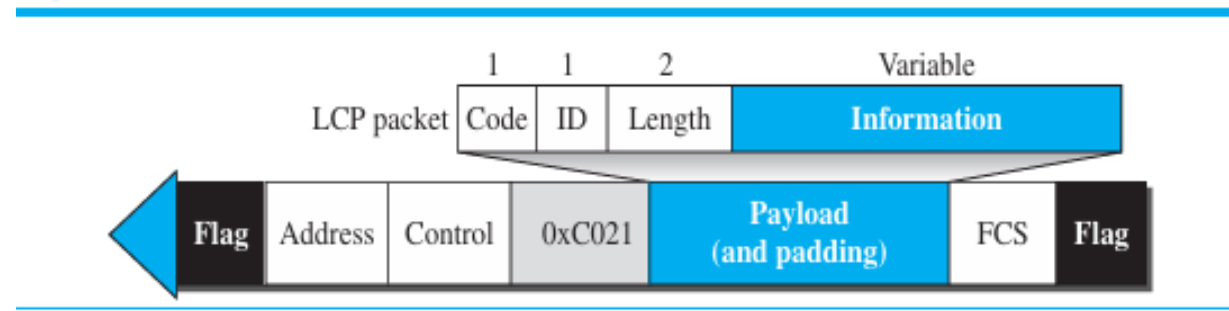
Data: 0x0021 and



POINT-TO-POINT PROTOCOL (PPP)

Multiplexing

Figure 11.23 LCP packet encapsulated in a frame



1. LCP (Link Control Protocol)

- Used to establish, configure, maintain, and terminate the link.
- Negotiates options (like authentication method, compression).
- Protocol field value: 0xC021 in PPP frame.
- Has packet types:
 1. Link setup: Configure-request, Configure-ack, Configure-nak, Configure-reject.
 2. Link termination: Terminate-request, Terminate-ack.
 3. Monitoring/debugging: Code-reject, Protocol-reject, Echo-request, Echo-reply, Discard-request.

POINT-TO-POINT PROTOCOL (PPP)

Multiplexing

2. Authentication Protocols (APs)

Used to verify identity during the authentication phase.

Two types:

1. PAP (Password Authentication Protocol)

1. Simple, sends username & password in plain text.
2. 3 packet types: Authenticate-request, Authenticate-ack, Authenticate-nak.
3. Protocol field: 0xC023.

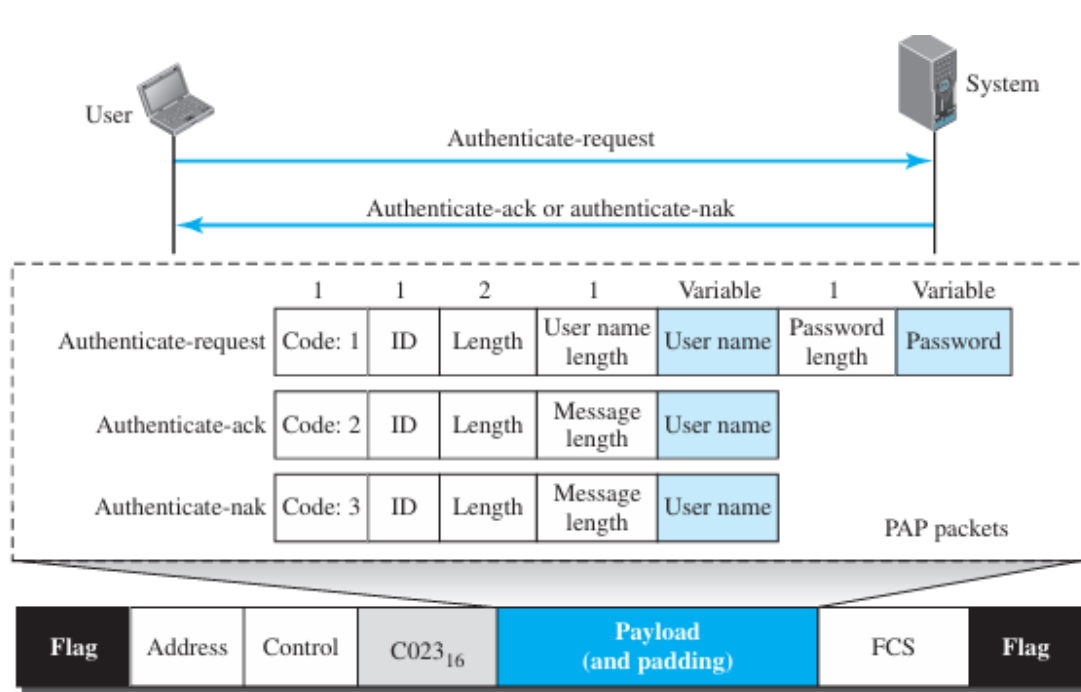
2. CHAP (Challenge Handshake Authentication Protocol)

1. More secure, uses challenge-response method.
2. Password is never sent directly.
3. 4 packet types: Challenge, Response, Success, Failure.
4. Protocol field: 0xC223.

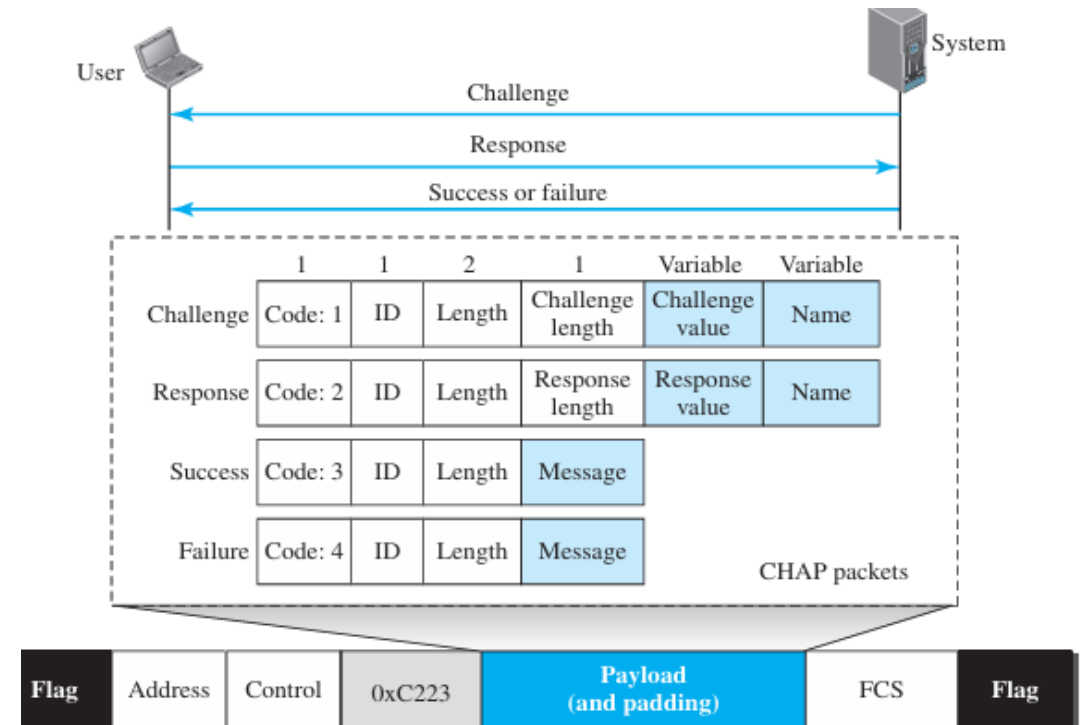
POINT-TO-POINT PROTOCOL (PPP)

Multiplexing

2. Authentication Protocols (APs)



PAP (Password Authentication Protocol)



CHAP (Challenge Handshake Authentication Protocol)

POINT-TO-POINT PROTOCOL (PPP)

Multiplexing

3. NCP (Network Control Protocols)

- PPP uses NCPs to support multiple network layer protocols at the same time.
- Each network protocol (like IP, IPv6, AppleTalk) has its own NCP. Example: IPCP for IP, IPXCP for IPX, ATCP for AppleTalk.
- After authentication, NCPs configure network settings (like assigning IP addresses, setting DNS servers, etc.) between the two connected devices.
- NCPs negotiate options needed for each network protocol, so both sides agree on how data will be transferred for that protocol.
- Only after NCP negotiation is complete can the actual user data from that network layer protocol be sent.

POINT-TO-POINT PROTOCOL (PPP)

Multiplexing

3. NCP (Network Control Protocols)

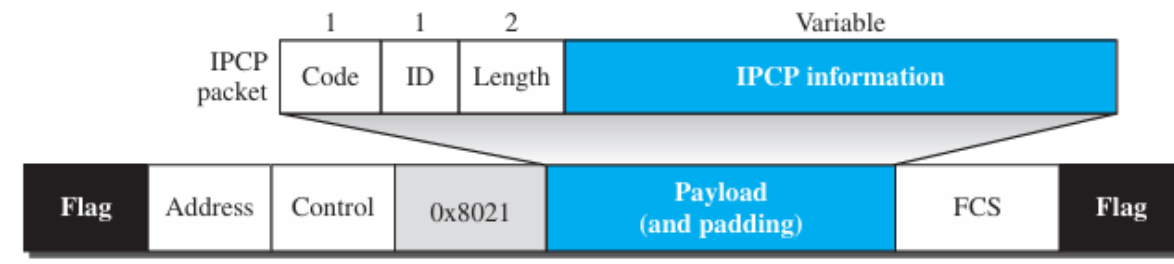
IPCP

IPCP (Internet Protocol Control Protocol) is a PPP protocol for setting up IP over a PPP link.

1. It runs after authentication to assign IP addresses and agree on IP settings.

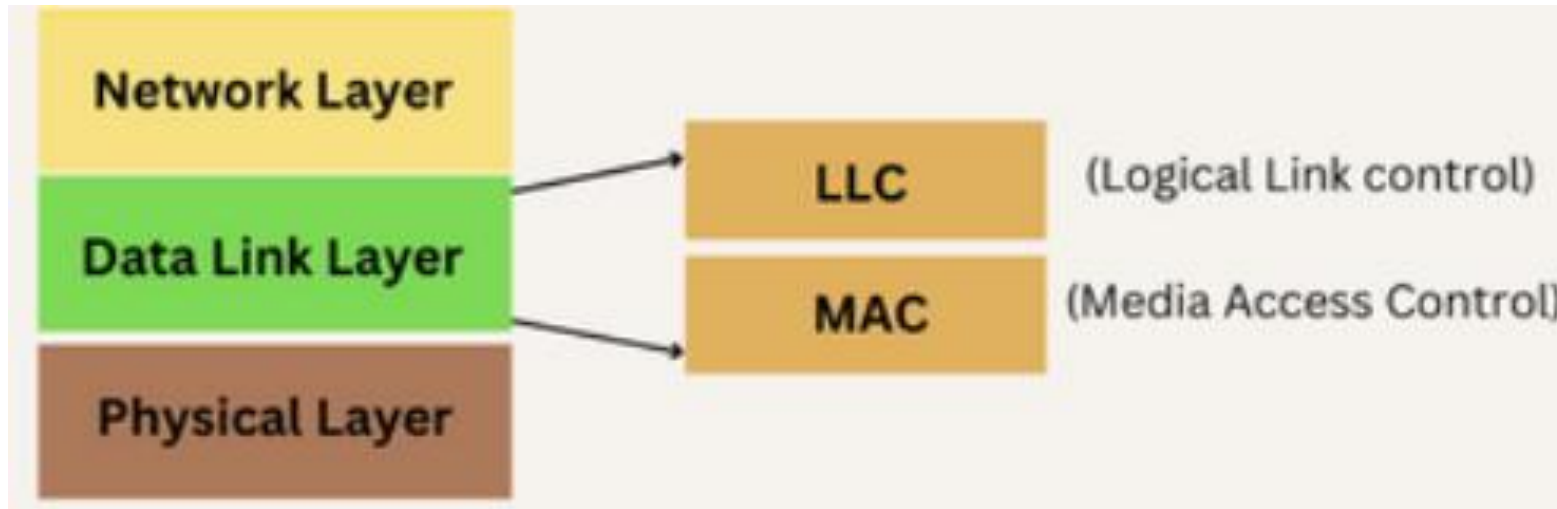
2. Protocol field value in PPP frame = 0x8021.

3. Only after IPCP setup can IP packets be sent over the link.



Chapter 3 - Media Access Control

Media Access Control



MAC: Controls how devices access a network medium and transmit data.

LLC: Identifies and encapsulates network layer protocols. It also controls error checking and frame synchronization.

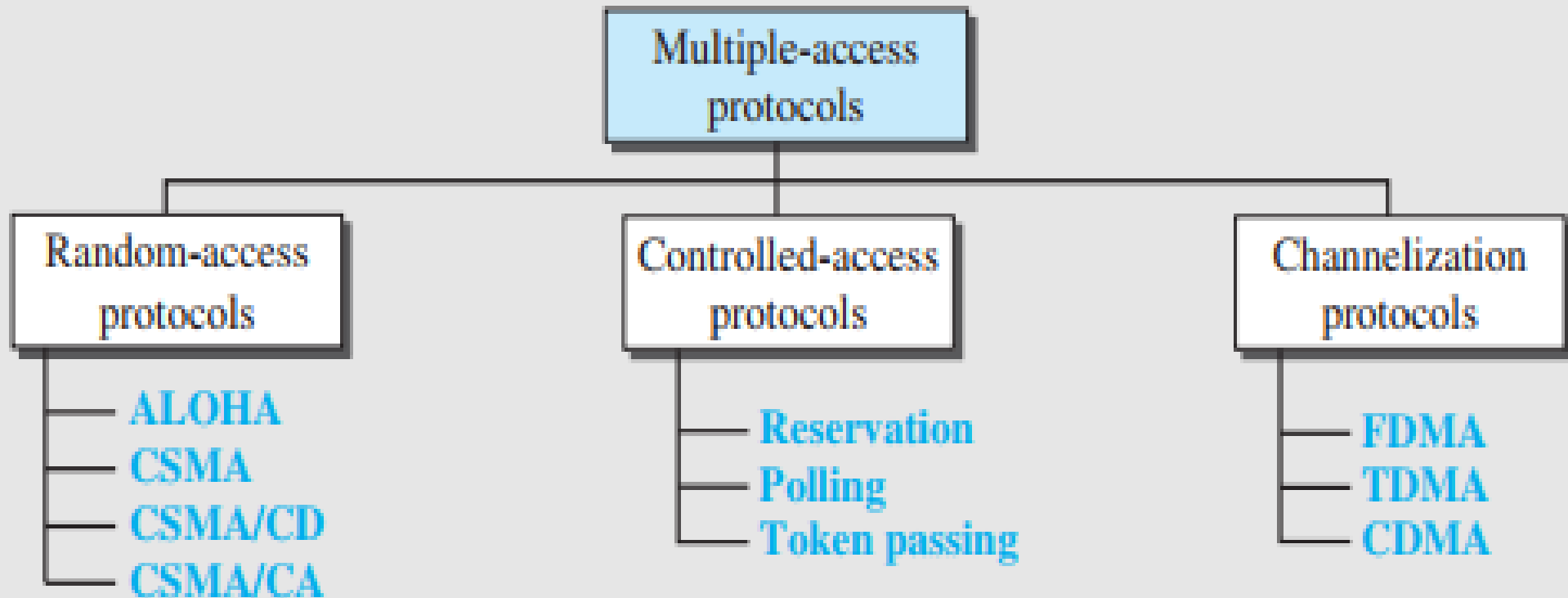
Media Access Control

- When **multiple nodes share a common link** (called a multipoint or broadcast link), they need a **multiple-access protocol to manage communication.**
- These protocols, which are part of the **Media Access Control (MAC) sublayer in the data link layer, help coordinate access to the shared link and fall into three main categories.**

Media Access Control

- Many protocols have been designed to handle access to a shared-link

Figure 12.1 *Taxonomy of multiple-access protocols*



Random-access

- In random-access methods, **no station has control over the others**, and **all stations are equal**.
- **When a station wants to send data**, it **checks whether the medium is free (idle) or busy**. If the **medium is free**, the station can transmit its data following specific rules.
- These methods are called **random access** because **there is no set schedule for when each station can send data**.
- However, **if multiple stations try to send data at the same time**, a **collision occurs**, leading to **lost or altered frames**.

ALOHA

- **A multiple access protocol that allows data to be transmitted over a public network channel.**
- **Since the medium is shared, collisions can occur when multiple stations try to send data simultaneously, resulting in garbled messages.**
- **After sending, the user waits for an acknowledgment. If an acknowledgment is not received (due to a collision with another user's transmission), the user resends the data after a random time.**
- **Different Versions of ALOHA**
 - 1. Pure Aloha**
 - 2. Slotted Aloha**

ALOHA

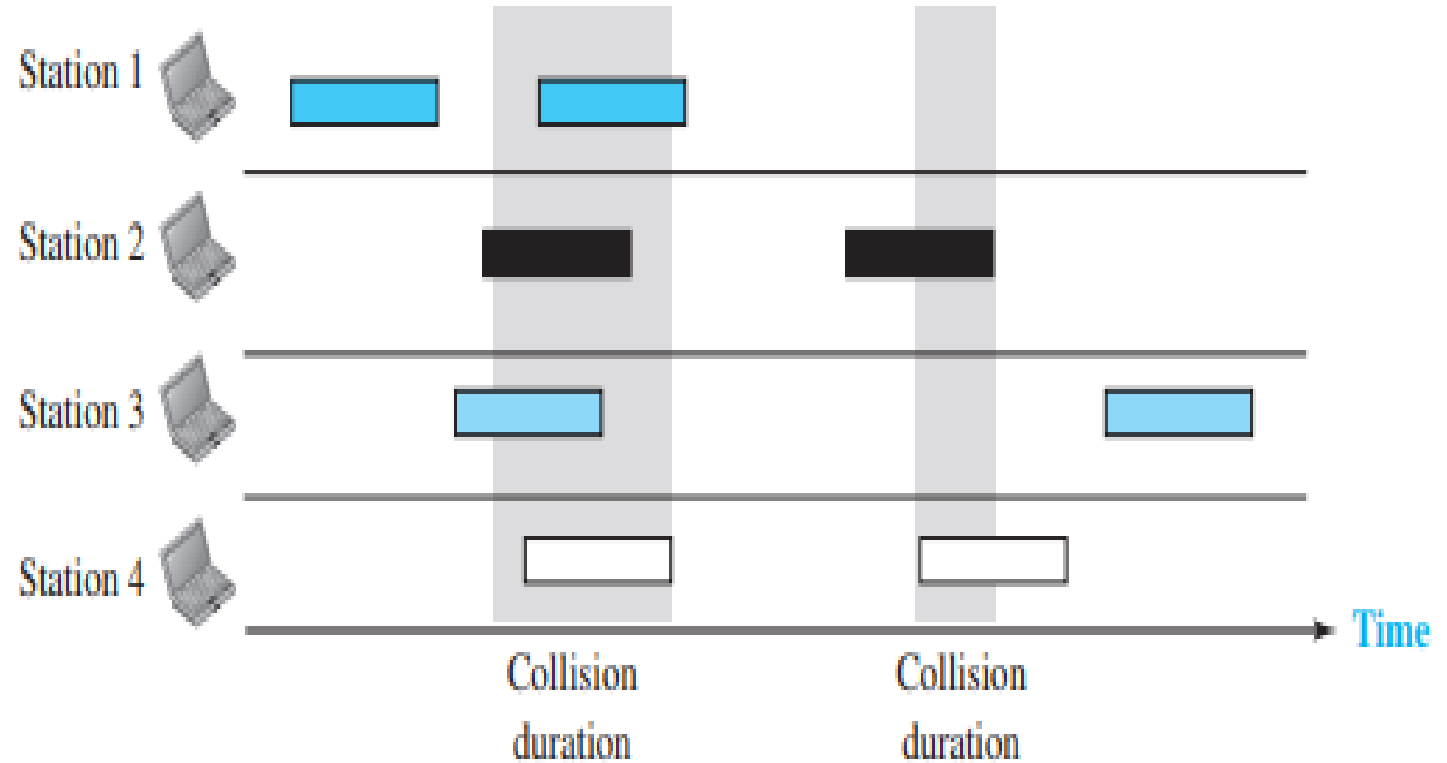
Pure ALOHA

- **The original version of ALOHA is known as pure ALOHA.** In this simple protocol, **each station sends data whenever it has something to send.**
- **However, with only one channel, there's a chance that frames from different stations will collide** (as shown in Figure 12.2), **leading to data loss.**

ALOHA

Pure ALOHA

Figure 12.2 *Frames in a pure ALOHA network*



In this example, **four stations are trying to access the same shared channel**, which is a simplified scenario.

ALOHA

- **Each station sends two frames**, resulting in a **total of eight frames** on the channel. However, **some frames collide because multiple stations are trying to send their data simultaneously.**
- As shown in Figure; **only two frames successfully transmit: one from station 1 and one from station 3.**

It's important to note that **if even a single bit of one frame overlaps with a bit from another frame on the channel, a collision occurs, and both frames are destroyed.**
- This means that **any frames lost during transmission must be resent.**

ALOHA

Acknowledgment in Pure ALOHA

- The **pure ALOHA protocol depends on acknowledgments** (ACKs) from the receiver.
- When a **station sends a frame, it expects to receive an acknowledgment back.**
- If the **acknowledgment doesn't arrive within a set time** (time-out period), the **station assumes the frame (or its acknowledgment) was lost and resends the frame.**

ALOHA

Handling Collisions

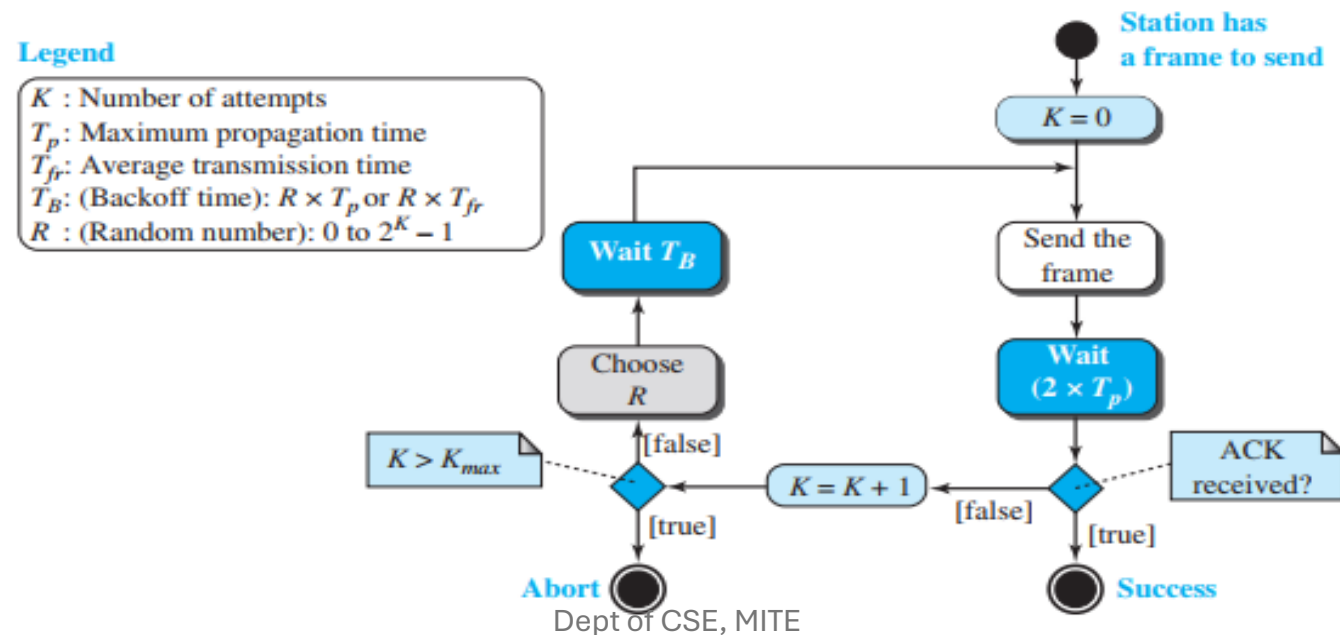
- When **multiple stations send frames simultaneously, a collision occurs.**
- If these stations attempt to resend their frames after the time-out, they **may collide again.**
- To reduce the chance of repeated collisions, **each station waits a random amount of time** (called **backoff time, T_B**) before resending its frame.

ALOHA

Maximum Retransmission Attempts

- Pure ALOHA has a **second method to prevent channel congestion:**
- **After reaching a maximum number of retransmission attempts (K_{max}), a station must stop trying to resend and wait before attempting again.**

Figure 12.3 Procedure for pure ALOHA protocol



K_{max} is left to the system designer and can be influenced by factors like network load, desired performance.

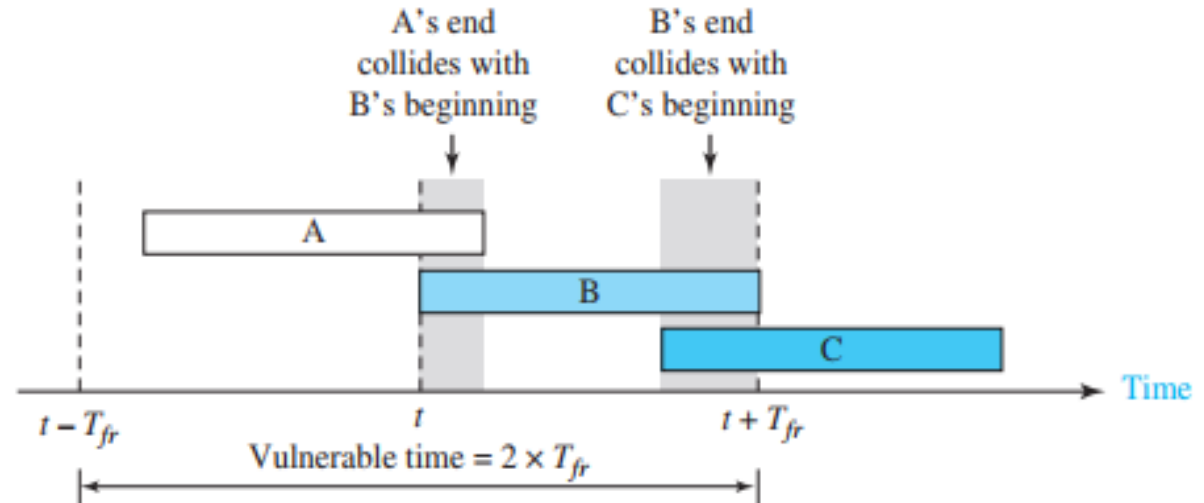
ALOHA

Vulnerable Time

- Vulnerable time is defined as the **time interval during which a frame sent by one station can collide with frames sent by other stations**. In Pure ALOHA, this vulnerable time is equal to twice the frame transmission time (T_{fr}), expressed mathematically as:
 - **$V_t = 2 \times T_{fr}$**
- The vulnerable time tells us the period **during which a frame is at risk of collision**. It is to Understand the Collision Probability

ALOHA

Figure 12.4 Vulnerable time for pure ALOHA protocol



- **t**: The time when a station (e.g., station A) begins sending its frame.
- **t - T_{fr}**: The **moment just before station A starts transmitting**. If another station (e.g., station B) sends its frame during this time, it can collide with A's transmission.
- **t + T_{fr}**: The **time immediately after station A starts sending**. If another station (e.g., station C) transmits during this period, it may also collide with A's frame.

ALOHA

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 T_f is 200 bits/200 kbps or 1 ms. The vulnerable time is $2 \times 1 \text{ ms} = 2 \text{ ms}$. 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.

ALOHA

Throughput (actual rate at which data is successfully transmitted over a communication channel)

- The average number of successful transmissions is given by

$$S = G \times e^{-2G}.$$

- where **G = average no. of frames in one frame transmission time (T_{fr})**
- For **G = 1/2**, (half of a frame time is being used for transmission attempts)

$$S_{\max} = 0.184.$$

- In other words, out of 100 frames, 18 frames reach their destination successfully.

ALOHA

Throughput

1. Setting the Value of G :

- The maximum throughput occurs when $G = \frac{1}{2}$.

2. Substituting into the Throughput Formula:

- Substitute $G = \frac{1}{2}$ into the formula:

$$S_{max} = \frac{1}{2} \times e^{-2 \times \frac{1}{2}} = \frac{1}{2} \times e^{-1}$$

3. Calculating e^{-1} :

- The value of e^{-1} is approximately 0.3679.

4. Final Calculation:

$$S_{max} = \frac{1}{2} \times 0.3679 \approx 0.18395$$

ALOHA

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.

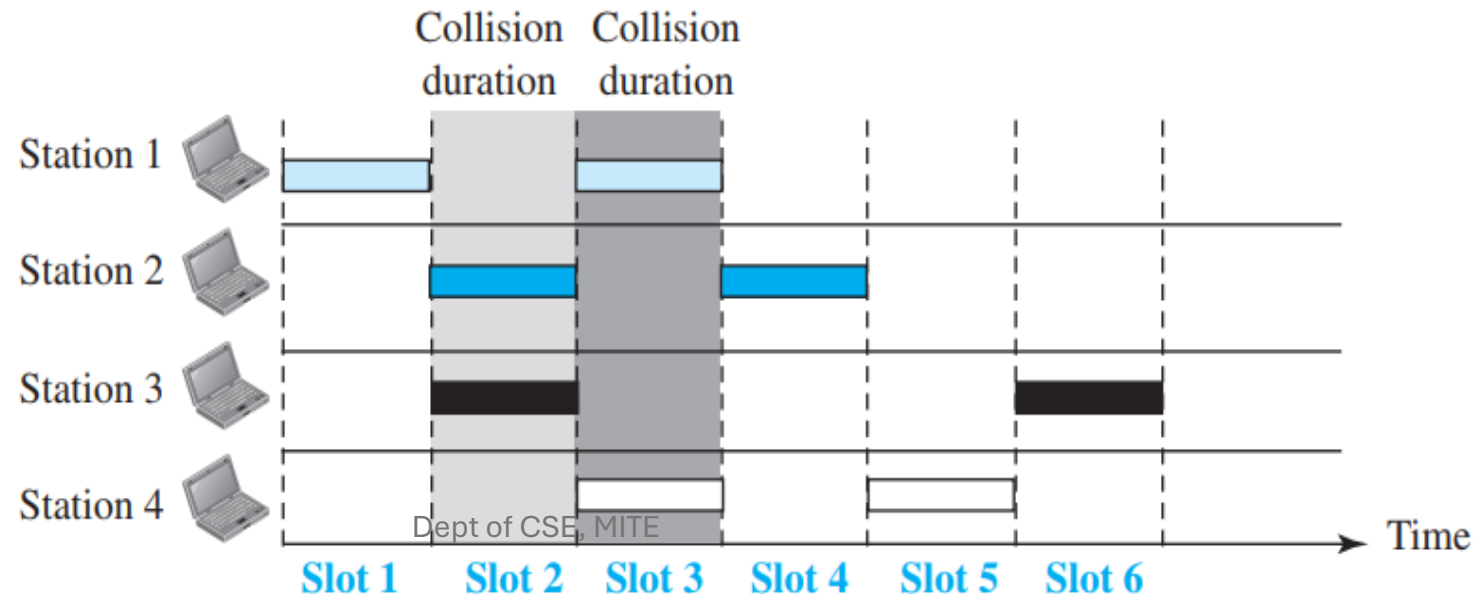
- a. If the system creates 1000 frames per second, or 1 frame per millisecond, then $G = 1$. In this case $S = G \times e^{-2G} = 0.135$ (13.5 percent). This means that the throughput is $1000 \times 0.135 = 135$ frames. Only 135 frames out of 1000 will probably survive.
- b. If the system creates 500 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.

Slotted ALOHA

Slotted ALOHA was invented to improve the efficiency of pure ALOHA.

- The time is divided into equal, discrete slots, and each slot is long enough to send one frame of data, T_{fr} seconds.
- The stations are allowed to send only at the beginning of the time-slot.

Figure 12.5 *Frames in a slotted ALOHA network*



Slotted ALOHA

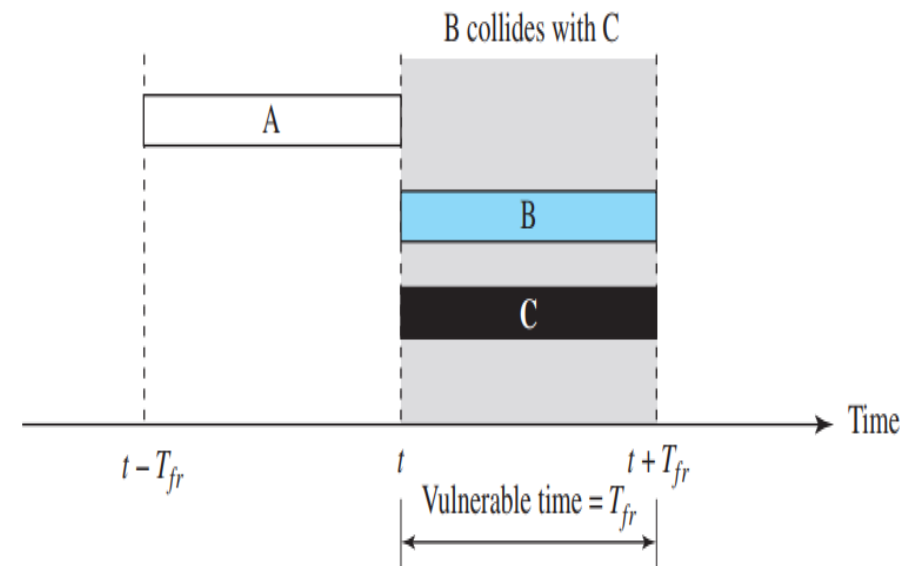
If a station misses the time-slot, the station must wait until the beginning of the next time-slot.

If 2 stations try to resend at beginning of the same time-slot, the frames will collide again

The vulnerable time is given by: T_{fr}

In Slotted Aloha, the vulnerable time is reduced to one frame time because transmissions are restricted to the start of each time slot, reducing the chances of overlapping frames.

Figure 12.6 Vulnerable time for slotted ALOHA protocol



Slotted ALOHA

Throughput

- The average number of successful transmissions is given by

$$S = G \times e^{-G}$$

- For $G = 1$, the maximum throughput $S_{\max} = 0.368$.
- In other words, out of 100 frames, 36 frames reach their destination successfully.

Carrier sense multiple access (CSMA)

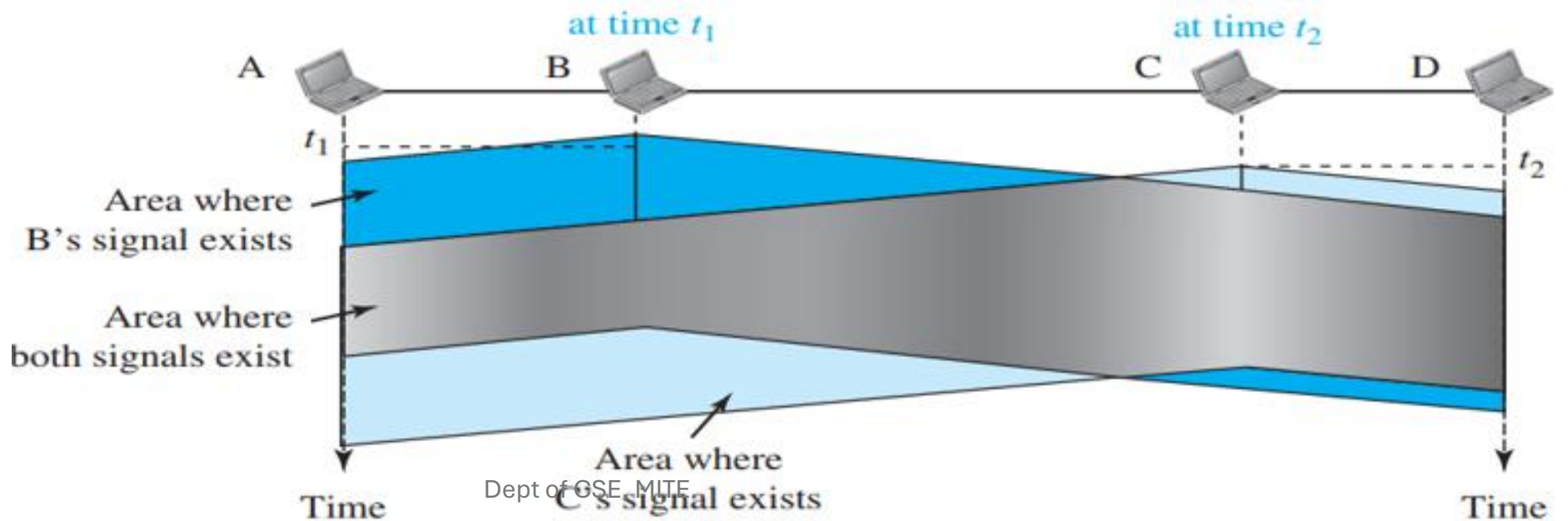
- CSMA was developed to **minimize the chance of collision** and, therefore, **increase the performance.**
- CSMA is based on the principle “**sense before transmit**” or “**listen before talk.**”
- Here is how it works:
 - 1) Each **station checks the state of the medium: idle or busy.**
 - i) If the **medium is idle**, the **station sends the data.**
 - ii) If the **medium is busy**, the **station defers sending.**

Carrier sense multiple access (CSMA)

- CSMA can **reduce the possibility of collision, but it cannot eliminate it.**
The possibility of collision still exists.
- For example:
- When a **station sends a frame, it still takes time for the first bit to reach every station and for every station to sense it.**

Carrier sense multiple access (CSMA)

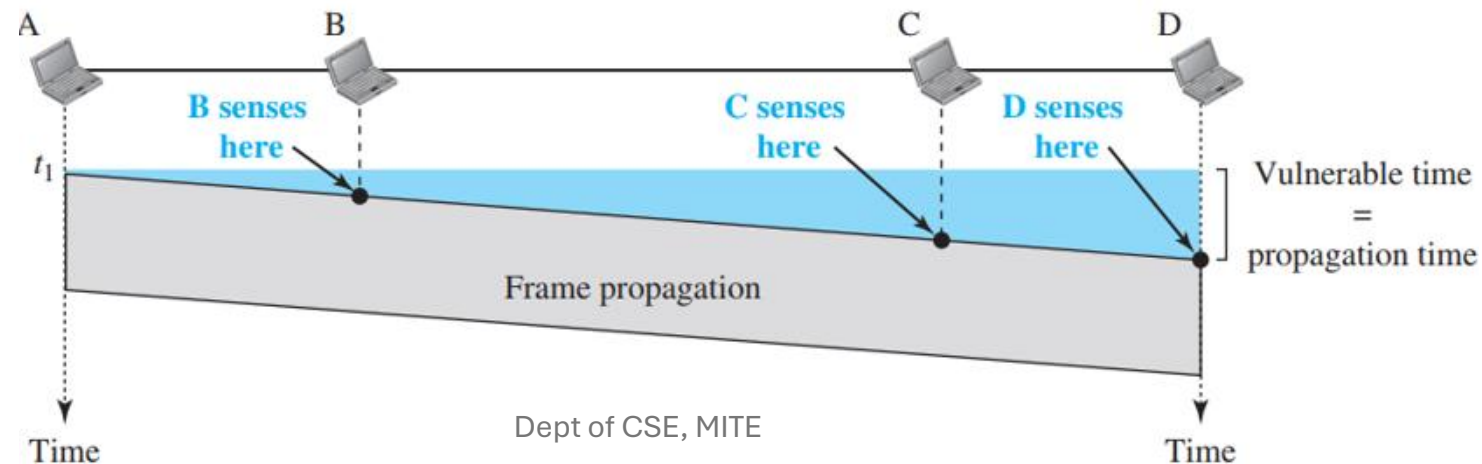
- **B starts transmitting at time t_1** , and its signal propagates over time.
- **C starts transmitting at time t_2** , but by then, **C's device is unaware that B has already started transmitting**, leading to a **collision** when the signals overlap.
- The **gray area** in the figure represents the **region where both signals from B and C exist at the same time, indicating the collision.**



Carrier sense multiple access (CSMA)

Vulnerable Time

- The **vulnerable time for CSMA is the propagation time T_p** . This is the time needed for a signal to propagate from one end of the medium to the other.
- The **maximum time it takes for all stations to be aware of a transmission in progress**.



Carrier sense multiple access (CSMA)

Vulnerable Time

- Collision occurs when

→ **A station sends a frame, and other station also sends a frame during propagation time.**

- **If the first bit of the frame reaches the end of the medium, every station will refrain(stop) from sending to avoid collision.**

Carrier sense multiple access (CSMA)

Persistence Methods

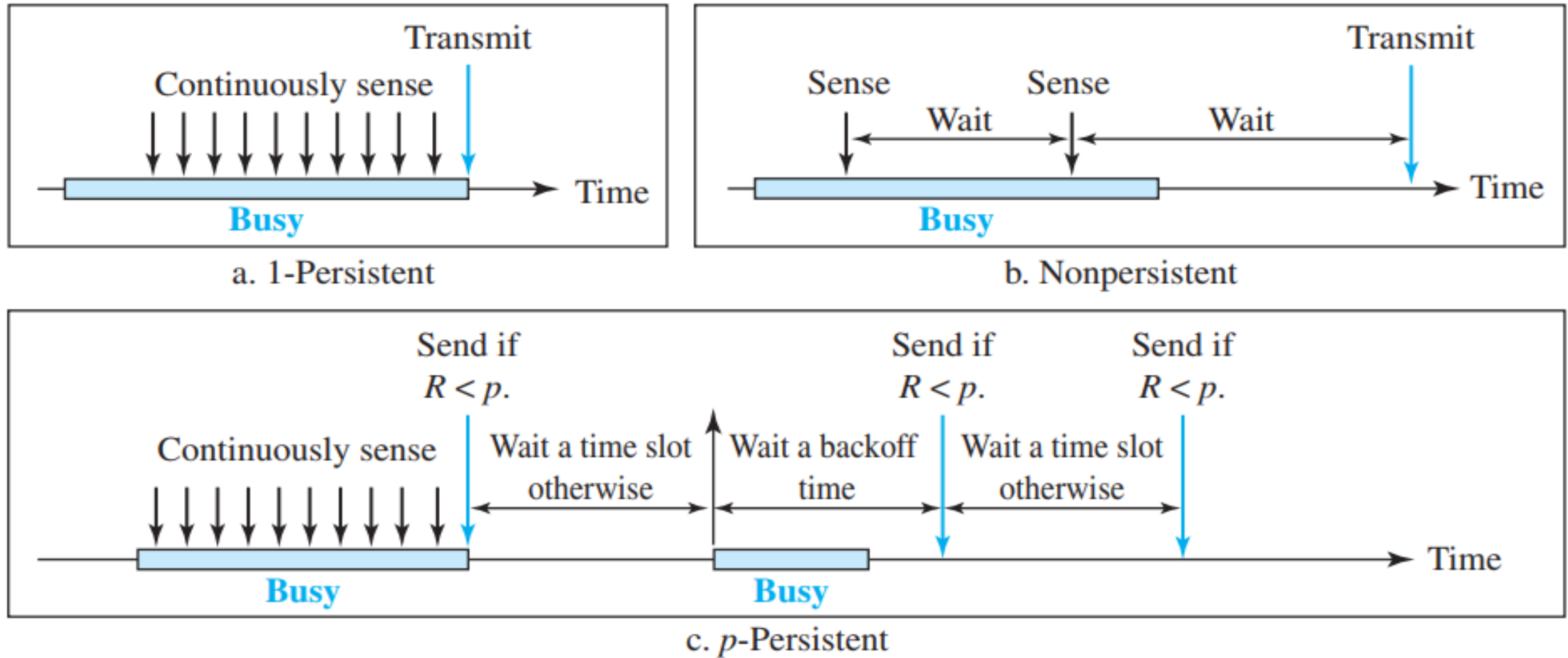
Helps to **Manage** how devices share a communication channel and handle **potential collisions**

- Q: **What should a station do if the channel is busy or idle?**

Three methods can be used to answer this question:

- 1) 1-persistent method
- 2) Non-persistent method and
- 3) p-persistent method

Figure 12.9 Behavior of three persistence methods



Carrier sense multiple access (CSMA)

1) 1-persistent method

- **Before sending a frame, a station senses the line.**
 - i) If the *line is idle, the station sends immediately* (with probability = 1).
 - ii) If the **line is busy, the station continues sensing the line.**
- This method **has the highest chance of collision** because

2 or more stations:

→ may find the line idle and

→ send the frames immediately.

Carrier sense multiple access (CSMA)

2) Non-Persistent

- **Before sending a frame, a station senses the line** (Figure 12.10b).

i) If the *line is idle, the station sends immediately.*

ii) If the *line is busy, the station waits a random amount of time and then senses the line again.*

- This method **reduces the chance of collision**

because 2 or more stations:

→ will not wait for the same amount of time and

→ will not retry to send simultaneously.

Carrier sense multiple access (CSMA)

3) P-Persistent

- This method is used if the channel has time-slots with a slot-duration equal to or greater than the maximum propagation time (Figure 12.10c).
- Advantages:
 - i) It combines the advantages of the other 2 methods.
 - ii) It reduces the chance of collision and improves efficiency.

Carrier sense multiple access (CSMA)

3) P-Persistent

- **After the station finds the line idle**, it follows these steps:

- 1) With *probability p* , *the station sends the frame(time slot 1)*.

- 2) With *probability $q=1-p$* , *the station waits for the beginning of the next time-slot and checks the line again(next slot 2)*

- i) **If line is idle**, it goes to step 1.

- ii) **If line is busy**, it assumes that **collision has occurred and uses the back off procedure.**

CSMA/CD (collision detection)

Disadvantage of CSMA: CSMA does not specify the procedure after a collision has occurred.

Solution: CSMA/CD enhances the CSMA to handle the collision.

- Here is how it works (Figure 12.12):

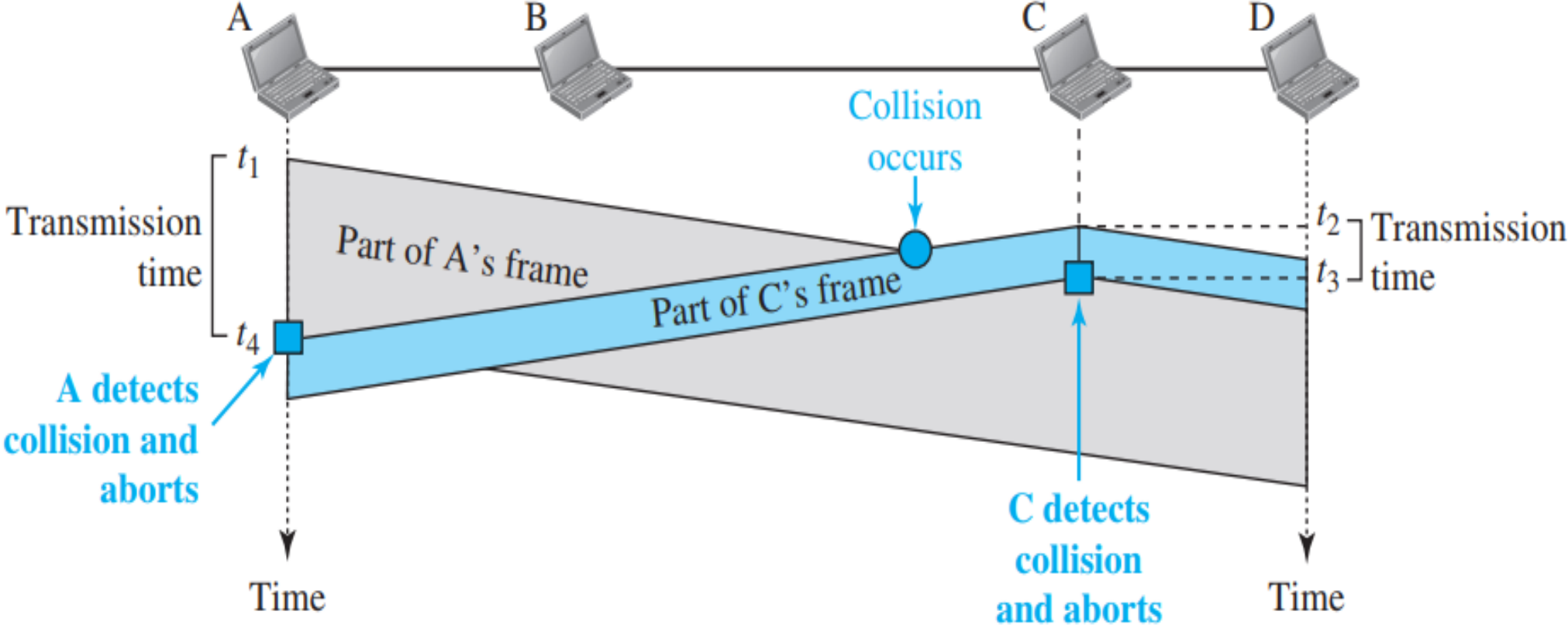
1) **A station**

→ *sends the frame & then monitors the medium to see if the transmission was successful or not.*

2) *If the transmission was unsuccessful (i.e. there is a collision), the frame is sent again.*

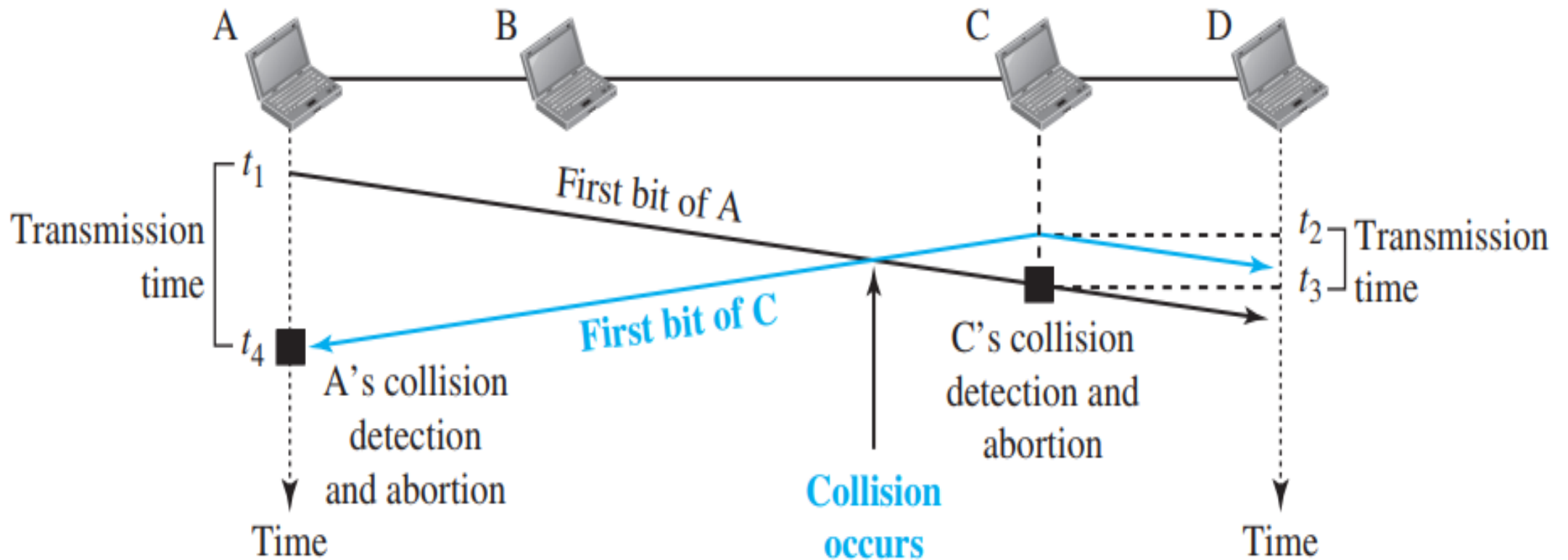
CSMA/CD (collision detection)

Figure 12.12 Collision and abortion in CSMA/CD



CSMA/CD (collision detection)

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



CSMA/CD (collision detection)

- At **time t1**, **station A** has executed its procedure and **starts sending** the bits of its frame. At **time t2**, **station C** has executed its procedure and **starts sending the bits** of its frame. The **collision occurs sometime after time t2**.
- **Station C detects a collision at time t3** when it receives the first bit of A's frame.
- **Station C immediately aborts transmission.**
- **Station A detects collision at time t4** when it receives the first bit of C's frame.

CSMA/CD (collision detection)

- **Station A** also immediately aborts transmission.
- **Station A** transmits for the duration t_4-t_1 . **Station C** transmits for the duration t_3-t_2 .
- **For the protocol to work:**
- The **length of any frame divided** by the **bit rate** must be more than either of these durations.

CSMA/CD (collision detection)

Minimum Frame Size

- 1. Frame Size Restriction:** The size of the *frame must be long enough to allow the sender to detect collisions while transmitting.*
- 2. Collision Detection:** The *sender needs to be able to notice if a collision occurs before it finishes sending the last bit of the frame.*
- 3. No Copy of Frame:** The *sender doesn't keep a copy of the frame and doesn't monitor the line continuously for collisions.*
- 4. Transmission Time:** The *time to send the entire frame (T_{fr}) must be at least twice the maximum time it takes for a signal to travel through the network ($2T_p$).* This ensures that if a collision happens, the sender can detect it before finishing the transmission.

CSMA/CD (collision detection)

Procedure

- CSMA/CD is similar to ALOHA with 2 differences (Figure 12.13):

1) **Addition of the persistence process.**

We need to **sense the channel before sending the frame** by using non-persistent, 1-persistent or p-persistent.

2) **Frame transmission.**

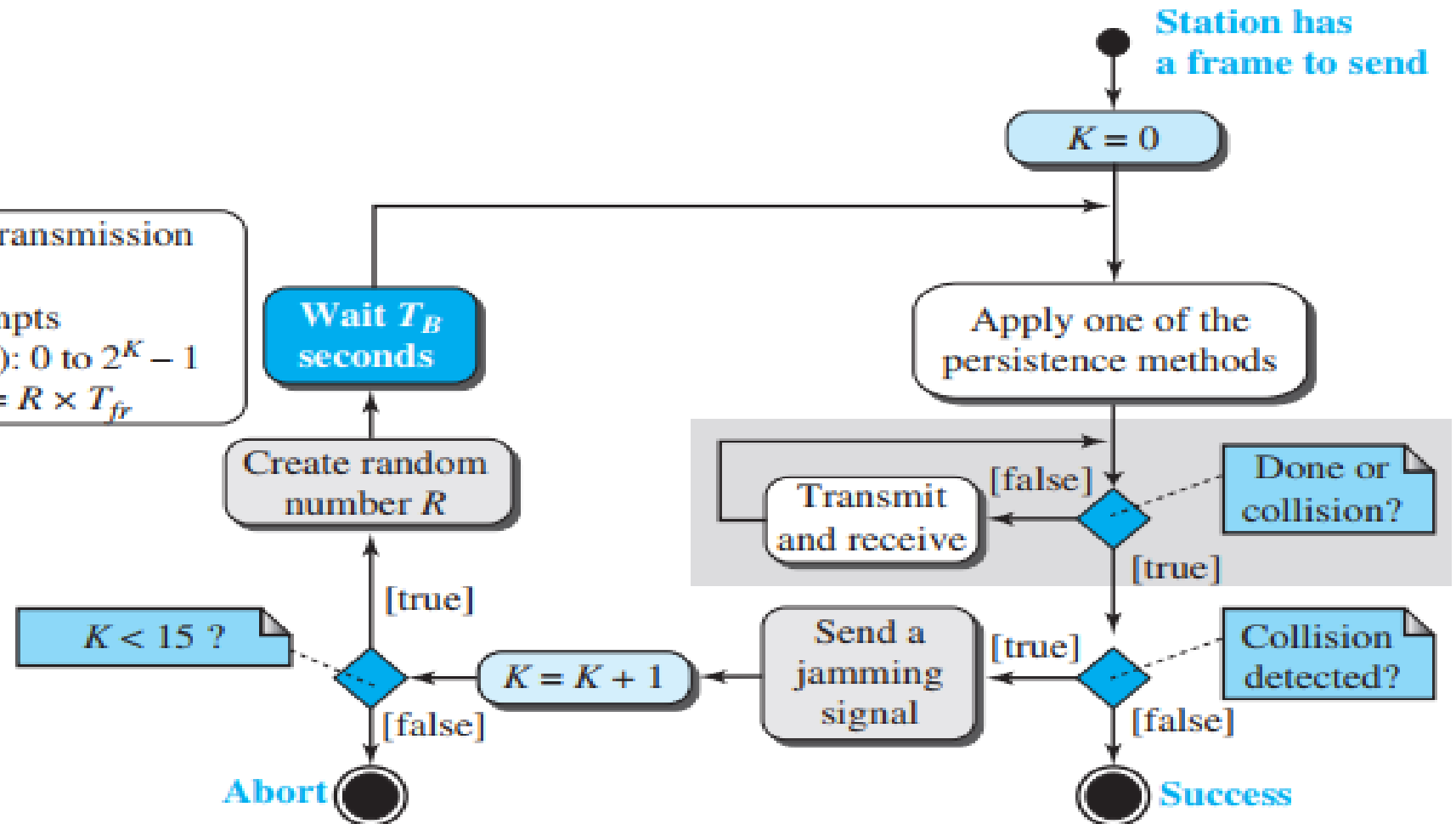
- i) In ALOHA, **first the entire frame is transmitted and then acknowledgment is waited for.**
- ii) In CSMA/CD, **transmission and collision-detection is a continuous process.**

CSMA/CD (collision detection)

Figure 12.13 Flow diagram for the CSMA/CD

Legend

T_{fr} : Frame average transmission time
 K : Number of attempts
 R : (random number): 0 to $2^K - 1$
 T_B : (Backoff time) = $R \times T_{fr}$



CSMA/CD (collision detection)

Energy Level of Channel

The energy-level can have 3 values: 1) Zero 2) Normal and 3) Abnormal.

1) At *zero level*, the channel is idle

2) At *normal level*, a station has successfully captured the channel and is sending its frame.

3) At *abnormal level*, there is a collision and the level of the energy is twice the normal level.

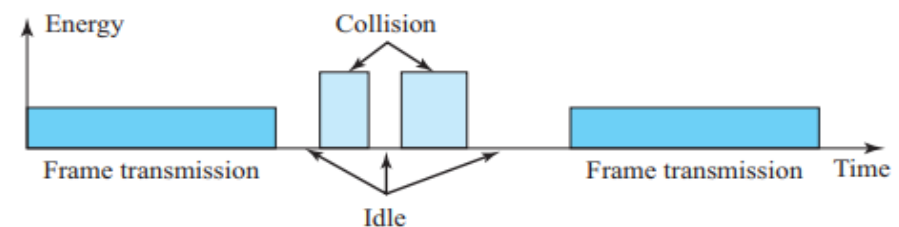
A sender needs to monitor the energy-level to determine if the channel is

→ Idle

→ Busy or

→ Collision mode

Figure 12.14 Energy level during transmission, idleness, or collision



CSMA/CD (collision detection)

Throughput

- The **throughput of CSMA/CD is greater than pure or slotted ALOHA.**

- The **maximum throughput is based on different value of G**

→ persistence method used (non-persistent, 1-persistent, or p-persistent) and → ‘p’ value in the p-persistent method.

- For 1-persistent method, the maximum throughput is 50% when $G = 1$.

- For non-persistent method, the maximum throughput is 90% when G is between 3 and 8.

CSMA/CD (collision detection)

Key Differences from CSMA/CD

1.No Collision Detection: CSMA does not actively detect collisions during transmission; devices may only realize a collision has occurred when a packet is lost or an acknowledgment isn't received.

2.Simpler Handling: CSMA lacks mechanisms like jam signals and structured backoff algorithms, making collision management less sophisticated and potentially leading to inefficiencies in busy networks.

CSMA/CA (collision avoidance)

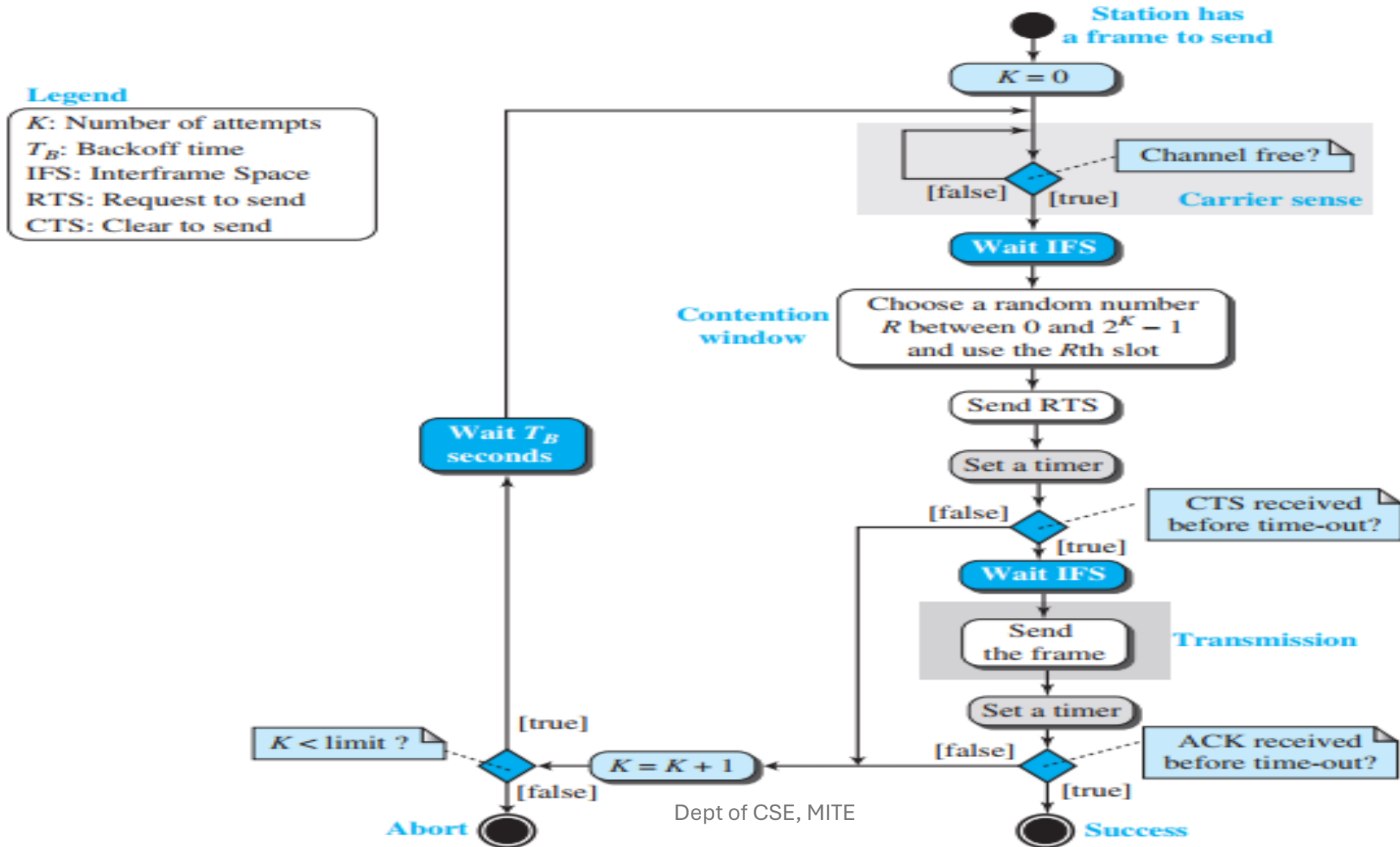
Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for wireless networks.

Here is how it works (Figure 12.15):

- 1) **A station needs to be able to receive while transmitting to detect a collision.**
 - i) When *there is no collision*, the *station receives one signal*: its own signal.
 - ii) When *there is a collision*, the *station receives 2 signals*:
 - a) **Its own signal** and
 - b) **Signal transmitted by a second station.**
- 2) To distinguish b/w these 2 cases, **the received signals in these 2 cases must be different.**

CSMA/CA (collision avoidance)

Figure 12.15 Flow diagram of CSMA/CA



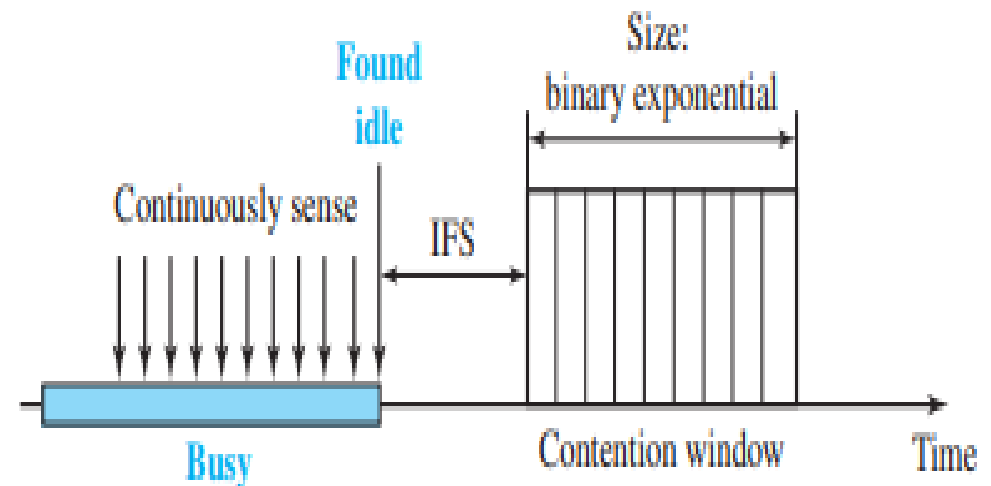
Feature	CSMA	CSMA/CD	CSMA/CA
Collision Handling	No specific action to detect or prevent collisions.	Detects collisions and retransmits after a random backoff time.	Avoids collisions using mechanisms like RTS/CTS and acknowledgment.
Transmission Medium	Can be used in any shared communication medium.	Primarily used in wired Ethernet networks.	Primarily used in wireless networks (e.g., Wi-Fi).
Collision Management	None (collisions may occur).	Collisions detected during transmission and handled after detection.	Collisions are avoided by coordinating transmissions.
Efficiency	Lower efficiency due to potential collisions.	Improved efficiency but still prone to delays from collisions.	Higher efficiency in avoiding collisions, especially in wireless environments.

CSMA/CA (collision avoidance)

Three methods to avoid collisions (Figure 12.16):

- 1) Interframe space
- 2) Contention window and
- 3) Acknowledgments

Figure 12.16 Contention window



CSMA/CA (collision avoidance)

1) Interframe Space (IFS)

- Collisions are avoided by **deferring transmission even if the channel is found idle.**
- When the **channel is idle, the station does not send immediately.**
- Rather, the **station waits for a period of time called the inter-frame space or IFS.**
- After the **IFS time, if the channel is still idle, then, the station waits for the contention-time & finally, the station sends the frame.**
- 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.

CSMA/CA (collision avoidance)

2) Contention Window

- The **contention-window** is an amount of time divided into time-slots.
- A **ready-station** chooses a **random-number of slots** as its wait time.
- In the window, the **number of slots changes according to the binary exponential back-off strategy.**
- For example:

At first time, number of slots is set to one slot and

Then, number of slots is doubled each time if the station cannot detect an idle channel.

CSMA/CA (collision avoidance)

3) Acknowledgment

- There may be a **collision resulting in destroyed-data**.
- In addition, the **data may be corrupted during the transmission**.
- To help guarantee that the receiver has received the frame, we can use

i) Positive acknowledgment and

ii) Time-out timer

CSMA/CA (collision avoidance)

Frame Exchange Time Line

- Two control frames are used:

- 1) Request to send (RTS)

- 2) Clear to send (CTS)

- The procedure for exchange of data and control frames in time

- 1) The source senses the medium by checking the energy level at the carrier frequency.

- ii) If the medium is idle, then the source waits for a period of time called the DCF interframe space (DIFS); finally, the source sends a RTS.

CSMA/CA (collision avoidance)

2) The destination

→ receives the RTS

→ waits a period of time called the short interframe space (SIFS)

→ sends a control frame CTS to the source.

CTS indicates that the destination station is ready to receive data.

3) The source

→ receives the CTS

→ waits a period of time SIFS

→ sends a data to the destination

CSMA/CA (collision avoidance)

4) The destination

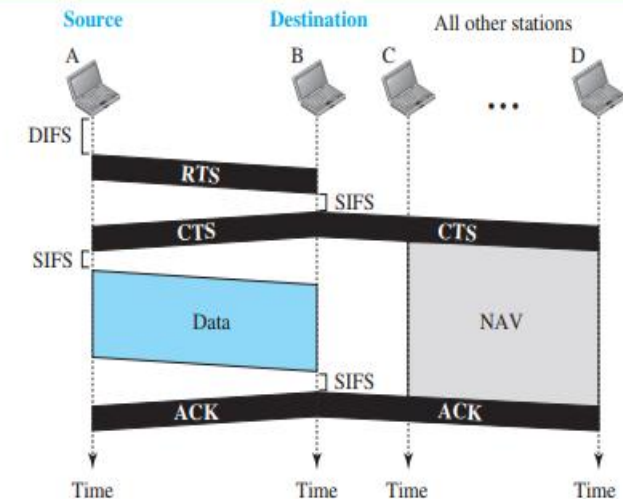
→ receives the data

→ waits a period of time SIFS

→ sends a acknowledgment ACK to the source.

ACK indicates that the destination has been received the frame.

Figure 12.17 CSMA/CA and NAV



CSMA/CA (collision avoidance)

Network Allocation Vector

- When a source-station sends an RTS, it includes the duration of time that it needs to occupy the channel.
- The remaining stations create a timer called a network allocation vector (NAV).
- NAV indicates waiting time to check the channel for idleness.
- Each time a station accesses the system and sends an RTS frame, other stations start their NAV.

CSMA/CA (collision avoidance)

Collision during Handshaking

- Two or more stations may try to send RTS at the same time.
- These RTS may collide.
- The source assumes there has been a collision if it has not received CTS from the destination.
- The backoff strategy is employed, and the source tries again.

CSMA/CA (collision avoidance)

Hidden Station Problem

- Figure 12.17 also shows that the RTS from B reaches A, but not C.
- However, because both B and C are within the range of A, the CTS reaches C.
- Station C knows that some hidden station is using the channel and refrains from transmitting until that duration is over.

CSMA/CA and Wireless Networks

- CSMA/CA was mostly intended for use in wireless networks.
- However, it is not sophisticated enough to handle some particular issues related to wireless networks, such as hidden terminals or exposed terminals.

CONTROLLED ACCESS PROTOCOLS

Here, 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.
- Three popular controlled-access methods are:

1) Reservation

2) Polling

3) Token Passing

CONTROLLED ACCESS PROTOCOLS

1) Reservation

- Before sending data, each station needs to make a reservation of the medium.
- Time is divided into intervals.
- In each interval, a reservation-frame precedes the data-frames.
- If no. of stations = N , then there are N reservation mini-slots in the reservation-frame.
- Each mini-slot belongs to a station.
- When a station wants to send a data-frame, it makes a reservation in its own mini slot.
- The stations that have made reservations can send their data-frames.

CONTROLLED ACCESS PROTOCOLS

2) Polling

- In a network, One device is designated as a primary station and Other devices are designated as secondary stations.
- Functions of primary-device:
 - 1) The primary-device controls the link.
 - 2) The primary-device is always the initiator of a session.
 - 3) The primary-device is determining which device is allowed to use the channel at a given time.
 - 4) All data exchanges must be made through the primary-device.

CONTROLLED ACCESS PROTOCOLS

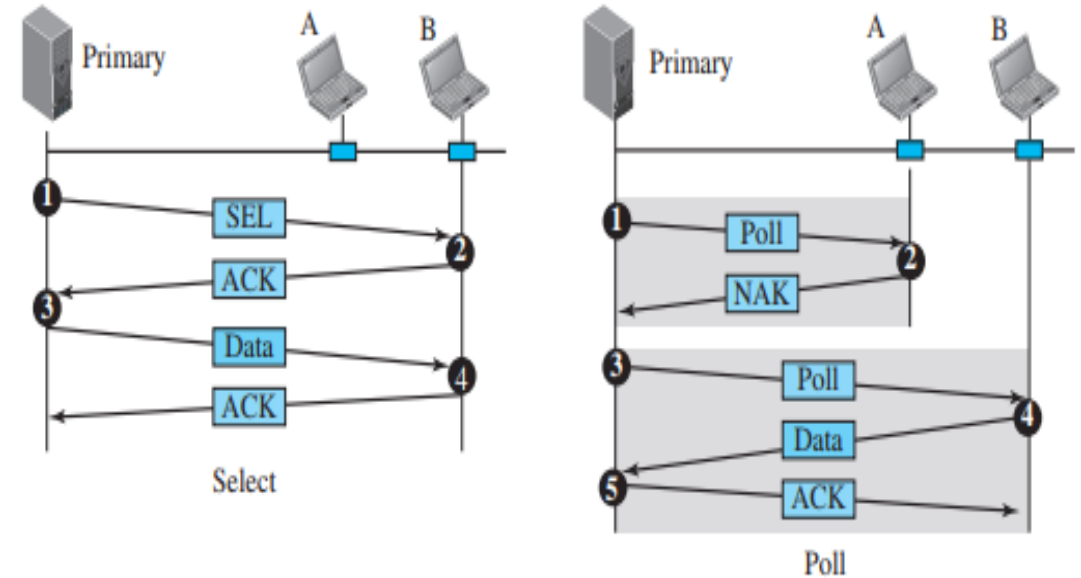
2) Polling

The secondary devices follow instructions of primary-device.

- **Disadvantage:** If the primary station fails, the system goes down.

- Poll and select functions are used to prevent collisions (Figure 12.19).

Figure 12.19 Select and poll functions in polling-access method



CONTROLLED ACCESS PROTOCOLS

a) Select

- If the primary wants to send data, it tells the secondary to get ready to receive; this is called select function.
- The primary alerts the secondary about upcoming transmission by sending select frame (SEL) then waits for an acknowledgment (ACK) from secondary
- Then sends the data frame and finally waits for an acknowledgment (ACK) from the secondary.

CONTROLLED ACCESS PROTOCOLS

b) Poll

If the primary wants to receive data, it asks the secondaries if they have anything to send; this is called poll function.

When the first secondary is approached, it responds either

→ with a NAK frame if it has no data to send or

→ with data-frame if it has data to send.

i) If the response is negative (NAK frame), then the primary polls the next secondary in the same manner.

ii) When the response is positive (a data-frame), the primary

→ reads the frame and

→ returns an acknowledgment (ACK frame).

CONTROLLED ACCESS PROTOCOLS

3) Token Passing

- In a network, the stations are organized in a ring fashion i.e. for each station; there is a predecessor and a successor.

- 1) The predecessor is the station which is logically before the station in the ring.

- 2) The successor is the station which is after the station in the ring.

- The current station is the one that is accessing the channel now.
- A token is a special packet that circulates through the ring.

CONTROLLED ACCESS PROTOCOLS

3) Token Passing

Here is how it works:

- A station can send the data only if it has the token.
- When a station wants to send the data, it waits until it receives the token from its predecessor.
- Then, the station holds the token and sends its data.
- When the station finishes sending the data, the station
 - releases the token
 - passes the token to the successor.

CONTROLLED ACCESS PROTOCOLS

3) Token Passing

Main functions of token management:

- 1) Stations must be limited in the time they can hold the token.
- 2) The token must be monitored to ensure it has not been lost or destroyed.

For ex: if a station that is holding the token fails, the token will disappear from the network

- 3) Assign priorities to the stations and to the types of data being transmitted.
- 4) Make low-priority stations release the token to high priority stations.

CONTROLLED ACCESS PROTOCOLS

Logical Ring

- In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one.

- Four physical topologies

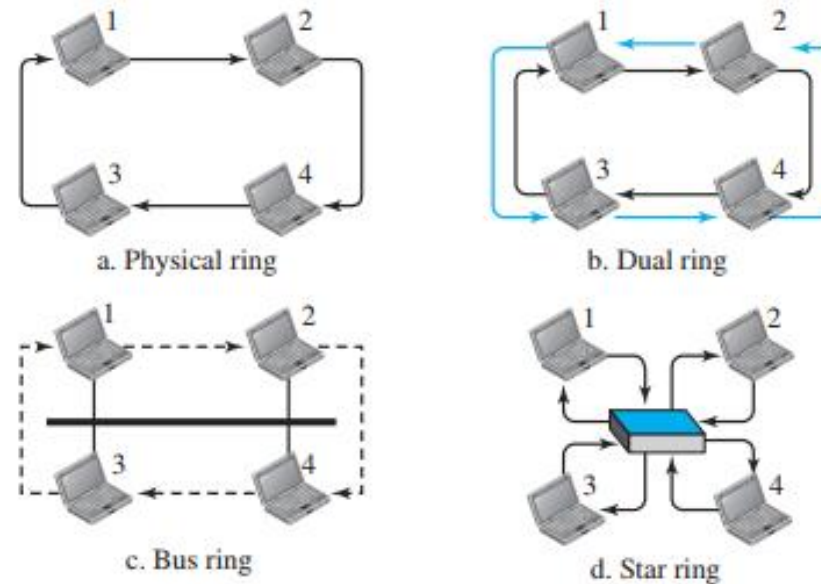
1) Physical ring

2) Dual ring

3) Bus ring

4) Star ring

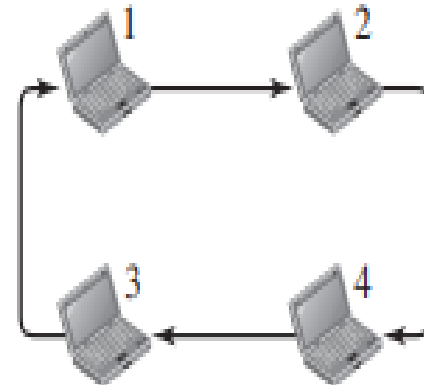
Figure 12.20 Logical ring and physical topology in token-passing access method



CONTROLLED ACCESS PROTOCOLS

1) Physical Ring Topology

- When a station sends token to its successor, token cannot be seen by other stations. (Figure 12.20a)
- This means that the token does not have the address of the next successor.
- Disadvantage: If one of the links fails, the whole system fails.

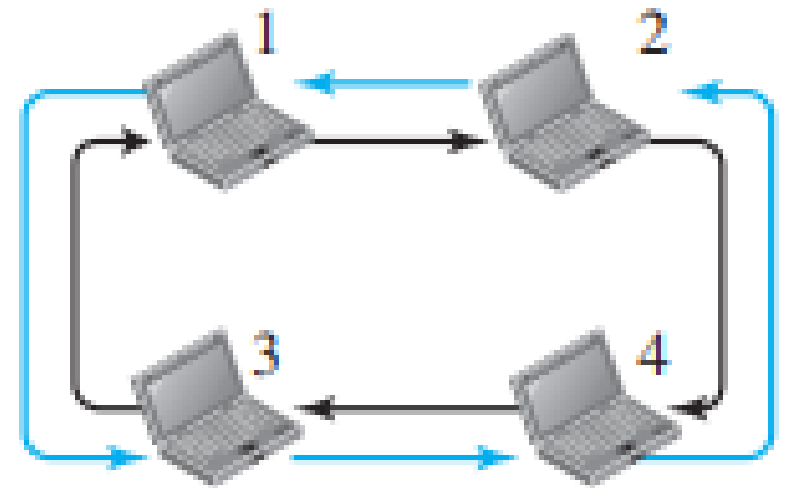


a. Physical ring

CONTROLLED ACCESS PROTOCOLS

2) Dual Ring Topology

- A second (auxiliary) ring is used along with the main ring (Figure 12.20b).
 - operates in the reverse direction compared with the main ring.
 - is used for emergencies only (such as a spare tire for a car).



b. Dual ring

CONTROLLED ACCESS PROTOCOLS

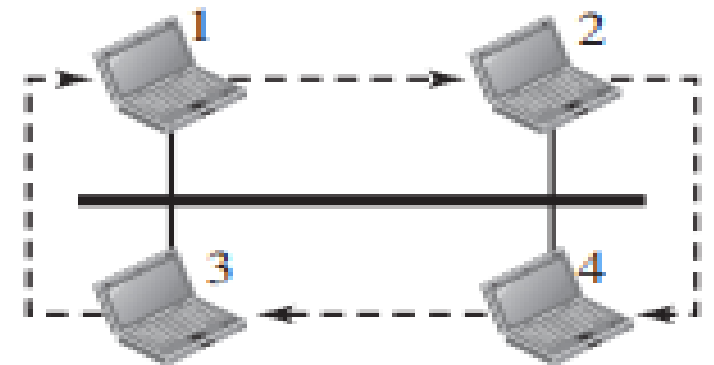
2) Dual Ring Topology

- If the main ring fails, the system automatically combines the 2 rings to form a temporary ring.
- After the failed link is restored, the second ring becomes idle again.
- Each station needs to have 2 transmitter-ports and 2 receiver-ports.
- This topology is used in
 - i) FDDI (Fiber Distributed Data Interface) and
 - ii) CDDI (Copper Distributed Data Interface).

CONTROLLED ACCESS PROTOCOLS

3) Bus Ring Topology

- The stations are connected to a single cable called a bus (Figure 12.20c). This makes a logical ring, because each station knows the address of its successor and predecessor.
- When a station has finished sending its data, the station
→ releases the token and inserts the address of its successor in the token. Only the station gets the token to access the shared media.
- This topology is used in the Token Bus LAN.

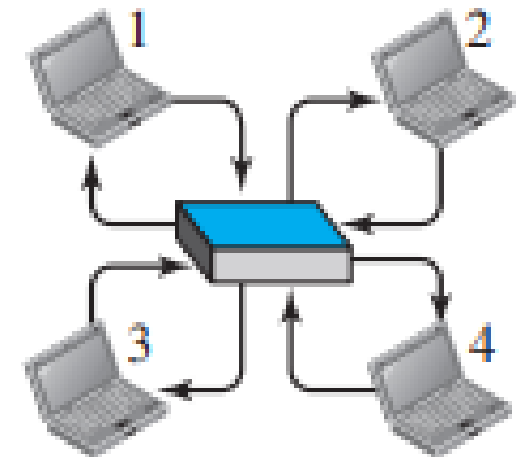


c. Bus ring

CONTROLLED ACCESS PROTOCOLS

4) Star Ring Topology

- The physical topology is a star (Figure 12.20d).
- There is a hub that acts as the connector.
- The wiring inside the hub makes the ring i.e. the stations are connected to the ring through the 2 wire connections.



d. Star ring

CONTROLLED ACCESS PROTOCOLS

4) Star Ring Topology

- **Disadvantages:**

- 1) This topology is less prone to failure because

If a link goes down, then the link will be bypassed by the hub and the rest of the stations can operate.

- 2) Also adding and removing stations from the ring is easier.

- This topology is used in the Token Ring LAN.

STANDARD ETHERNET

- Original Ethernet technology with 10 Mbps data rate. Still retains some core features in modern Ethernet.

Service Characteristics

- **Connectionless:**
 - No setup or termination before sending frames.
 - Each frame is independent; sender sends whenever it has data.
- **Unreliable:**
 - No guarantee of delivery.
 - Corrupted or dropped frames are discarded silently.
 - Error handling is the job of upper-layer protocols (e.g., TCP).

STANDARD ETHERNET

Ethernet Frame Format

(7 fields)

1.Preamble (7 bytes): 56 bits alternating 0/1 for sync (added at physical layer).

Synchronizes clocks between sender and receiver.

2.Start Frame Delimiter – SFD (1 byte): 10101011, marks frame start.

3.Destination Address – DA (6 bytes): Link-layer address of recipient (unicast/multicast/broadcast). MAC address of the intended receiver(s).

STANDARD ETHERNET

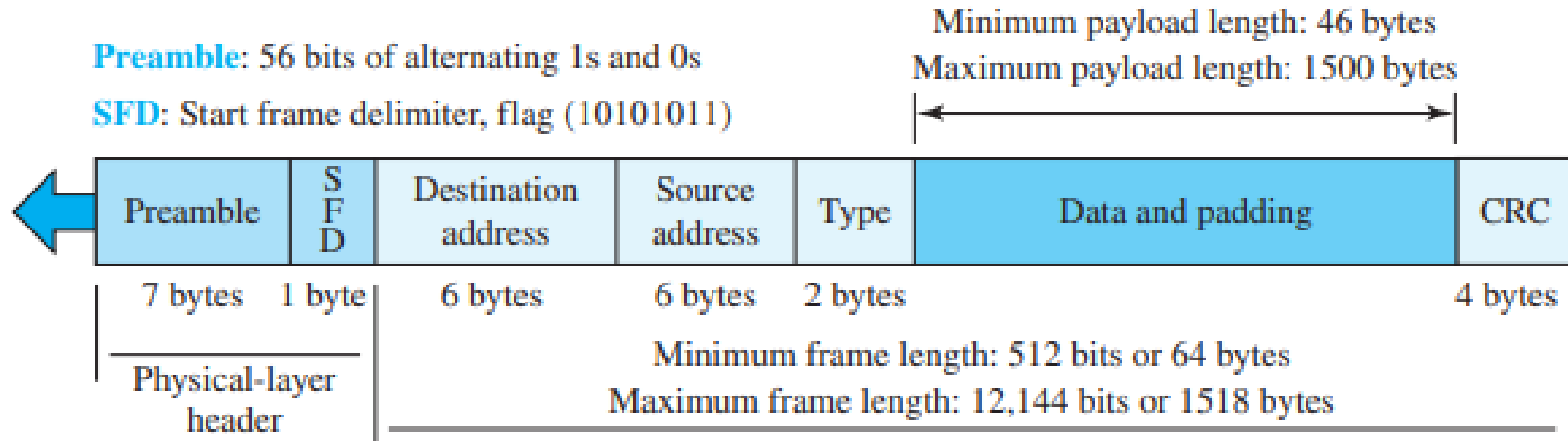
Ethernet Frame Format

(7 fields)

- 4. Source Address – SA (6 bytes):** Sender's link-layer address (always unicast). MAC address of the sender.
- 5. Type (2 bytes):** Identifies upper-layer protocol (IP, ARP, OSPF, etc.). Identifies the upper-layer protocol (e.g., IP, ARP).
- 6. Data (46–1500 bytes):** Payload; pad with 0s if < 46 bytes. Payload from upper layers.
- 7. CRC (4 bytes):** Error detection (CRC-32). Frame discarded if error found. Error detection using CRC-32.

STANDARD ETHERNET

Ethernet Frame Format



Frame Length Rules

Minimum frame: 64 bytes (512 bits) – ensures CSMA/CD works.

Maximum frame: 1518 bytes.

Minimum data: 46 bytes.

Maximum data: 1500 bytes.

STANDARD ETHERNET

Addressing

- **NIC:** Each station has a unique 48-bit MAC address (hex, colon-separated).
- **Transmission:** Bytes sent left-to-right, but bits in each byte sent LSB-first.
- **Types:**
 - **Unicast:** LSB of first byte = 0.
 - **Multicast:** LSB of first byte = 1.
 - **Broadcast:** All bits = 1 (FF:FF:FF:FF:FF:FF).

Encoding/Decoding

- All versions use baseband digital signaling at 10 Mbps.
- **Sender:** Data → digital signal via line coding.
- **Receiver:** Digital signal decoded back to data.