

## *Data Link Layer*

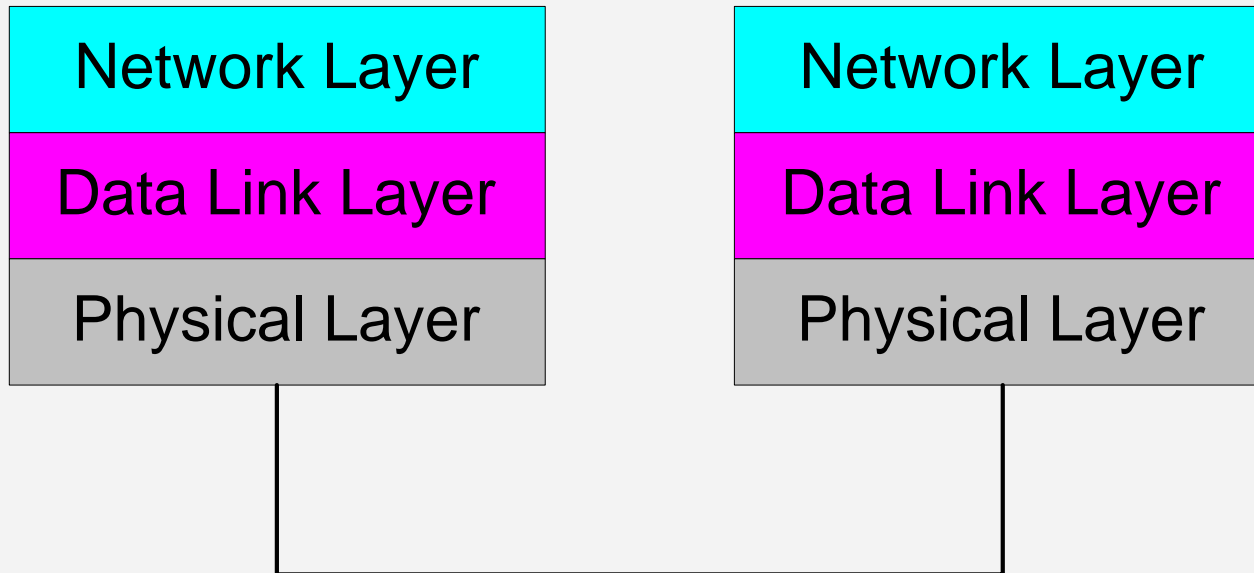
---

# Digital Data Communication Techniques

# Introduction

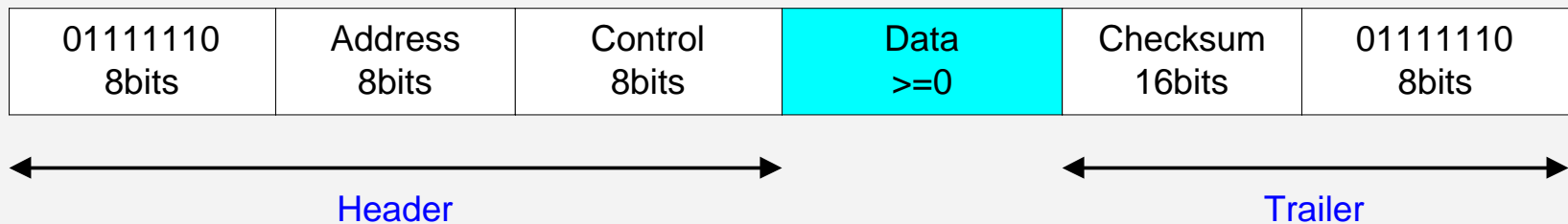
Main Task of the **data link layer**:

- Provide error-free transmission over a link



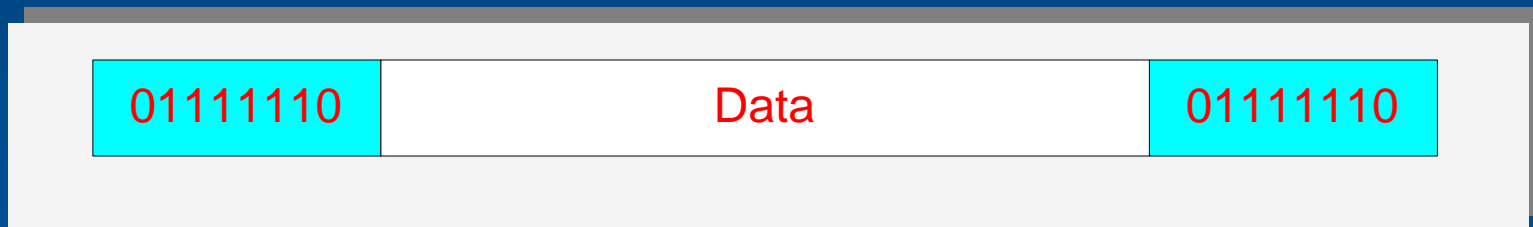
# Introduction

- The PDU at the Data Link Layer (DL-PDU) is typically called a **Frame**. A Frame has a header, a data field, and a trailer
- **Example**



# Framing

- **Problem:** Identify the beginning and the end of a frame in a bit stream
- **Solution (bit-oriented Framing):** A special bit pattern (flag) signals the beginning and the end of a frame (e.g., "01111110")



- **Problem:**
  - The sequence '01111110' must not appear in the data of the frame

# Bit-Oriented Framing and Bit Stuffing

- **'Bit stuffing'**: If the sender detects five consecutive '1' it adds a '0' bit into the bit stream. The receiver removes the '0' from each occurrence of the sequence '11110'

Original  
bit sequence:

01101111111111111111100

After stuffing bits  
at sender:

01101111101111110111110100

Stuffed bits

After stuffing bits  
are removed by  
receiver:

01101111111111111111100

# Error Control

---

Two basic approaches to handle bit errors:

- **Error-correcting codes**
  - Used if retransmission of the data is not possible
  - Data are encoded with sufficient redundancy to correct bit errors
  - **Examples:** Hamming Codes, Reed Solomon Codes, etc.
- **Error-detecting codes plus retransmission**
  - Used if retransmission of corrupted data is feasible
  - Receiver detects error and requests retransmission of a frame.

# *Error Detection Techniques*

---

- Error Detection Techniques:
  - Parity Checks
  - Cyclic Redundancy Check

# Parity Checks

## General Method:

- Append a **parity bit** to the end of each character in a frame such that the total number of 1's in a character is:
  - even (even parity) or
  - odd (odd parity)
- **Example:** With ASCII code, a parity bit can be attached to an 7-bit character
  - ASCII "G" = **1 1 1 0 0 0 1**
  - with even parity = 11100010
  - with odd parity = 11100011

# Cyclic Redundancy Codes (CRC)

---

## General Method:

- The transmitter generates an **n-bit check sequence number** from a given **k-bit frame** such that the resulting  $(k+n)$ -bit frame is divisible by some number
- The receiver divides the incoming frame by the same number
- If the result of the division does not leave a remainder, the receiver assumes that there was no error

# Cyclic Redundancy Codes (CRC)

---

- CRC is used by all advanced data link protocols, for the following reasons:
  - Powerful error detection capability
  - CRC can be efficiently implemented in hardware
  
- We discuss the CRC method in five easy steps:
  - 1. Preliminaries**
  - 2. CRC Encoding Method**
  - 3. Example**
  - 4. Error Detection with CRC**
  - 5. Capabilities of CRC**

# Step 1: Preliminaries

## ■ Modulo-2 Arithmetic:

+	1	0
1	0	0
0	1	0

*	1	0
1	1	0
0	0	0

## ■ Examples of polynomials based on Modulo-2 operations:

1.  $(x+1)(x+1) =$

2.  $(x^2+1)(x^3+x+1) =$

3. If  $F(x)$  contains  $(x+1)$  as a factor, then  $F(1) = 0$

## Step 2: CRC Encoding Method

---

- Let us view each block of data as a polynomial with binary coefficients:

1 0 1 1 0 1 is viewed as  $x^5 + x^3 + x^2 + 1$

Define:

- $M(x)$  Data block is a polynomial (= Message, Frame)
- $P(x)$  "Generator Polynomial" which is known to both sender and receiver (degree of  $P(x)$  is  $n$ )

## Step 2: CRC Encoding Method

---

- (I) Append  $n$  zeros to  $M(x)$ , i.e.,  $M(x) x^n$
- (II) Divide  $M(x) x^n$  by  $P(x)$  and obtain:  
$$M(x) x^n = Q(x) P(x) + R(x)$$
- (III) Set  $T(x) = M(x) x^n + R(x)$ .  $T(x)$  is the encoded message

Note:  $T(x)$  is divisible by  $P(x)$ . Therefore, if the received message does not contain an error then it can be divided by  $P(x)$ .

- **Exercise:** Encode the frame **1 1 0 1 0 1 1 0 1 1**

## Step 3: Encoding

$$M(x) = x^9 + x^8 + x^6 + x^4 + x^3 + x + 1$$

- Generator polynomial is  $P(x) = x^4 + x + 1$
- Degree of  $P(x)$  is 4

$$(I) \quad M(x) x^4 = x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4$$

$$(II) \quad M(x) x^4 = Q(x) P(x) + R(x) \\ = (x^9 + x^8 + x^3 + x) P(x) + x^3 + x^2 + x$$

$$(III) \quad T(x) = M(x) x^4 + R(x) \\ = x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + x$$

Transmitted Bit Sequence: **1 1 0 1 0 1 1 0 1 1 1 1 1 0**

## Step 4: Error Detection with CRC

---

- Errors can be expressed as Error Polynomials
- For example,

Sent Message:           **1 0 1 1 1 0 1**

Received Message:      **1 1 1 1 0 0 1**

---

**Error:                   0 1 0 0 1 0 0**

- In the example, the Error Polynomial  $E(x)$  is given by:

$$E(x) = x^5 + x^2$$

## Step 5: Error Detection Capability of CRC

---

Note:

- Errors will not be detected by CRC if

$$E(x) / P(x) = B(x) + 0,$$

i.e.,  $E(x)$  contains  $P(x)$  as a factor

- Observe the following trade-off:
  - Large values of  $n$  introduce a lot of overhead, but improve the error detection capability.

## Step 5: Error Detection Capability of CRC

- All single-bit errors are detected, if  $P(x)$  has a factor with at least 2 terms:

$$E(x) = x^i$$

If  $P(x) = (x+1)P'(x)$ , then  $x^i$  cannot contain  $(x+1)$

- All double-bit errors are detected if  $P(x)$  has a factor with at least 3 terms

$$E(x) = x^i + x^j = x^j(x^{k+1} + 1) \quad (\text{for } k = i-j, j < i)$$

- If  $P(x)$  does not divide  $(x^k + 1)$  for any  $k$  up to the max. frame length then all double errors can be detected.

$P(x) = x^{15} + x^{14} + 1$  will not divide  $E(x) = x^k + 1$  for any  $k \leq 32,768$

## Step 5: Error Detection Capability of CRC

---

- If  $P(x)$  contains  $(x+1)$  as a factor then all errors with “odd” number of bits are detected.
  - CRC detects 50% of all errors
- A CRC with  $n$  check bits will detect all burst errors of length  $\leq n$  bits.

Burst Error of Length  $k$ :  $E(x) = x^i(x^{k-1} + \dots + 1)$

# CRC Implementations

- CRC can be efficiently implemented in hardware by a set of XOR gates and a shift register
- The following generator polynomials are widely used:

**CRC-12:**             $P(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$

**CRC-16:**             $P(x) = x^{16} + x^{15} + x^2 + 1$

**CRC-CCITT:**         $P(x) = x^{16} + x^{12} + x^5 + 1$

**CRC-32:**  $P(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$