# Hamming Code in Computer Networks

## 1. Introduction

- **Error detection and correction** are vital in computer networks because transmission errors may occur due to noise, interference, or attenuation.
- **Hamming Code** (proposed by Richard W. Hamming in 1950) is one of the earliest and most widely used **error-correcting codes**.
- It is a **linear block code** that can detect up to **two-bit errors** and correct **one-bit errors** without retransmission.

## 2. Basic Concept

- In Hamming Code, we add **redundant bits (parity bits)** at certain positions in the original data.
- These parity bits are placed at positions that are powers of 2: **1, 2, 4, 8, 16, …**
- Each parity bit covers a specific set of data bits to check even (or odd) parity.

## 3. Formula for Number of Parity Bits

To determine how many parity bits are needed:

$2r \geq m+r+1$
Where:
m = number of data bits

r = number of parity bits

Example:

- For **4 data bits (m=4)**, find r

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

  **This is called a (7,4) Hamming Code.**

## 4. Placement of Parity Bits

Example (7,4 Hamming Code):

| **Position** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| Bit Type | P1 | P2 | D1 | P3 | D2 | D3 | D4 |

# 5. How Parity Bits Are Calculated

Each parity bit checks bits whose positions (in binary) have **1** in the same place as that parity bit.

- **P1** covers positions 1,3,5,7 (binary ending in 1).
- **P2** covers positions 2,3,6,7 (binary with second digit 1).
- **P3** covers positions 4,5,6,7 (binary with third digit 1).

Parity can be **even** or **odd**. In networks, **even parity** is common.

# 6. Example 1: Encoding with (7,4) Hamming Code

Suppose the data bits are: **1011**

### Step 1: Place data in correct positions

| **Position** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| Bits | P1 | P2 | 1 | P3 | 0 | 1 | 1 |

### Step 2: Calculate parity bits (even parity)

- **P1** → covers (1,3,5,7) → (P1,1,0,1) → to make even → **P1 = 0**
- **P2** → covers (2,3,6,7) → (P2,1,1,1) → 3 ones → need one more to make even → **P2 = 1**
- **P3** → covers (4,5,6,7) → (P3,0,1,1) → 2 ones → already even → **P3 = 0**

### Step 3: Final codeword

1011011

(7 bits with parity)
1011011(7 bits with parity)

# 7. Example 2: Error Detection and Correction

Suppose the codeword **1011011** is sent, but due to noise it is received as **1111011**.

### Step 1: Recalculate parity checks

- **C1** = parity of (1,3,5,7) = (1,1,0,1) = odd → error in this group.
- **C2** = parity of (2,3,6,7) = (1,1,1,1) = even → no error.
- **C3** = parity of (4,5,6,7) = (1,0,1,1) = odd → error in this group.

### Step 2: Error position

Binary of checks:

C3C2C1=1012 =510 .
Error at position **5**.

### Step 3: Correct error

Bit at position 5 was **0**, flip it to **1** → corrected codeword = **1111111**.

## 8. Generalization

- **Hamming (7,4)** → corrects 1 error, detects 2 errors.
- Extended Hamming Code (adding one more parity bit for entire code) → detects up to 3-bit errors.
- Widely used in **memory systems (ECC RAM), satellite communication, and networking protocols**.

## 9. Advantages

- Simple to implement.
- Low redundancy compared to simple repetition codes.
- Useful for short messages and reliable single-bit error correction.

## 10. Limitations

- Only corrects **single-bit errors**.
- Can detect (but not correct) **double-bit errors**.

- Not suitable for **burst errors** (multiple adjacent errors).

# 11. Applications in Computer Networks

- **Wi-Fi (802.11 standard)** – uses advanced ECC (inspired from Hamming).
- **Ethernet frame checks** – uses CRC (but Hamming explains foundation).
- **Memory systems (ECC RAM)** – corrects single-bit memory flips.
- **Satellite and wireless communication** – to overcome noise-induced bit flips.