# Week 3 – Data Link Layer



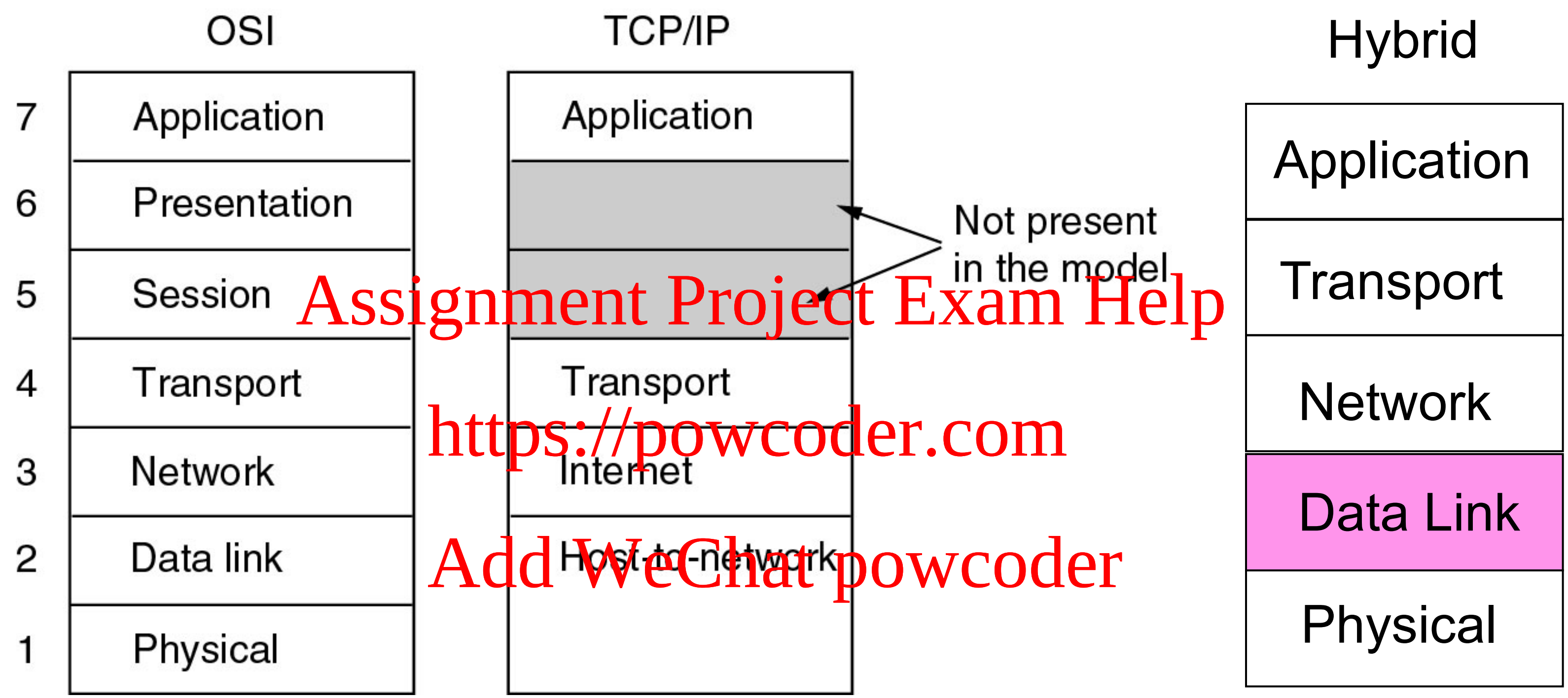
## COMP90007 Internet Technologies



Lecturer: Ling Luo

Semester 2, 2021

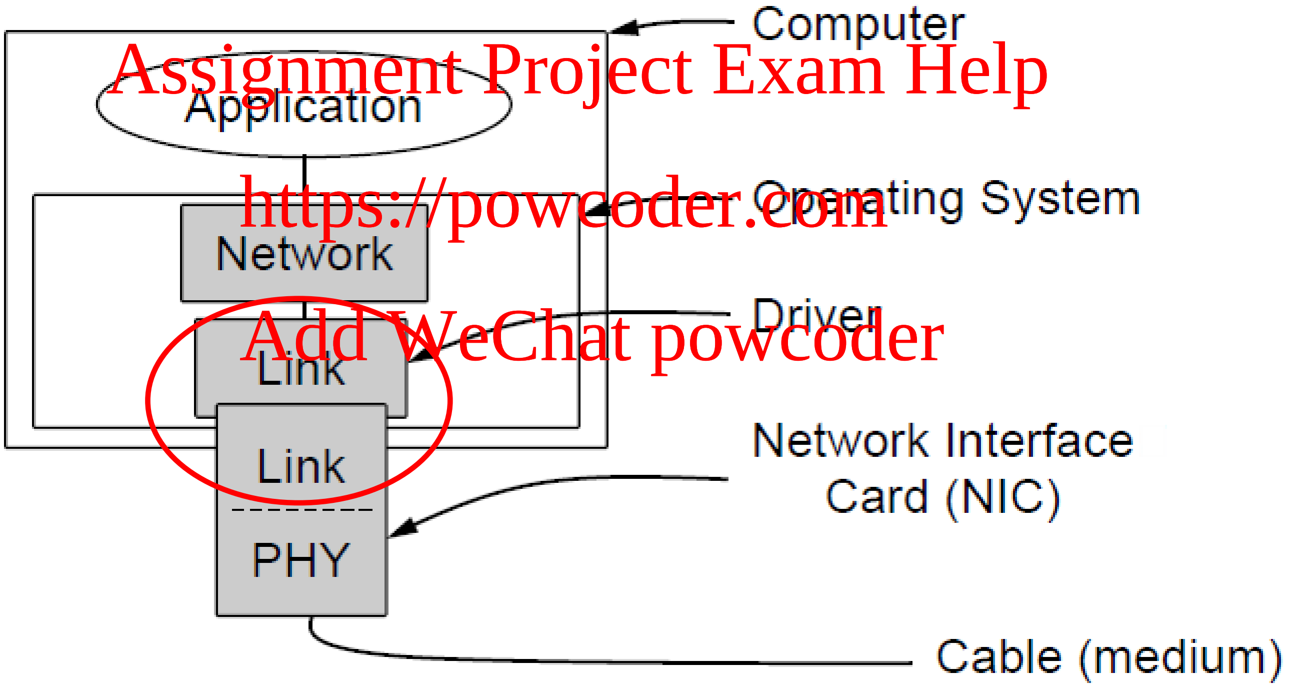
# The Data Link Layer in OSI and TCP/IP






- **Reliable**, **efficient** communication of "**frames**" between two adjacent machines.
- Handles transmission errors and flow control.

# Typical Implementation



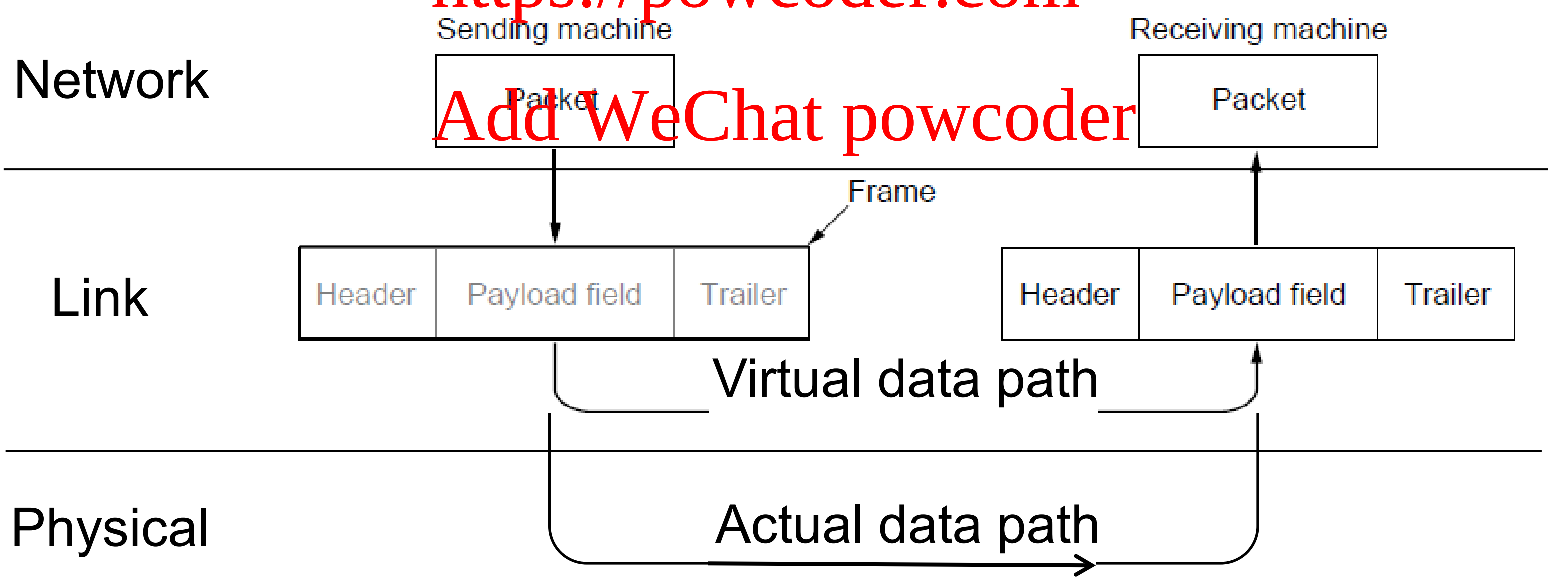
# Functions of the Data Link Layer

- Functions of the data link layer:
  1. Provide a well-defined service interface to network layer
  2. Handling transmission errors
  3. Data flow regulation
- Primary process:
  - Take **packets** from network layer, and encapsulate them into **frames**

# Relation Between Packets and Frames

- Each frame contains a header, a payload and a trailer
- Link layer accepts packets from the network layer, and encapsulates them into frames that it sends using the physical layer; reception is the opposite process





# Type of Services

- **Connection-Oriented vs Connectionless**: Whether a connection is setup before sending a message
- **Acknowledged vs Unacknowledged**: Whether the receiver gives the sender an acknowledgement upon receiving the message

# Services Provided to Network Layer

- Transferring data from the network layer on source host to the network layer on destination host



- Services provided:
  - Unacknowledged connectionless service
  - Acknowledged connectionless service
  - Acknowledged connection-oriented service

# Unacknowledged Connectionless Service

- Source host transmits independent frames to recipient host with no acknowledgement
- No logical connection establishment or release
- No lost frame recovery mechanism (or left to higher levels)
- Applications:
  - Ethernet LANs
  - Real-time traffic, e.g. voice

# Acknowledged Connectionless Service

- Source host transmits independent frames to recipient host with acknowledgement
- No logical connection establishment or release
- Each frame is individually acknowledged, and retransmitted if lost or errors
- Application: Wireless – IEEE 802.11 WiFi

# Acknowledged Connection-Oriented Service

- Source host transmits independent frames to recipient host after connection establishment and with acknowledgement
- Connection established and released (communicate rate and details of message)
- Frames are numbered, counted, acknowledged with logical order enforced
- Application: Unreliable links such as satellite channel

# Framing (1)

- Framing: breaks raw bit stream into discrete units

- Physical layer provides no guarantee a raw stream of bits is error free



- The primary purpose of framing is to provide some level of reliability over the unreliable physical layer





- Checksums can be computed and embedded at the source, then computed and compared at the destination

checksum = f(payload)

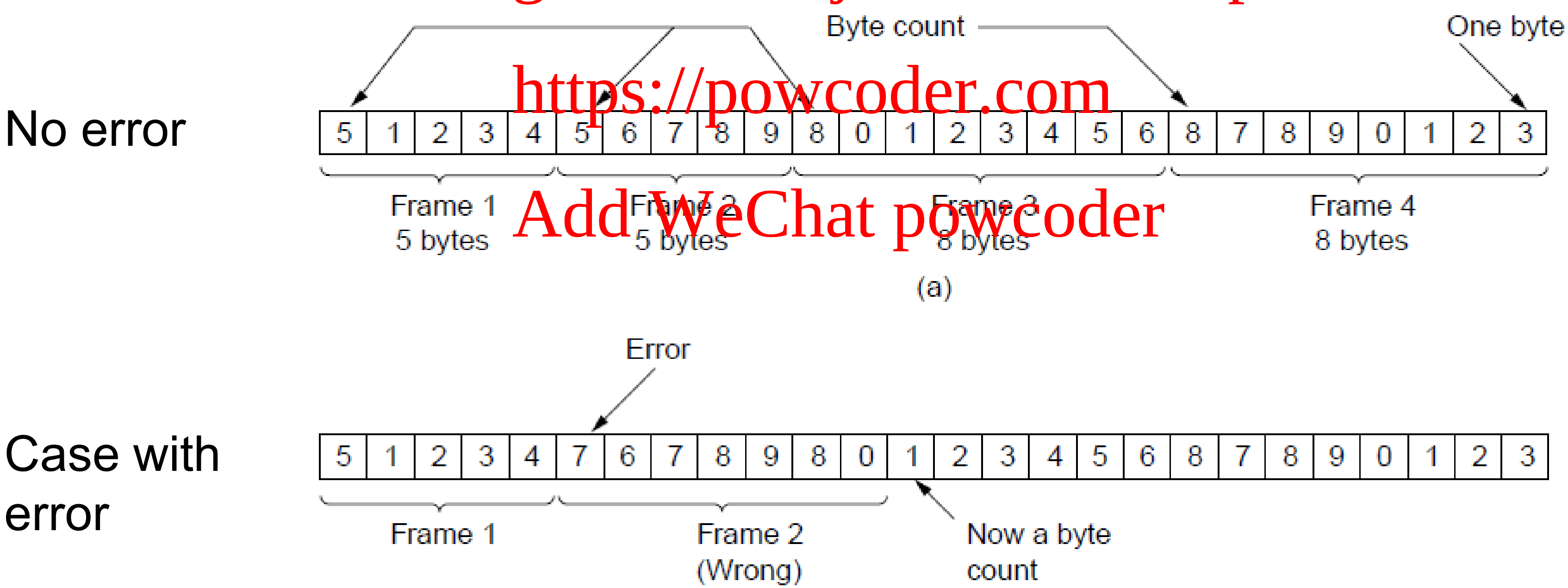
# Framing (2)

- Methods:
  - Character (Byte) count
  - Flag bytes with byte stuffing
  - Start and end flags with bit stuffing
- Most data link protocols use a combination of character count and one other method

# Character Count

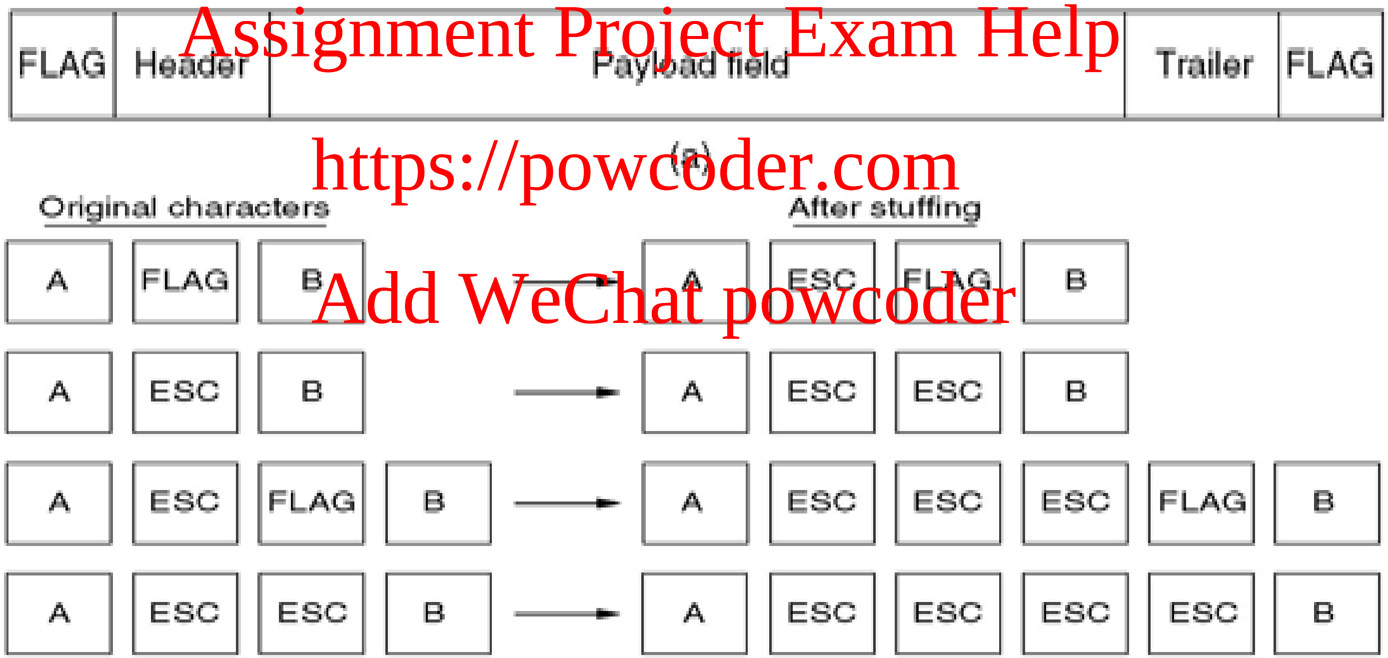
- Uses a field in the frame header to specify the number of characters in a frame





# Flag Bytes with Byte Stuffing

- Each frame starts and ends with a special byte -“flag byte”

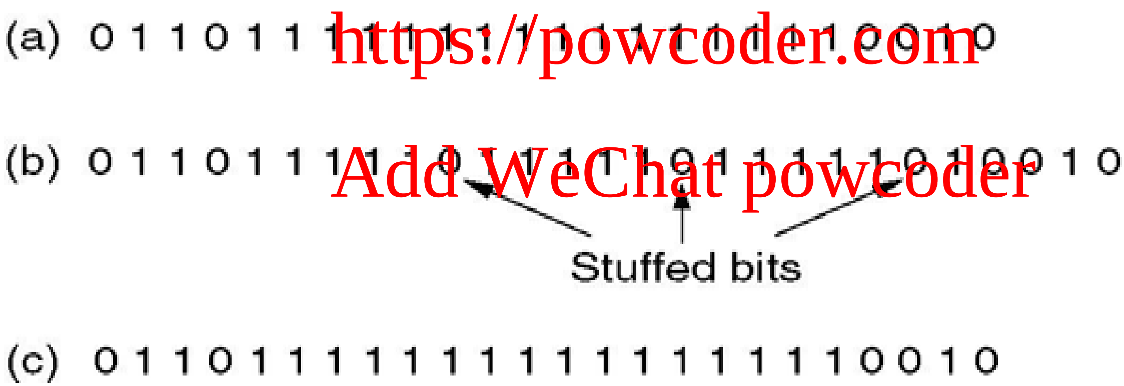



(b)

# Start and End Flags with Bit Stuffing

- Frames contain an arbitrary number of bits
- Each frame begins and ends with a special bit pattern **01111110**






The original data

Sent data

Destuffing at receiver

Insert 0 after five ones (11111)

# Error Control

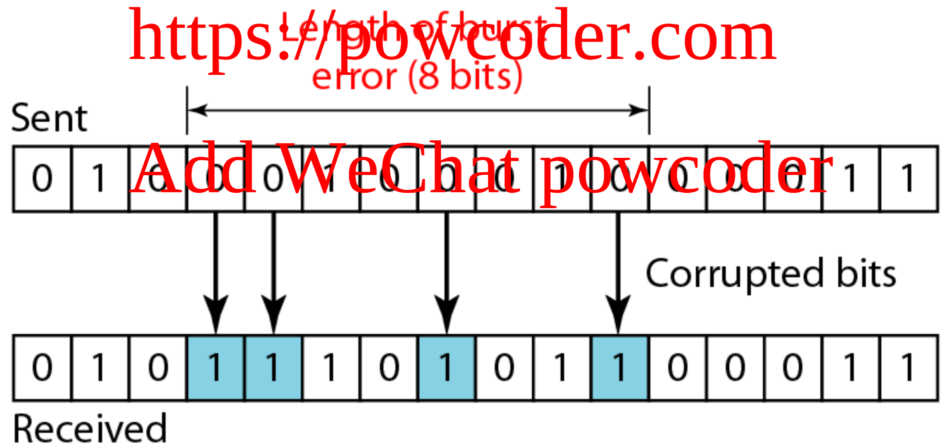
- Adding check bits to ensure that a garbled message by the physical layer is not considered as the original message by the receiver
- Error Control deals with
  - **Detecting** the error, and **retransmitting**
  - **Correcting** the error

# Error Detection and Correction (1)

- Physical media may be subject to errors
- Errors may occur **randomly or in bursts**
  - Single-bit error
  - Burst error: two or more bits have changed





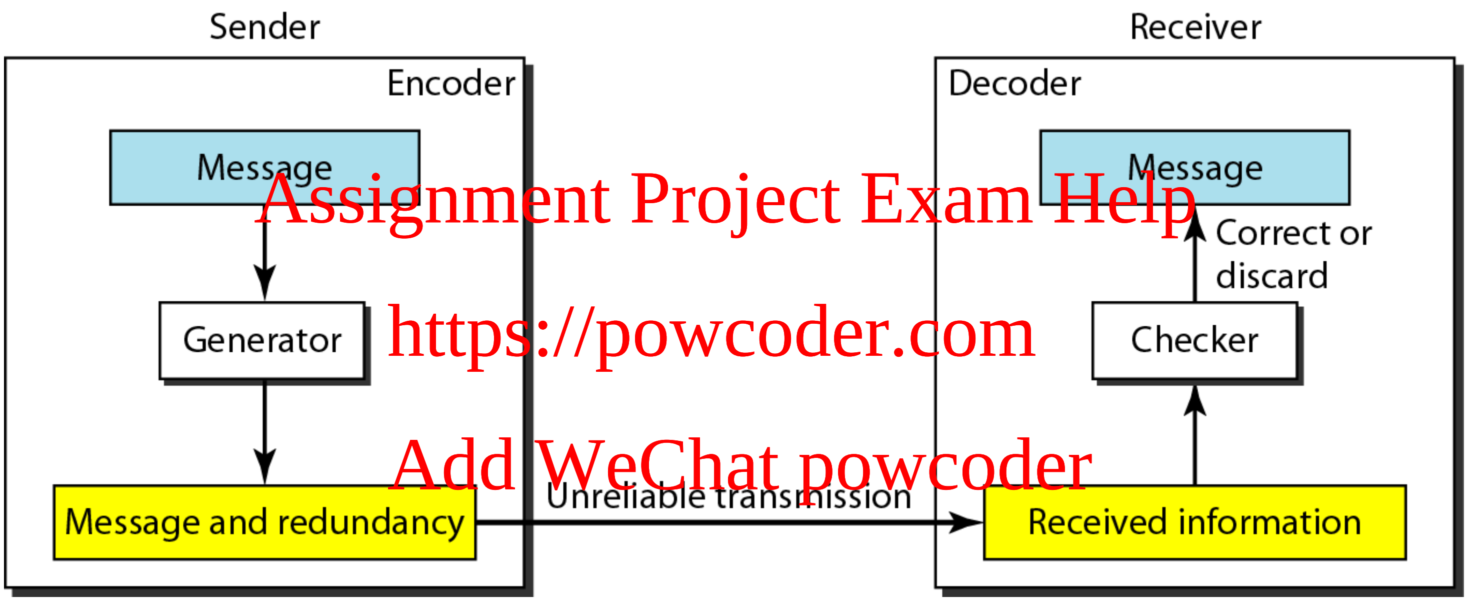





Bursts of errors are easier to detect but harder to resolve

# Error Detection and Correction (2)

- Resolution needs to occur before handing data to network layer






- Key issues
  - ***Fast*** mechanism and ***low computational overhead***
  - ***Minimum amount of extra bits*** sent with the data
  - Detection of ***different kinds of error***

# Example

- Repeat the bits, if a copy is different than the other, there is an error
  - 0 -> 000 and 1 -> 111
- What is the overhead?
- Given the 3 bits received,
  - How many errors can receiver detect?
  - How many errors can receiver correct?
  - What is the minimum number of errors that can fail the algorithm?

# Error Bounds – Hamming Distance

- A code turns **data** of *n* bits into **codewords** of *n+k* bits
- Hamming distance is the **minimum bit flips** to turn one valid codeword into any other valid one.
  - Example with 4 codewords of 10 bits (n=2, k=8):
    - 0000000000
    - 0000011111
    - 1111100000
    - 1111111111



Hamming distance is 5



- A code with Hamming distance:
  - *d+1* → can detect up to *d* errors (e.g., 4 errors above)
  - *2d+1* → can correct up to *d* errors (e.g., 2 errors above)

# Error Bounds – Detection

Q: Why can a code with distance $d+1$ **detect** up to $d$ errors?

- Errors are detected by receiving an invalid codeword, e.g. 00001 11111.
- If there are more than $d$ errors, then the received codeword may become another valid codeword.



- Can receiver detect errors in 11100 00011?

  The receiver cannot detect all 5-bit errors.

# Error Bounds – Correction

Q: Why can a code with distance *2d+1* **correct** up to *d* errors?

- Errors are corrected by **mapping** a received invalid codeword **to the nearest valid codeword**, i.e., the one that can be reached with the fewest bit flips
- If there are more than *d* bit flips, then the received codeword may be closer to another valid codeword than the codeword that was sent



Example: Sending 0000000000 with 2 flips might give 1100000000 which is closest to 0000000000, correcting the error.

But with 3 flips 1110000000 might be received, which is closest to 1111100000, which is still an error