CISC 7332X T6
C07a: Error Handling

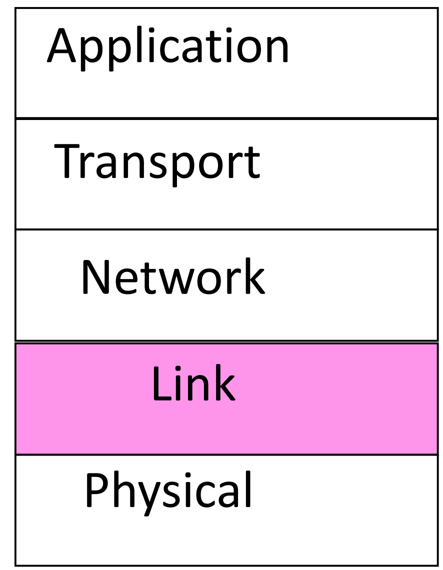
Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Data Link Layer

- Responsible for delivering frames of information over a single link
  - Handles transmission errors
  - Regulates the flow of data



Design Issues in Data Link Layer

- Frames
- Possible services
- Framing methods
- Error control
- Flow control

Outline

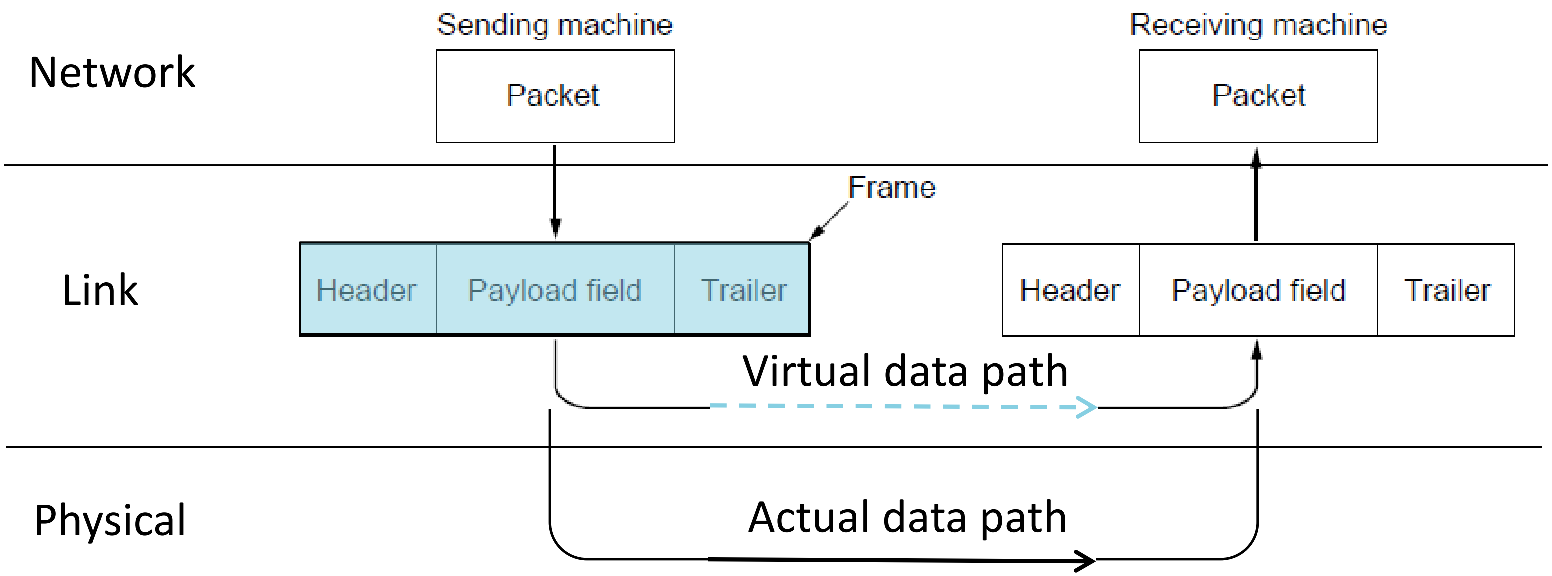
- Frames
- Concept of error detection and correction
- Error detection
- Error correction

Frames

- Sender
  - Link layer accepts <u>packets</u> from the network layer, and encapsulates them into <u>frames</u> that it sends using the physical layer
- Receiver: reception is the opposite process

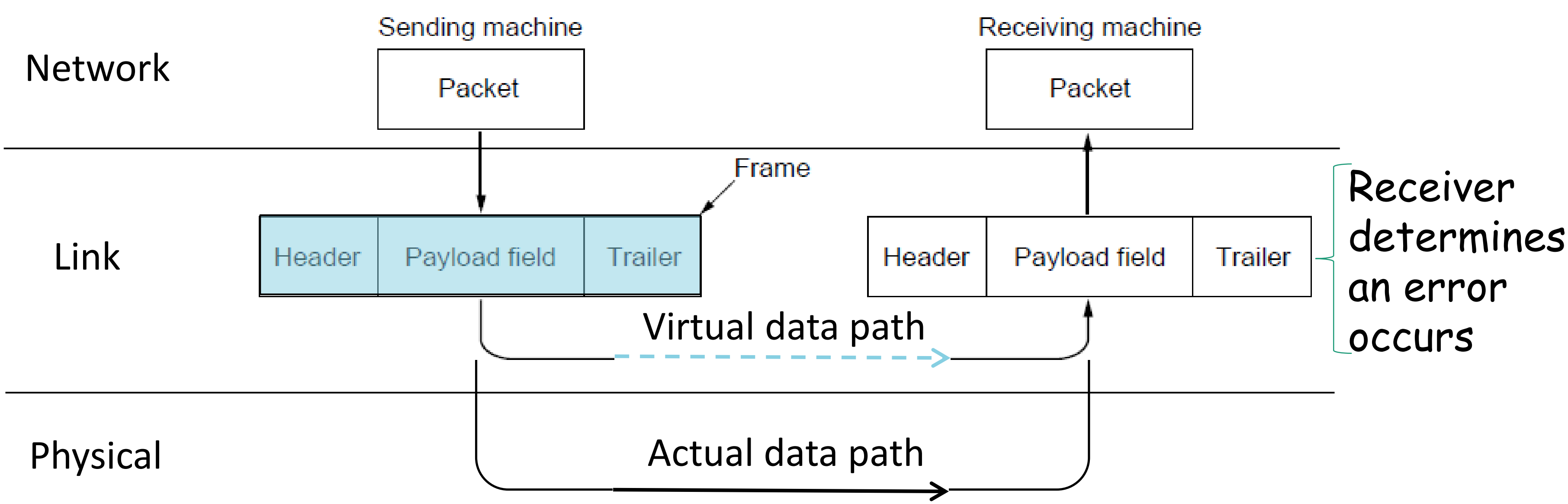
Frames: Interactions between Layers

- The packet from the network layer is the payload of the data link layer



Error Detection and Correction

- Error control repairs frames that are received in error
  - Requires errors to be detected at the receiver



Error Detection and Correction

- Error codes add structured redundancy to data so errors can be either detected, or corrected
- Error detection codes
  - Parity
  - Checksums
  - Cyclic redundancy codes
- Error correction codes:
  - Hamming codes
  - Binary convolutional codes
  - Reed-Solomon and Low-Density Parity Check codes
    - Mathematically complex, widely used in real systems

Two-Dimensional Parity

- Use it to demonstrate the concept of error detection and correction
- Not used in practice (why?)

Parity Bit

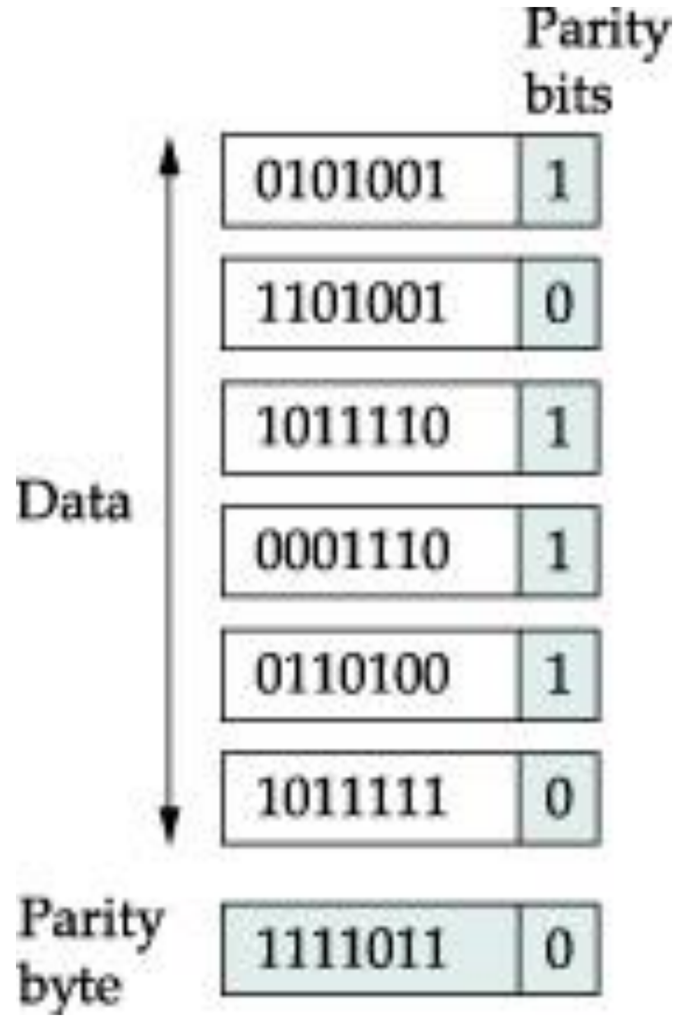
- Append a parity bit to each character
- Even parity
  - Set the parity bit as either 0 or 1 such that the number of 1's in the character is <u>EVEN</u>
- Odd parity
  - Set the parity bit as either 0 or 1 such that the number of 1's in the character is <u>ODD</u>

Error Detection via Parity Bit

- Assume even parity:
  1. Add parity bit at the sender: parity bit is added as the modulo 2 sum of data bits
     - Equivalent to XOR; this is even parity
     - Ex: 1110000 → 11100001
  2. Detect error at the receiver when the sum is wrong
- Examples
  - 1 bit error, 11100101; detected, sum is wrong
  - 3 bit errors, 11011001; detected sum is wrong
  - Error can also be in the parity bit itself
  - 1 bit error, 11100000; detected, sum is wrong
  - 2 bit errors, 11000011; not detected, sum is correct.

Two-Dimensional Parity

- Assume event parity is used
- Parity carried out on both directions
- Each byte has a parity bit
  - Even number of 1's: 1 → parity bit
- Each frame has a parity byte
  - Event number of 1's: 1 → corresponding bit in parity byte



Exercise C07a-1

- Q1: Sending the following message over a link

  H E L O

  determine its two-dimensional parity bits and byte.

  Assume using the ASCII code (**not** the Extended ASCII).
- Q2: In above case, show an example of received "frame" (i.e., data // parity bits and byte) that has detectable error. Include both data bits and parity bits and byte.
- Q3: Show an example of received "frame" (i.e., data // parity bits and byte) that has non-detectable error.

How Good is Two-Dimensional Parity for Error Detection?

- What types of errors does it catch?
  - Any 1-bit error? 2-bit error? 3-bit error? 4-bit error? …
- How much extra data are needed to detect errors?
- How efficient is the algorithms to compute the two-dimensional parity and detect errors?

Two-Dimensional Parity Code as Error Correction Code: 1 Bit Error

0 0 0

0 1 0

Parity bits 0 0 0

Parity bits

- Assuming even parity, is there any bit error?
- Assuming 1 bit error, where is the error?

Two-Dimensional Parity Code as Error Correction Code: 2 Bit Errors

0 0 0

1 1 0

Parity byte → 0 0 0

Parity bit

- Assuming even parity, is there any bit error?
- Assuming 2 bit error, where are the errors?

Two-Dimensional Parity Code as Error Correction Code: 2 Bit Errors?

0 0 0
1 1 0
0 0 0

0 0 0
0 0 0
0 0 0

1 1 0
1 1 0
0 0 0

- Assuming 2 bit error, where are the errors?

Two-Dimensional Parity Code as Error Correction Code: Quality?

- How many bit errors can two-dimensional parity code correct?
  - 1-bit error?
  - 2-bit error?
  - ……
- Is there a systematic method to gauge this?
- How much extra data are needed to correct errors?
- How efficient is the algorithms to compute the two-dimensional parity and detect and correct errors?

Questions?

- Concept of error detection and correction
- Parity and two-dimensional parity
- Quality of error detection and correction codes

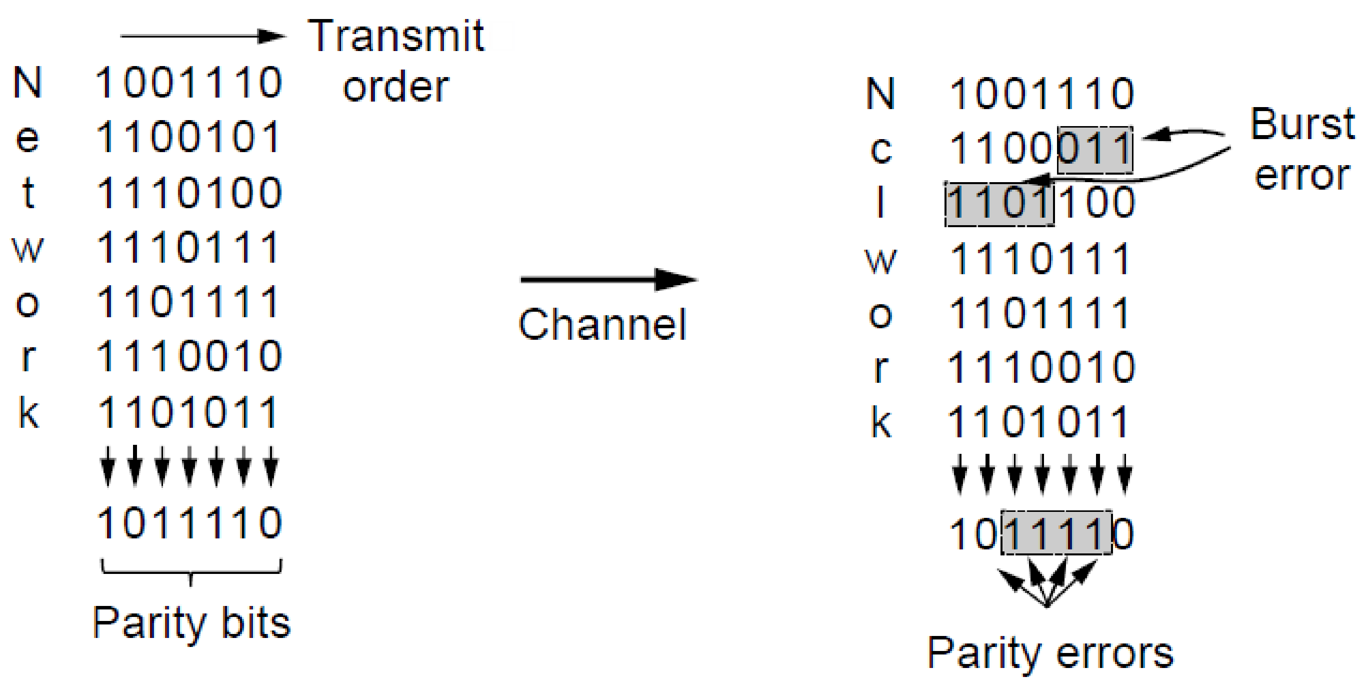
Error Detection

- Parity, revisited
- CRC

Parity as Error Detection Code

- Errors often appear in bursts of bits.
- Interleaving of N parity bits detects burst errors up to N
  - Each parity sum is made over non-adjacent bits
  - An even burst of up to N errors will not cause it to fail

Burst of Bit Errors



Concept of Error Detection, Revisited

- Error detection code
  - Sender has a message M, a n-bit message to send to receiver
  - For error detection, add *k* bits of redundant data to an *n*-bit message
    - Generate a bit string P: M // E
    - Send P to the receiver
- Quality of the error detection code
  - Low redundancy: $k \ll n$
  - High probability of detecting errors
  - Can be implemented efficiently

Cyclic Redundant Check

- Represent *n*-bit string as *n*-1 degree polynomial
  - Bit position as power of each term
  - Digital signal: coefficients are either *0* or *1*
  - Bit string: *11011* as $M(x) = 1\ x^4 + 1\ x^3 + 0\ x^2 + 1\ x^1 + 1x^0 = x^4 + x^3 + x + 1$
- Sender and receiver agrees on a divisor polynomial $C(x)$
  - Digital signal: coefficients are either *0* or *1*
  - Degree of $C(x)$: k
  - Example: $C(x) = x^3 + x^2 + 1$ and $k = 3$

CRC: Example using Polynomials

- Algorithm generating M//E
  - Left shift *M* by *k* bits
    - Example
      - 11011 becomes 11011000
      - New polynomial: $T(x) = M(x)x^k$
  - Get remainder of T(x)/C(x)
    - Example: $(x^4 + x^3 + x + 1)x^3 / (x^3 + x^2 + 1) \rightarrow$
      - Result must be 0 or 1: modular 2 arithmetic → "-" = XOR
      - Quotient: $X^4 + 1$
      - Remainder: $R(x) = x^2+1$
  - Subtract R(x) from T(x)
    - Example
      - $(x^4 + x^3 + x + 1)x^3 - (x^2+1) = x^7+x^6+x^4+x^3+x^2+1$
  - The result is M//E
- Send the result to receiver

CRC: Previous Example using Shift and XOR

- Message: 11011000
- Divisor: 1101

```
   11011000
   1101
⊕-------------
   00001000
       1101
⊕-------------
       0101
   11011000
⊕------------
   11011101
```

Obtain remainder

Obtain M//E

CRC: Error Detection Algorithm

- Algorithm verifying received message
  - Message represented as polynomial $T(x)$
  - Calculate remainder of $T(x) / C(x)$
  - If the remainder is not 0, an error
  - Otherwise, *no errors detected*

Quality of CRC

- Algorithm efficiency
  - Shift and XOR
- Redundancy
  - Depends on $C(x)$
- Error detection probability
  - Depends on $C(x)$

CRC in Practice

- Common CRC Polynomials
  - CRC-8: 1 0000 0111
  - CRC-10: 110 0011 0011
  - CRC-32: used in Ethernet

Exercise C07a-2

- Q1: Sending the following data (1 byte in hexadecimal numbers) over a link

  A1

  determine the "frame" (data // CRC) to be transmit using CRC-8 (divisor = x^8+x^2+x+1)
- Q2: In above case, show an example of received frame (data // CRC) that contains a detectable error.
- Q3: Show an example of received frame that has non-detectable error.

Question?

- A frame can be corrupted
  - Error detection
  - Parity
  - CRC
- Error detection not 100% reliable! protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction

Error Correction

- Error bounds and Hamming distance
- Hamming code
- Convolutional codes

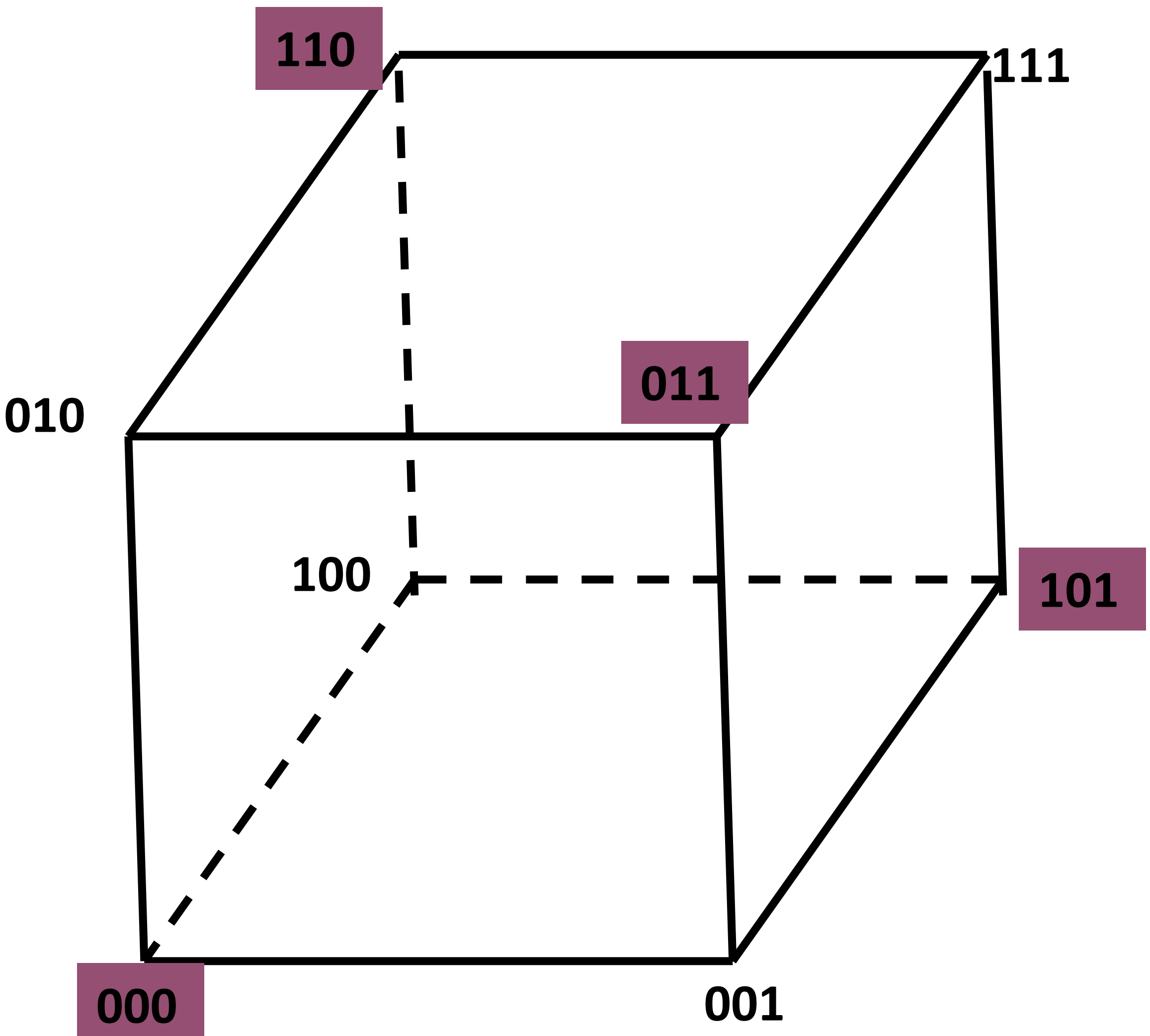
Error Bounds: Hamming distance

- Code turns data of n bits into codewords of n+k bits
  - M → M//K: n → n + k
    - # of total possible bit strings: $2^{(n+k)}$
    - k << (n + k)
- Hamming distance
  - The minimum bit flips to turn one valid codeword into any other valid one.
  - # of bit positions in which two code words differ
  - Example: h(10001001, 10110001) = 3

Code Words and Hamming Distance: Example

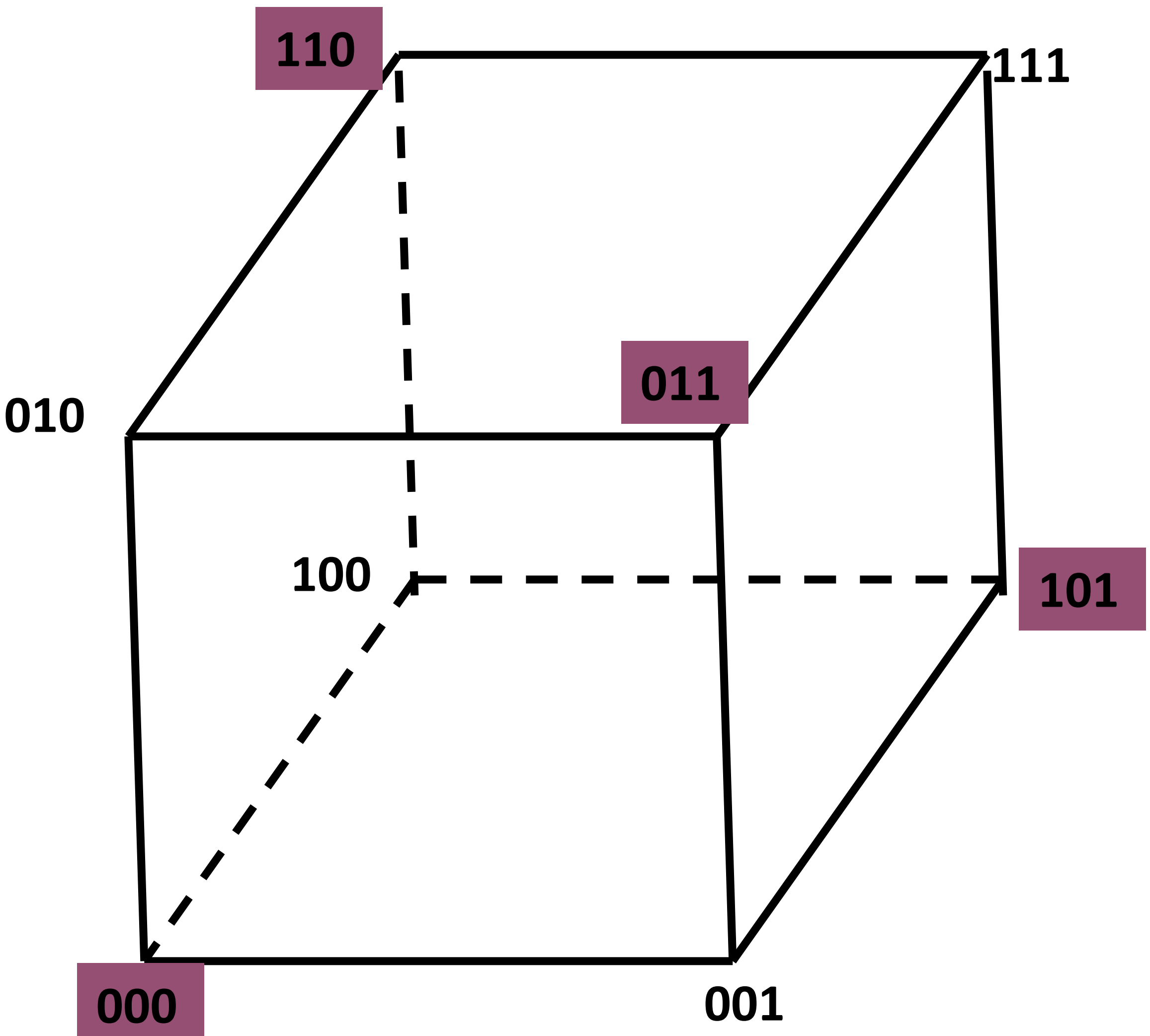
- Message size 2: $n = 2$
- 1 bit parity bit: $k = 1$
- $2^{(n+k)} = 2^3 = 8$
- Select code words: 000, 011, 101, 110
  - # of code words = 4
  - Minimum distance of any pair = 2

Example: Geometric Perspective



- Edge is 1 bit flip
- Detect 1 bit errors
- Cannot detect any 2-bit errors (why)

Example: Error Correction

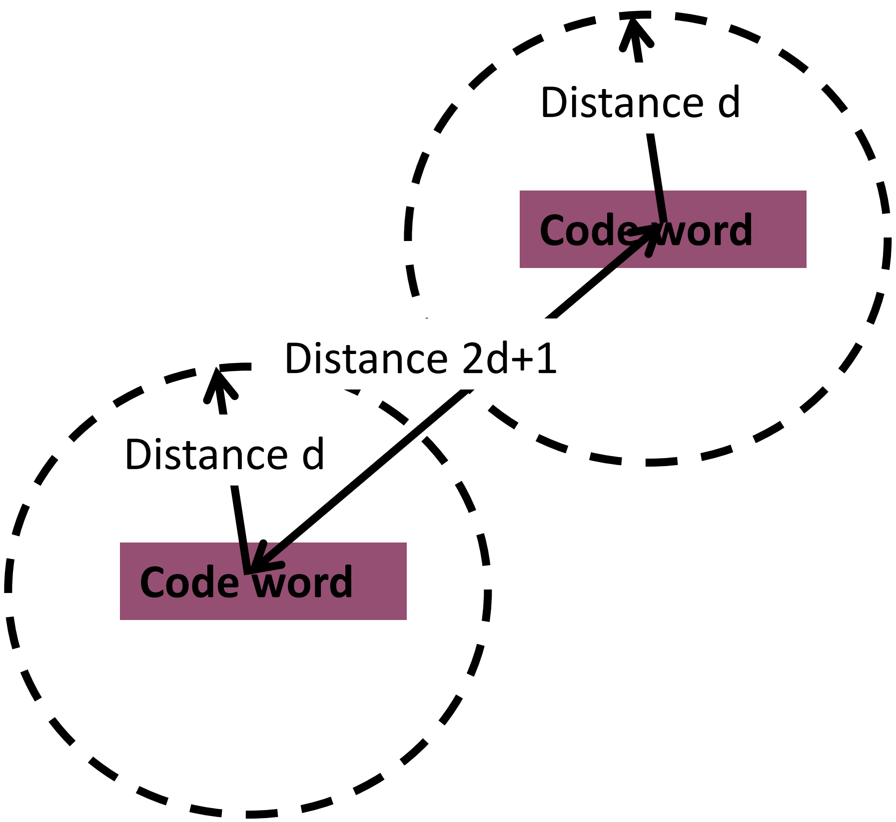


- Edge is 1 bit flip
- Detect 1 bit errors
- Cannot detect any 2-bit errors
- Distance of the code is 2
- Cannot correct any error (why?)

Hamming distance: Detection & Correction

- 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, and 1111111111
  - Hamming distance is 5
- Bounds for a code with distance:
  - 2d+1 – can correct d errors (e.g., 2 errors above)
  - d+1 – can detect d errors (e.g., 4 errors above)

Detection and Correction: Geometric Perspective



- Correct d errors, need distance 2d + 1 code words
  - After d errors, the closest code word remains the correct one.
  - Code words 5 = 2x2+1
    - 00000 00000
    - 00000 11111
    - 11111 00000
    - 11111 11111
    - Correct at most 2 errors

Redundant Data

- Observation
  - 2d + 1 distance code → correct d errors
  - 2d + 1 distance code → detect 2d errors
- Error correction codes generally more redundant
- Error correction or error detection?
  - Error detection example: *m + k* with error rate *r*
    - N (m + k) + r N (m + k) with error correction
  - Error correction example: m + K with error rate *r* and K >> k
    - N (m + K)
  - N (m + k) + r N (m + k) - N (m + K) = N k + r N (m + k) - NK = N (r + rm + rk) – N K = N (r + rm + rk – K)
  - r + rm + rk – K > 0? r + rm + rk – K < 0?

Questions?

- Geometric perspective of error correction and detection
- Hamming distance