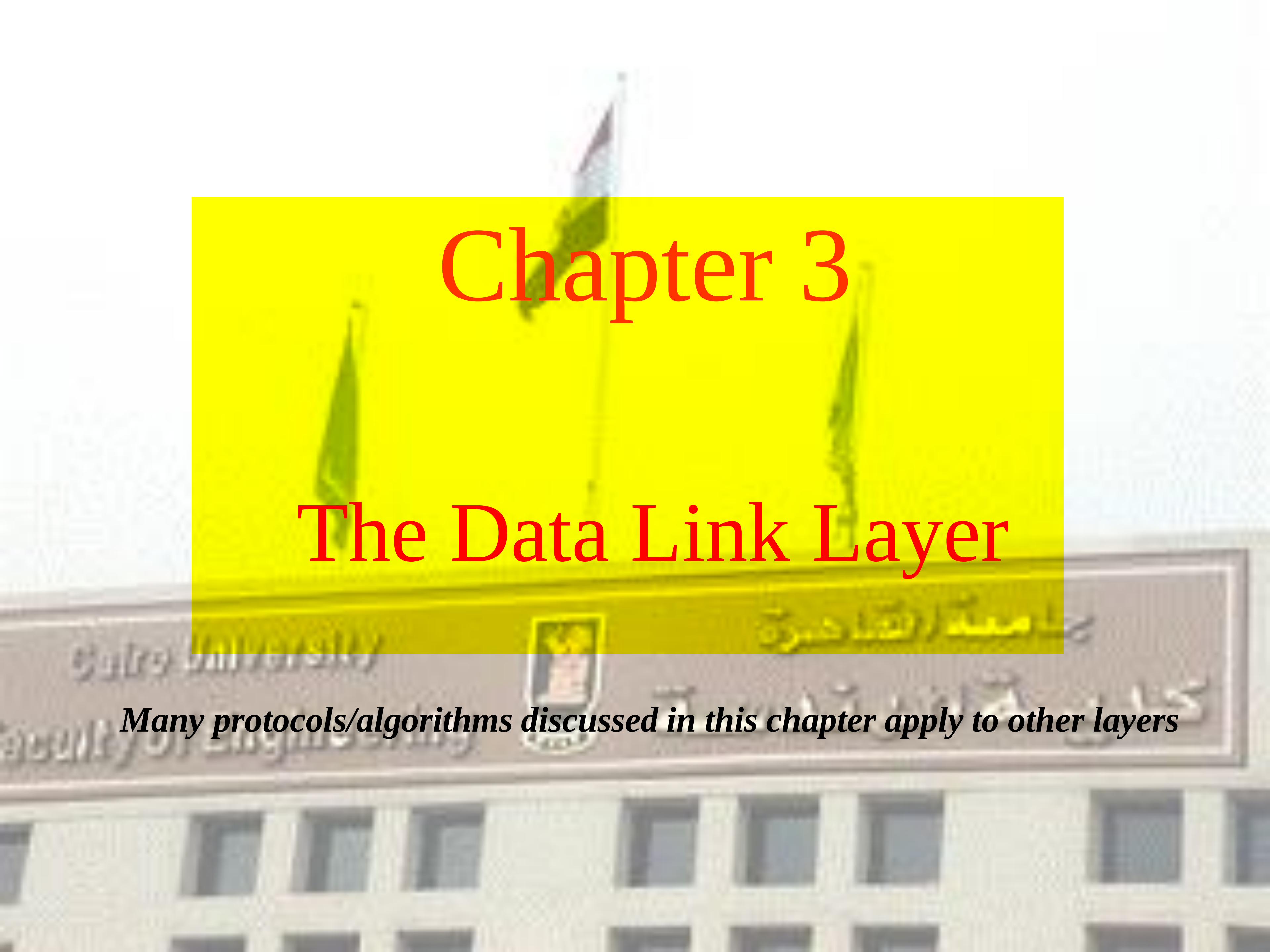


Chapter 3

The Data Link Layer

Many protocols/algorithms discussed in this chapter apply to other layers

2. Error Detection and Correction

- **Two main strategies**
  - **Error-Correcting Codes** (Forward Error Correction) FEC : enable the receiver to deduce what the transmitted data must have been
  - **Error-Detecting Codes** : allow the receiver to deduce that an error has occurred (but not which error) and have it request a retransmission

- **FEC** good for low reliability (e.g. wireless) because retransmission is frequent and may contain error itself
- **Error detection** is good for reliable links because it is cheaper to re-transmit the rare corrupted frames

- An error is a bit the value of which is reversed
  - Error sometimes is referred to as "corruption"

- Types of Errors: Isolated Single bit errors OR Bursty errors
  - Bursty errors cause less packets to be corrupted than random error
  - Much harder to correct because *many* bits are corrupted.

2. Error Detection and Correction

- **What is the basic idea?**

  1. ***Agree on valid symbols***
  2. ***Add check (redundant) bits***

- ***Why do we need redundant bits?***

Error Correcting Codes

1. **Hamming codes.**
2. Binary convolutional codes.
3. Reed-Solomon codes.
4. Low-Density Parity Check codes.

All of these codes add redundancy to the information that is sent. A frame consists of m data (i.e., message) bits and r redundant (i.e. check) bits.

Original Hamming Method

- The message is composed of message bits and check bits
- Use **power of 2** bit position as check bits (positions: Pos.2^0 = 1, 2, 4, 8, 16)
- All other bit position are message bits
- The parity bit at position 2^k checks bits in positions having bit *k* set in their binary representation.
- Example, bit 13 in the data, i.e. 1101, is checked by bits 1000 = 8, 0100=4 and 0001= 1.

Error Bounds – Hamming distance

- Code turns data of m bits into codewords of n bits (m+r bits)

Hamming distance is the number of bit positions in which two codewords differ. It is also the minimum bit flips to turn one valid codeword into any other valid one.

  - Example with 4 codewords of 10 bits:
    - 0000000000, 0000011111, 1111100000, and 1111111111
    - Hamming distance is 5

- Bounds for a code with distance d:
  - d = 2e+1 – can correct e errors (e.g., for d =5 ; you can correct single and double errors)
  - d = e+1 – can detect e errors (e.g., 4 errors)

Hamming Method

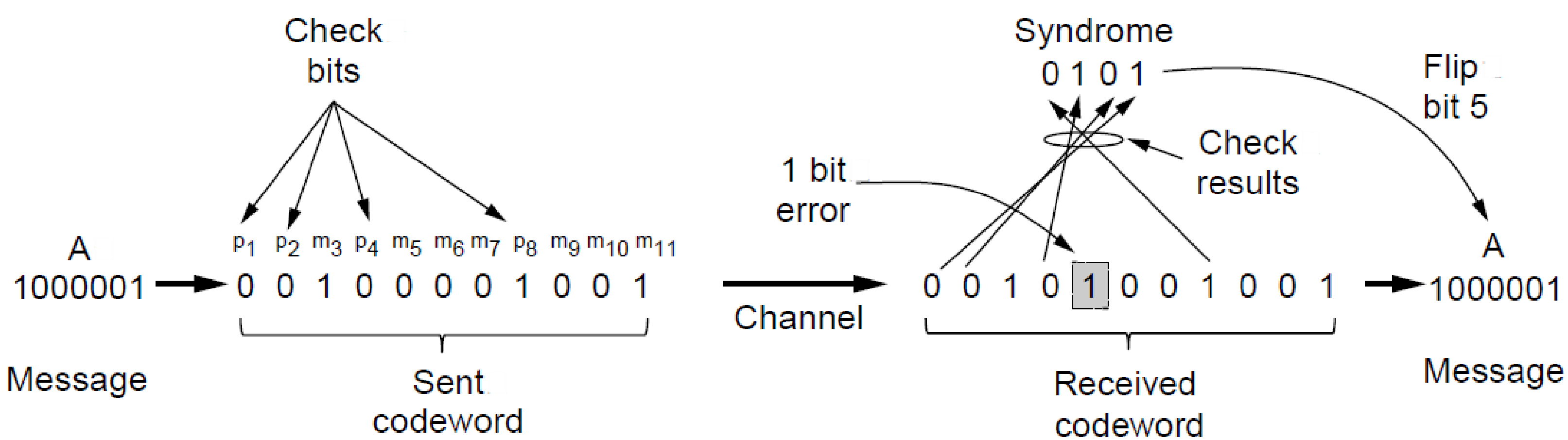
- **For 1-bit error-correction, message m, number of redundant r bits will be calculated as follows:**
  - $\mathbf{(m+r+1) <= 2^r}$
  - Let m=64. Then we need:

    $$r + 65 <= 2^r$$

    Remember **powers of 2**
    - i.e. $r >= 7$

Error Correction – Hamming code

- Hamming code gives a simple way to add check bits and correct up to a single bit error:
  - Check bits are parity over subsets of the codeword
  - Re-computing the parity sums (<u>syndrome</u>) gives the position of the error to flip, or 0 if there is no error



(11, 7) Hamming code adds 4 check bits and can correct 1 error

Even parity: P1 (m3,m5,m7,m9,m11)=0 – P2 (m3,m6,m7,m10,m11)= 0 – P4 (m5,m6,m7)=0 – P8 (m9,m10,m11)=1

Error-Detecting Codes

- Better for reliable channels
- Detect an error and re-transmit
- Far less redundancy bits.

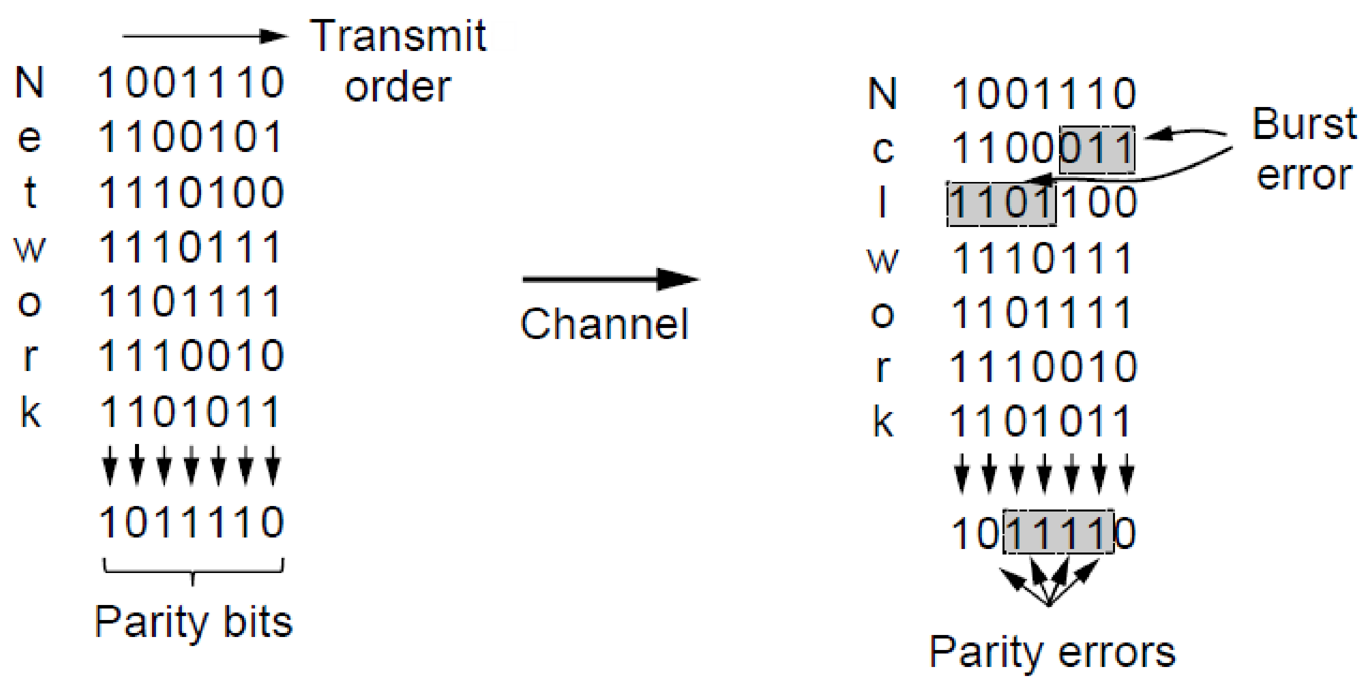
Error Detection – Parity (1)

- Parity bit is added as the modulo 2 sum of data bits
  - Equivalent to XOR; this is even parity
  - Ex: 1110000 → 11100001
  - Detection checks if the sum is wrong (an error)

- Simple way to detect an *odd* number of errors
  - Ex: 1 error, 11100101; detected, sum is wrong
  - Ex: 3 errors, 11011001; detected sum is wrong
  - Ex: 2 errors, 1110110 ; *not detected*, sum is right!
  - Error can also be in the parity bit itself
  - Random errors are detected with probability ½

Error Detection – Parity (2)

- <u>Interleaving</u> 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



Parity Error-Detecting Codes

- Compare Parity bits with Hamming error correction
- Suppose the error rate is 10^{-6}
- Hamming error correction
  - message size is 1000 bits and we send a block of 1000 messages
  - Using $m + r + 1 \leq 2^r$, we need 10 redundant bits per message ($2^{10}=1024$).
  - Single bit error correction 10kbits per Mbits
- Parity Bits error detection + retransmission
  - Single bit burst error ***correction*** overhead is 2001 bits per Mbits (*how did we calculate this number?)*

Parity Error-Detecting Codes

- 1 M of data needs 1,000 check bits.
-

  Once every 1000 blocks (1 M), 1 block needs to be re-transmitted (extra 1001)

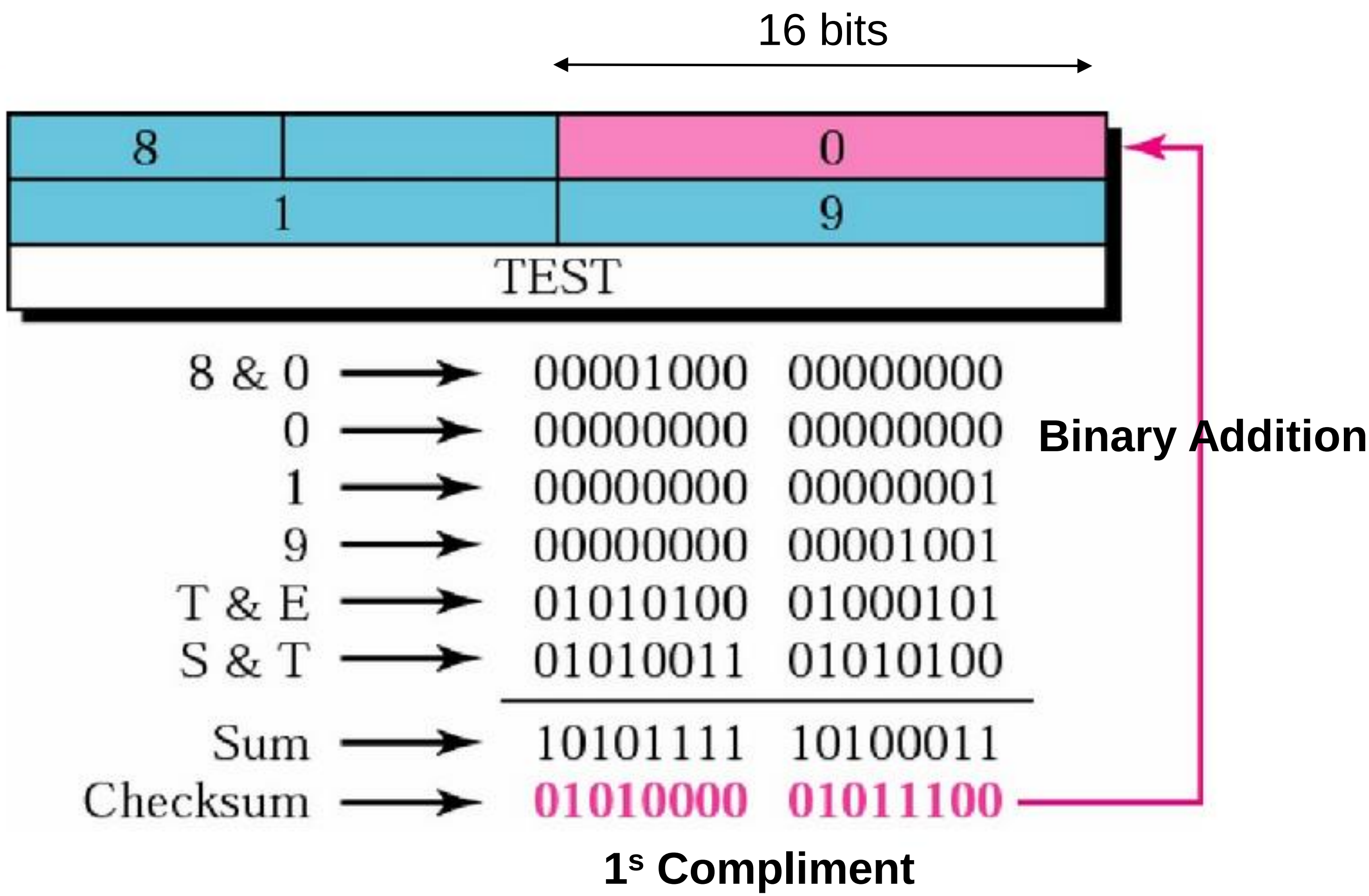
Error Detection – Checksums

- Checksum treats data as N-bit words and adds N check bits that are the modulo 2^N sum of the words
  - Ex: Internet 16-bit 1s complement checksum
  - Simply Binary Add your N-bit words and your checksum is 1s complement of the result

- Properties:
  - Improved error detection over parity bits
  - Detects bursts up to N errors
  - Detects random errors with probability $1\text{-}2^N$
  - Vulnerable to systematic errors, e.g., added zeros

Error Detection – Checksums

Example of checksum calculation



Cyclic Redundancy Check Idea

- We all know that x divides y if <u>remainder</u> of y/x is 0
- Transmitter and receiver agree on **generating polynomial** $G(x)$
  - Both high and low order bits of $G(x)$ must be **1**.
  - Frame must be longer than $G(x)$ $\Rightarrow$The order of $M(x)$ is larger than $G(x)$
- Represent the message (i.e the frame) by the polynomial $M(x)$
- Append **checksum** bits to the message resulting in the polynomial $T(x)$. ($T(x)$ contains both $M(x)$ and the **checksum** bits)
- Checksum bits are calculated such that $G(x)$ divides $T(x)$
- Transmit the frame corresponding to $T(x)$ to the receiver
- At the receiver, compute the *received* $T(x)/G(x)$
- If $G(x)$ does ***not*** divide the received $T(x)$, then there is an ***error***

How to Calculate Transmit Frame

- Given *G(x)* and *M(x)*
- Let *r* be the degree of *G(x)*

  ⇒ *G(x)* has at most (*r+1*) terms
- Let *m* be the maximum number of bits in the message
  - *m* is the size of the payload
- Append *r 0*'s to *M(x)*
  - The frame now contains ***m+r*** bits
  - The frame corresponds to the polynomial $x^r M(x)$ (*Why?*)
- Divide $x^r M(x)$ by *G(x)* using modulo 2 arithmetic
- *What is the number of bits in the remainder*?
- Subtract the remainder from $x^r M(x)$
  - Remember that *subtraction* is just **XOR**.
- *T(x)* is the result of the subtraction.

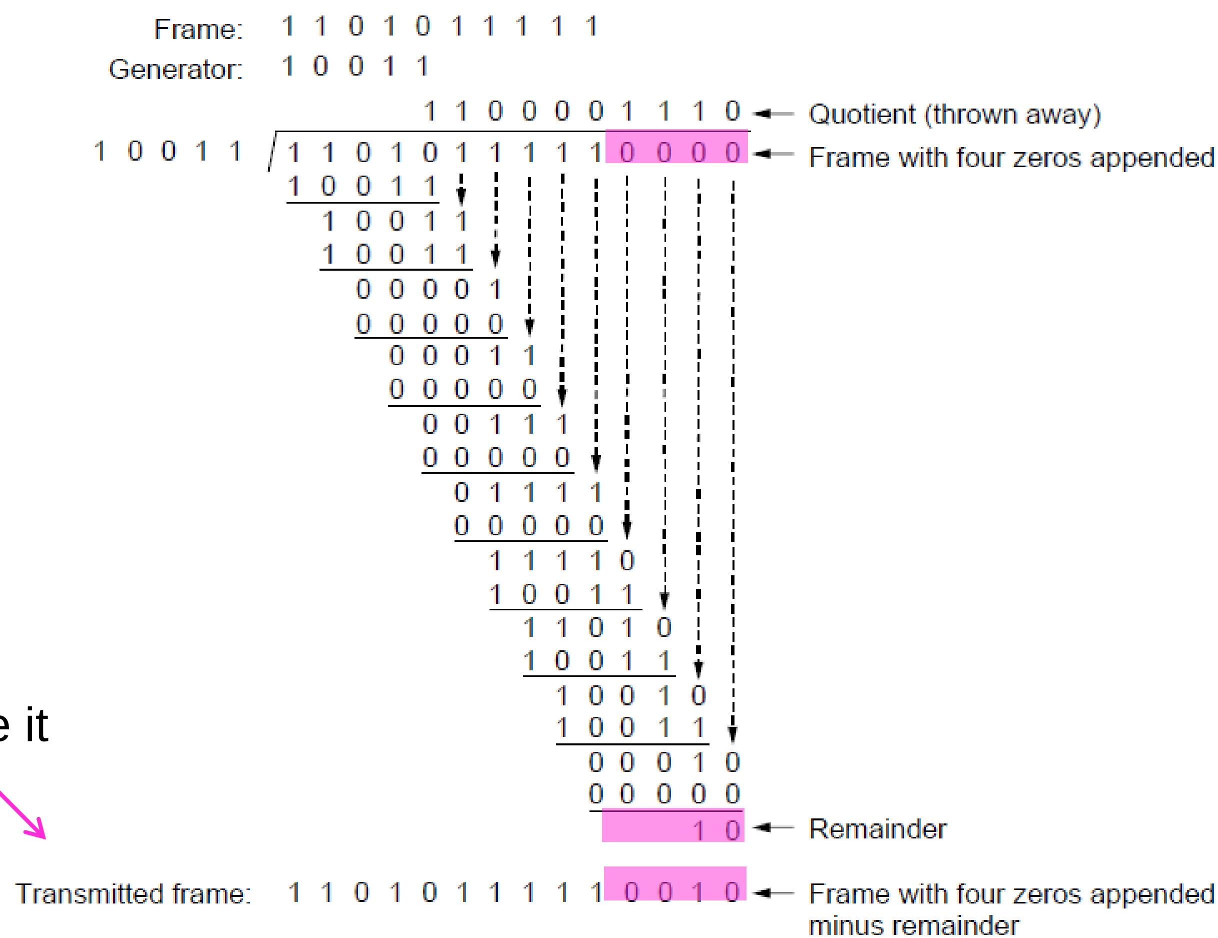
Error Detection – CRCs

- Adds bits so that transmitted frame viewed as a polynomial is evenly divisible by a generator polynomial

Start by adding 0s to frame and try dividing

- Add zeros to end
- Simple Xor and shift
- Get the remainder

Offset by any remainder to make it evenly divisible



Error Detection – CRCs

- Based on standard polynomials:
  - Ex: Ethernet 32-bit CRC

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$$

  - Computed with simple shift/XOR circuits

- Stronger detection than checksums:
  - E.g., can detect all double bit errors
  - Not vulnerable to systematic errors