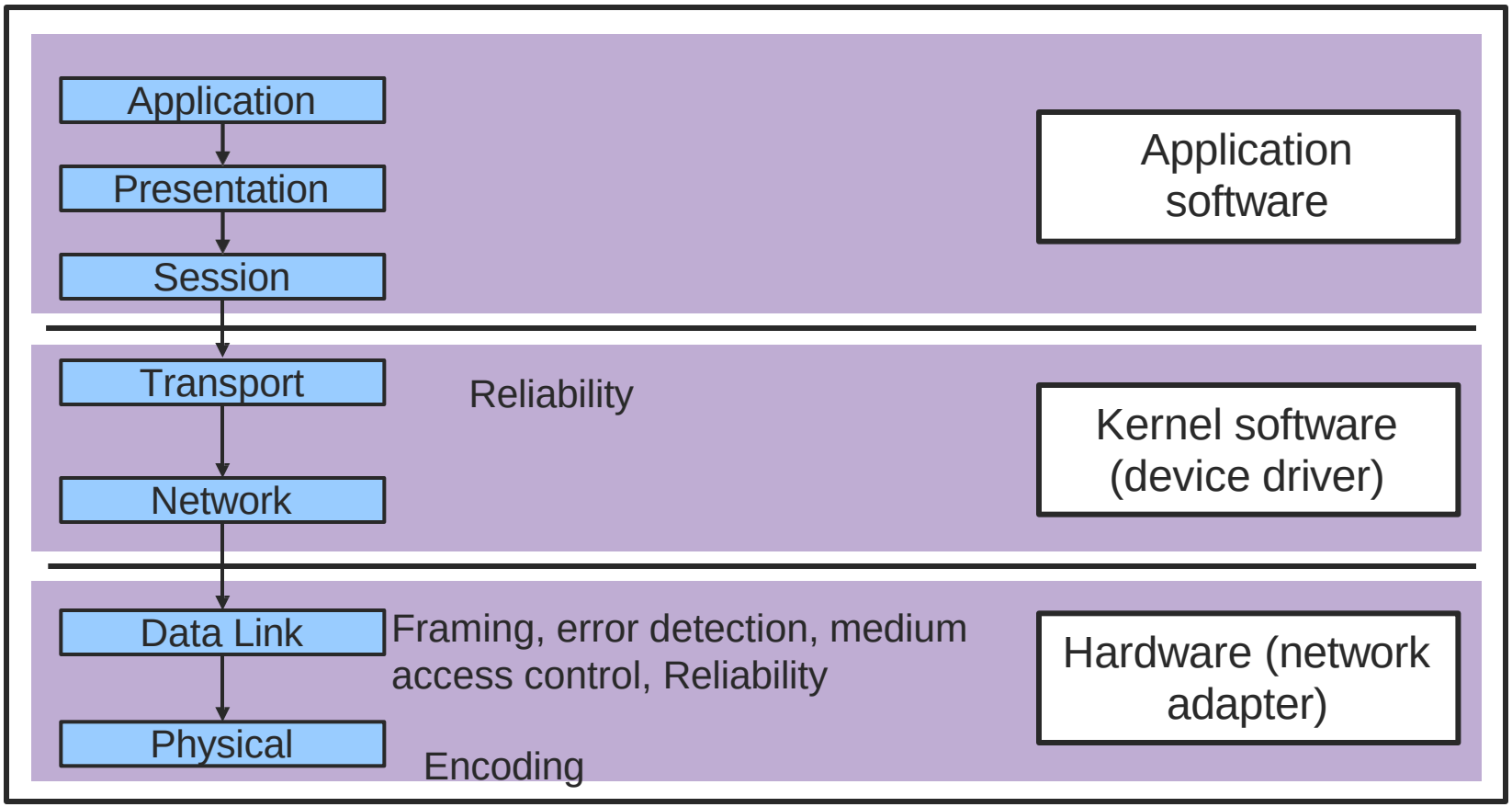
Direct Link Networks

Reading: Peterson and Davie, Chapter 2

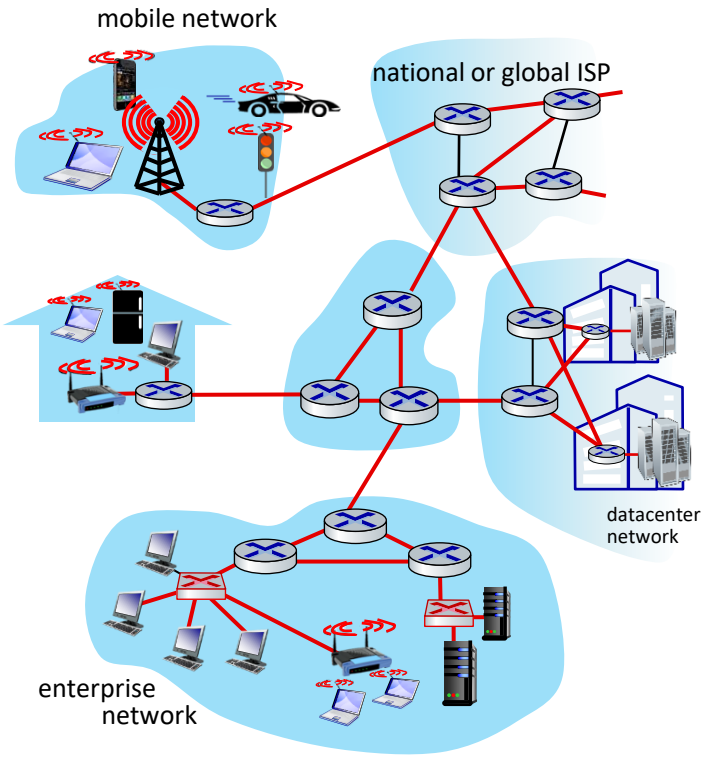
Internet Protocols



Link layer: introduction

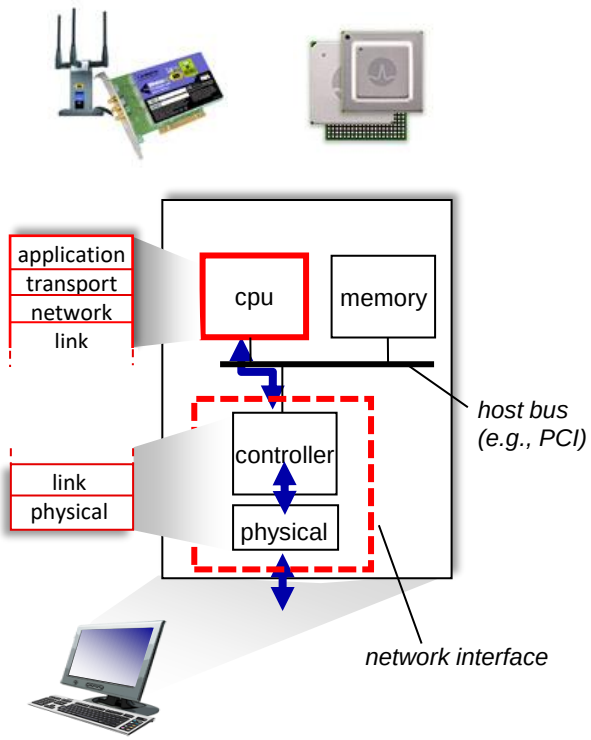
Responsibility:

- Transferring datagram from one node to physically adjacent node over a link
- Communication channels, i.e., links, that directly connect physically adjacent nodes could be:
  - wired , wireless
  - LANs (single network)
- Layer-2 packet: *frame*, encapsulates datagram



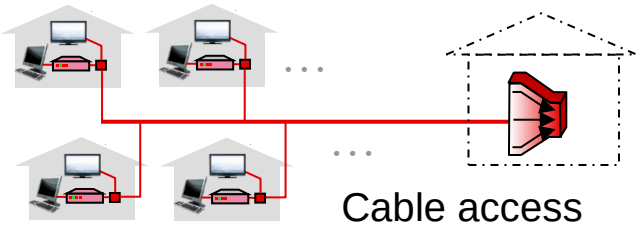
Host link-layer implementation

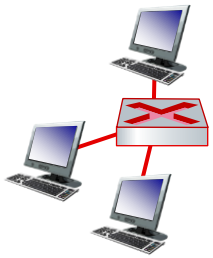
- In each-and-every host
- link layer implemented on-chip or in network interface card (NIC)
  - implements link, physical layer
- Attaches into host's system buses
- Combination of hardware, software, firmware



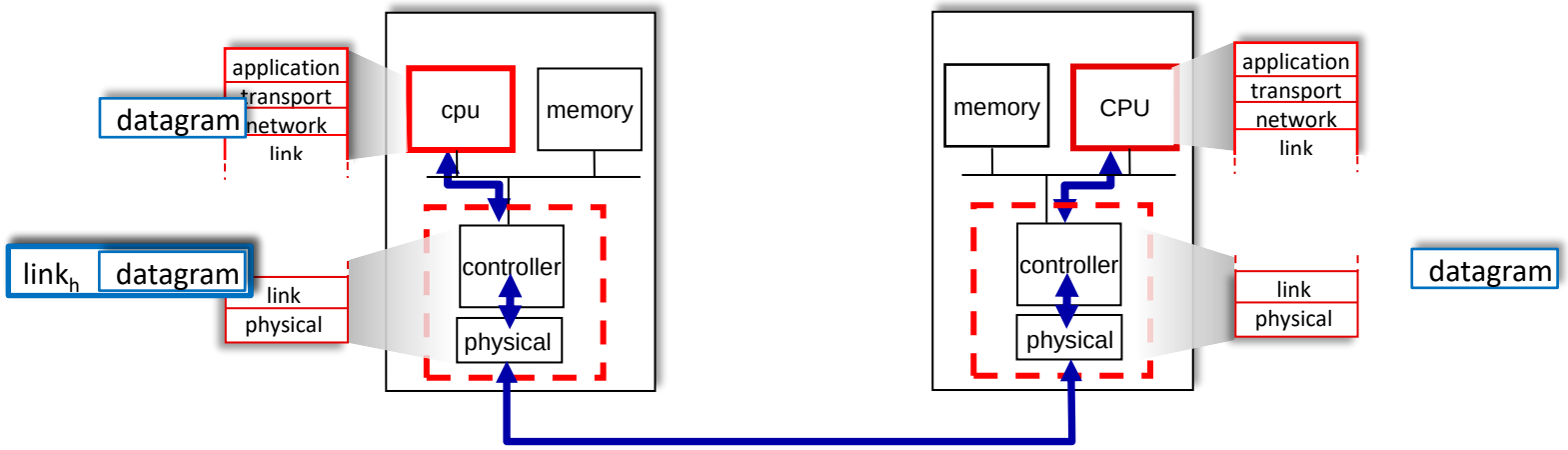
Link layer: services

- Framing, link access:
  - Encapsulate datagram into frame, adding header, trailer
  - Regulate the channel access if shared medium
  - Embed "MAC" addresses in frame headers identify source, destination (different from IP address!)
- Reliable delivery between adjacent nodes
  - Link-level reliability.
  - Different from end-to-end reliability at the transport layer.
- Flow control:
  - Pacing between adjacent sending and receiving nodes
- Error detection:
  - Errors caused by signal attenuation, noise.
  - Receiver detects errors, signals retransmission, or drops frame
- Error correction:
  - Receiver identifies *and corrects* bit error(s) without retransmission
- Half-duplex and Full-duplex:
  - With half duplex, nodes at both ends of link can transmit, but not at same time





Interfaces communicating



Sending side:

- Encapsulates datagram in frame
- Adds error checking bits, reliable data transfer, flow control, etc.

Receiving side:

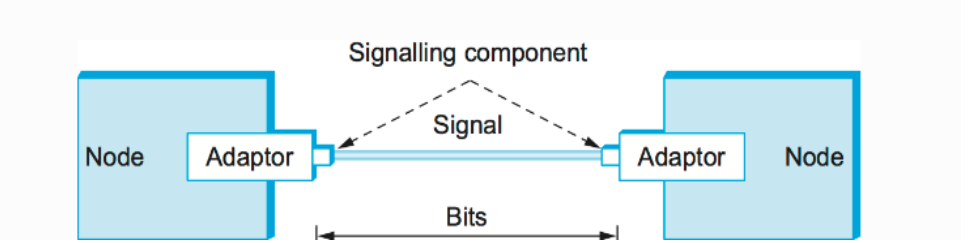
- looks for errors, reliable data transfer, flow control, etc.
- Extracts datagram, passes to upper layer at receiving side

Functionality

- **Encoding**
- Framing
- Error detection
- Multiple access media (MAC examples)
- Reliable transmission

First Steps First

- How bits can be transmitted from one node to the other?



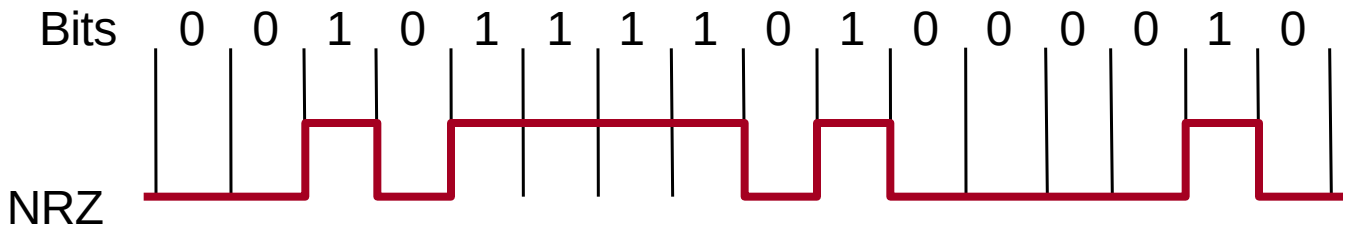
  - Bits → Signals
    - propagate over physical links
    - electrical or electromagnetic
    - Ignore modulation
  - Focus on high & low discrete signals (voltages, etc)
  - Network adapter (Connects node to a link)

Binary Voltage Encoding

- Common binary voltage encodings
  - Non-return to zero (NRZ)
  - NRZ inverted (NRZI)
  - Manchester (used by IEEE 802.3—10 Mbps Ethernet)
  - 4B/5B

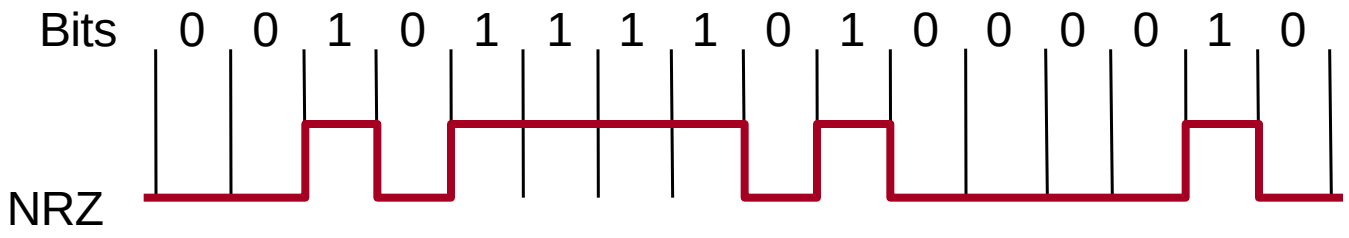
Non-Return to Zero (NRZ)

- Signal to Data
  - High --- 1
  - Low --- 0
- How to determine “high” vs “low”
  - Baseline – the running average of signal power
  - Baseline as the threshold
    - Significantly higher than baseline -> 1
    - Significantly lower than baseline -> 0



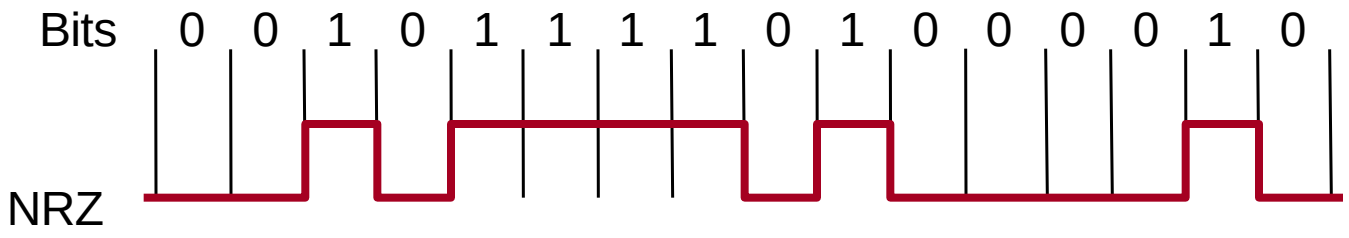
Non-Return to Zero (NRZ)

- Signal to Data
  - High --- 1
  - Low --- 0
- Problem 1
  - Long strings of 1s or 0s causes "**baseline wander**"



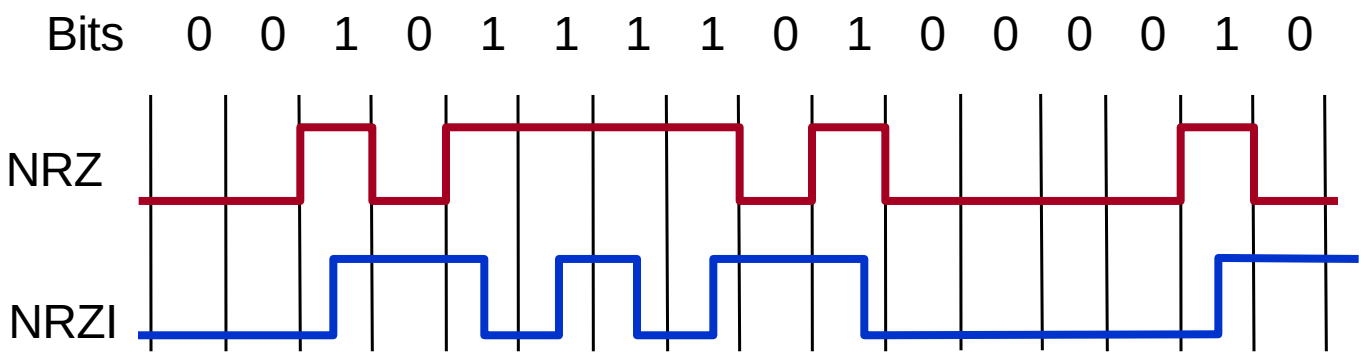
Non-Return to Zero (NRZ)

- Signal to Data
  - High --- 1
  - Low --- 0
- Problem 2
  - Long strings of 1s or 0s: Problem in sender/receiver clock recovery/synchronization
    - To know clock cycle boundary
    - How many 1s or 0s?



Non-Return to Zero Inverted (NRZI)

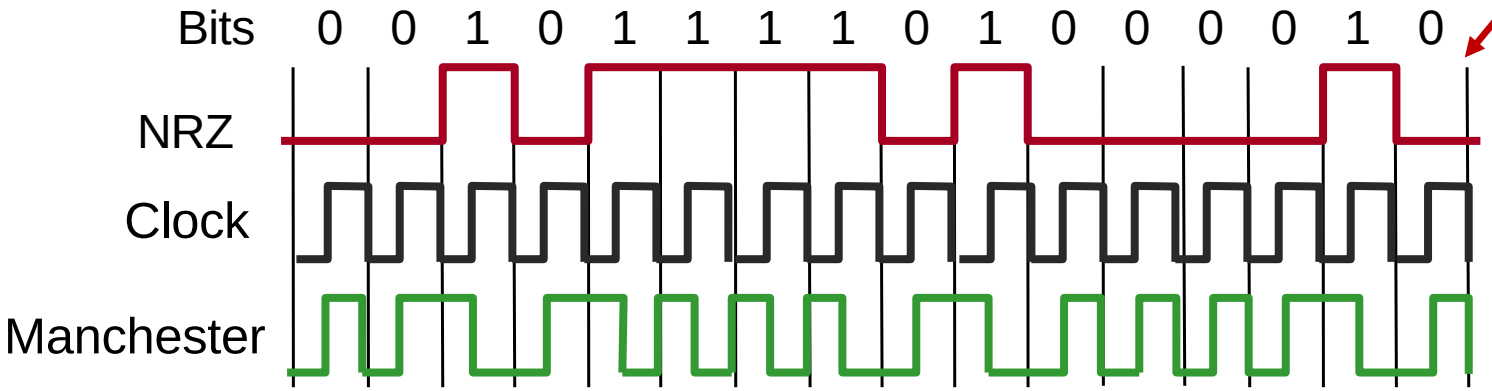
- Signal to Data
  - 1⇨ Transition
  - 0 ⇨ Maintain
- Comments
  - Solves long strings of 1s, but not 0s



Manchester Encoding

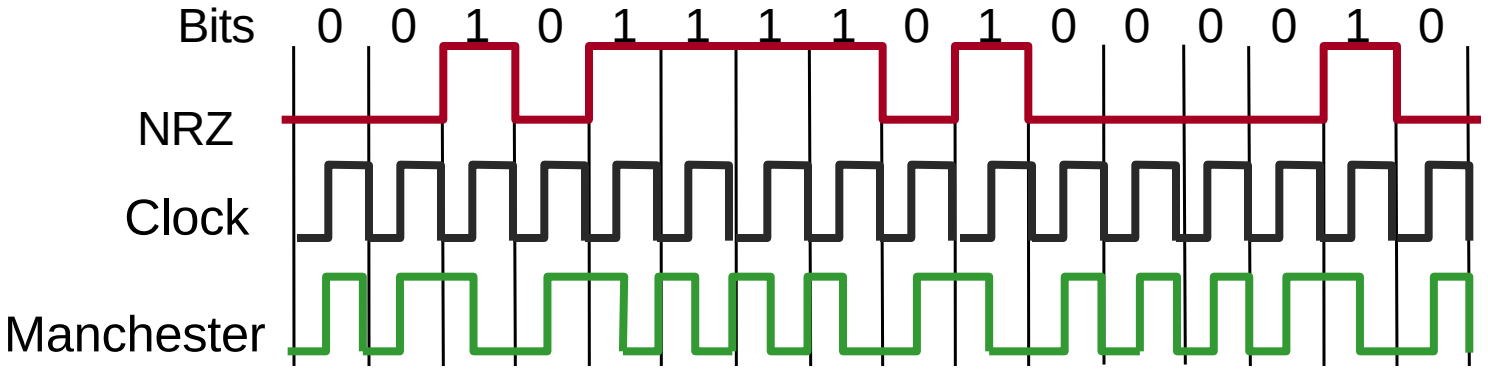
- Signal to Data
  - XOR NRZ data with clock
    - XOR: inputs are the same: 0; else: 1.
    - 0 ➔ a low-to-high transition
      1 ➔ a high-to-low transition.
- Comments
  - Solves clock recovery problem
  - Solves long strings of 0s and 1s.

Remember NRZ encode 0 with low signal and 1 with high signal (no dependence on the current value of signal as NRZI)



Manchester Encoding

- Signal to Data
  - XOR NRZ data with clock
    - XOR: inputs are the same: 0; else: 1.
- Comments
  - Only 50% efficient (two transitions carry one bit)
  - It doubles the rate at which signal transitions are made on the link (bit rate is half the signal transition rate)



4B/5B

- Solves inefficiency in Manchester code
- Insert extra bits to break long 0s or 1s
- Signal to Data
  - Encode every 4 consecutive bits as a 5-bit code
- Symbols
  - Never more than 3 consecutive 0s
    - At most 1 leading 0
    - At most 2 trailing 0s
    - When sent back to back: No more than 3 consecutive 0s in the 5-bit code

4-bit symbol	5-bit Code
0000	11110
0001	01001
…	…
1000	10010
…	…
1111	11101

4B/5B

- The resulting 5-bit codes are transmitted with NRZI
  - Already solves the consecutive 1s
- Comments
  - 80% efficient

4-Bit Data Symbol	5-Bit Code
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Outline

- Encoding
- **Framing**
- Error detection
- Reliable transmission
- Multiple access media (MAC examples)

Framing

- Framing demarcates units of transfer
  - (data link layer) separates continuous stream of bits into frames
  - Marks start and end of each frame
  - i.e. makes a stream of bits into a larger aggregate called a frame
- Objective
  - Efficient error detection
    - Per-frame error checking and recovery
- Challenge
  - How can we determine exactly what set of bits constitute a frame?
  - How do we determine the beginning and end of a frame?

Framing

- Approaches
  - Sentinel
  - Clock-based
  - Length-based
- Characteristics
  - Bit- or byte-oriented (characters)
  - Fixed or variable length
  - Data-dependent or data-independent length

Sentinel-Based Framing

- End of Frame
  - Marked with a special byte or bit pattern
    - SYN (synchronization) character (Start of frame)
    - STX (start of text), ETX (end of text)
    - Frame length is data-dependent
  - Challenge
    - Frame marker (ETX) may exist in data
  - “escaping” for solution, DLE (data-link escape)
    - Preceding ETX with a DLE
    - Called “character stuffing”, because extra character is inserted in the data portion
- Examples:

  ARPANET IMP-IMP, HDLC, PPP, IEEE 802.4 (token bus)

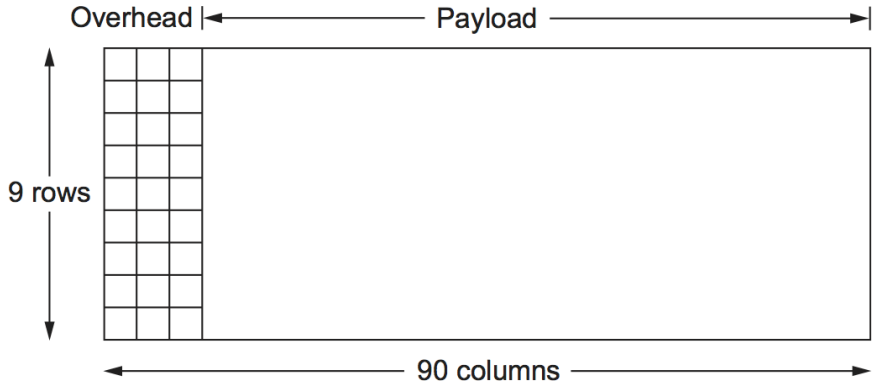
Length-Based Framing

- The length of the frame is included as a field of the header in the frame.
- End of frame
  - Calculated from length sent at start of frame
  - Challenge: Corrupt length markers (“count” field)
  - Called “Framing error”
- Solution:
  - Error detection field

Clock-Based Framing

- Continuous stream of fixed-length frames
- Clocks must remain synchronized
  - Start of frame, but no bit or byte stuffing
  - Rely on clock to determine the start of the next frame.
- Example:
  - Synchronous Optical Network (SONET)
  - STS-1 (OC-1) link: each frame - 125μs long/ 51.84 Mbps
- Problems:
  - Clock synchronization

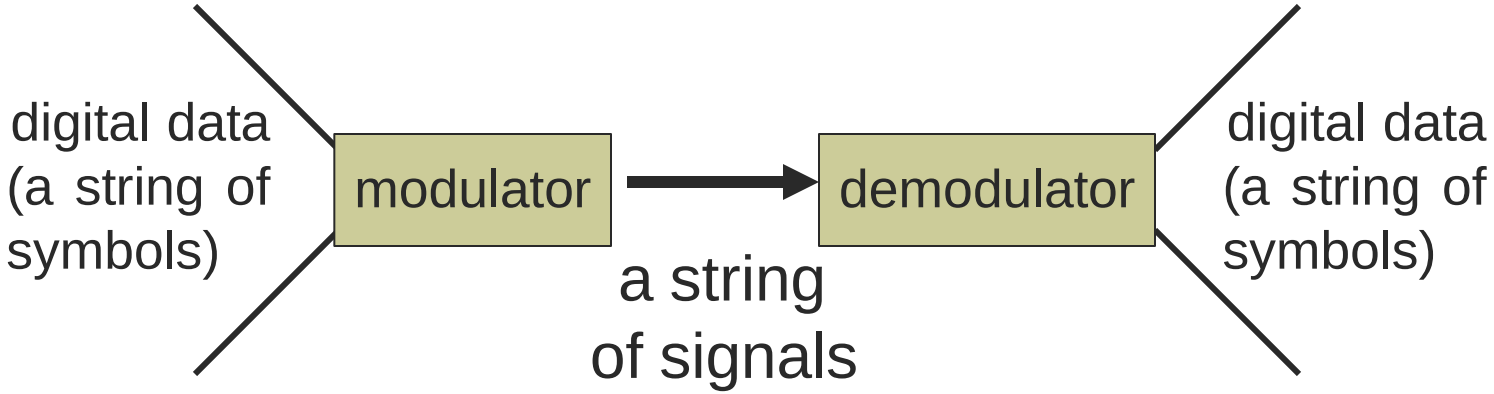
    Payload scrambling



Outline

- Encoding
- Framing
- **Error detection**
- Reliable transmission
- Multiple access media (MAC examples)

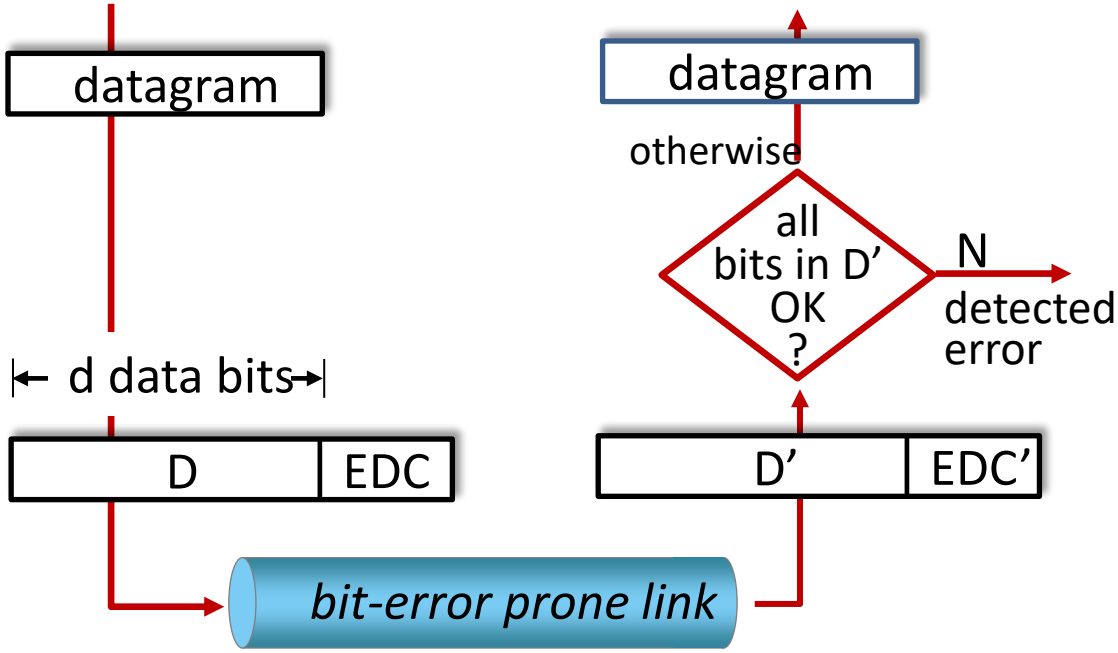
Error Detection (and correction)



- Encoding translates symbols (binary data from source) to signals, that propagates over physical links (manchester, 4B/5B etc)
- Framing demarcates units of transfer
- Error detection validates correctness of each frame

Error Detection

- Idea
  - Add redundant information, *Error detecting codes (EDC)*, that can be used to determine if errors have been introduced, and potentially fix them.
- Errors checked at many levels
  - Within IP layer (IP checksum)
  - Within network adaptor (CRC check)
  - Possibly higher layers and/or within application as well



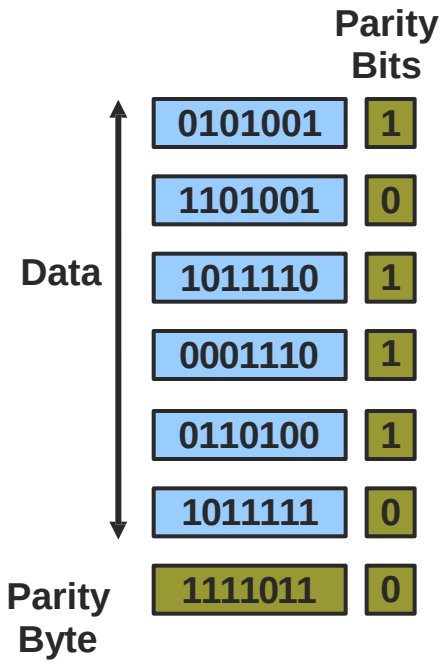
Error Bits or Bursts?

- Common model of errors
  - Probability of error per bit
  - Error in each bit independent of others
  - Value of incorrect bit independent of others
- Burst model
  - Burst: sequence of consecutive errored bits
  - Probability of back-to-back bit errors
  - Error probability dependent on adjacent bits
  - Value of errors may have structure
- Why assume bursts?
  - Appropriate for some media (e.g., radio)

Digital Error Detection Techniques

- Two-dimensional parity
  - Detects up to 3-bit errors
  - Good for burst errors
- IP checksum
  - Simple addition
  - Simple in software
  - Used as backup to CRC
- Cyclic Redundancy Check (CRC)
  - Powerful mathematics
  - Tricky in software, simple in hardware
  - Used in network adapter

Two-Dimensional Parity



- Use 1-dimensional parity
  - Add one bit to a 7-bit code to ensure an even/odd number of 1s
- Add 2nd dimension
  - Add an extra byte to frame
    - Bits are set to ensure even/odd number of 1s in that position across all bytes in frame
- Comments
  - Catches all 1-, 2- and 3-bit and most 4-bit errors

AT Receiver

Can detect *and* correct single bit errors (without retransmission!)

no errors:

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

detected and correctable single-bit error:

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

parity error

parity error

Background: One's Complement System

- The one's complement of a binary number is defined as the value obtained by inverting all the bits in the binary representation of the number (swapping 0s for 1s and vice versa).
  - 0011 → 1100; 0101 → 1010
- A ones' complement system is a system in which negative numbers are represented by the one's complement of their corresponding positive numbers.
  - E.g., **-5** → corresponding positive number: 5 → binary representation of 5: 0101 → one's complement: **1010**.

Internet Checksum

- Idea
  - Add up all the words and get the sum
  - Transmit the sum, called checksum
  - Receiver do the same calculation, and compare the checksum
- Internet Checksum
  - Use 1's complement addition on 16bit codewords
  - Example
    - Codewords: -5 -3
    - 1's complement binary: 0101->1010 0011->1100
      - 1010+1100= 10110 => 0110(ignore carry), then increment-> 0111
    - 1's complement sum (-8)=>0111
      - (ones complement of -8) , is 8=> 1000 => 0111
- Comments
  - Small number of redundant bits
  - Easy to implement
  - Not very robust (consider one increase by 1, one decrease by 1)

Cyclic Redundancy Check (CRC)

- Goal
  - Maximize protection, Minimize extra bits
- Idea
  - Add k bits of redundant data to an n-bit message
  - N-bit message is represented as a (n-1)-degree polynomial with each bit in the message being the corresponding coefficient in the polynomial
  - Example
    - Message = 10011010
    - Polynomial

      $M(x) = 1 * x^7 + 0 * x^6 + 0 * x^5 + 1 * x^4 + 1 * x^3 + 0 * x^2 + 1 * x + 0$

      $= x^7 + x^4 + x^3 + x$

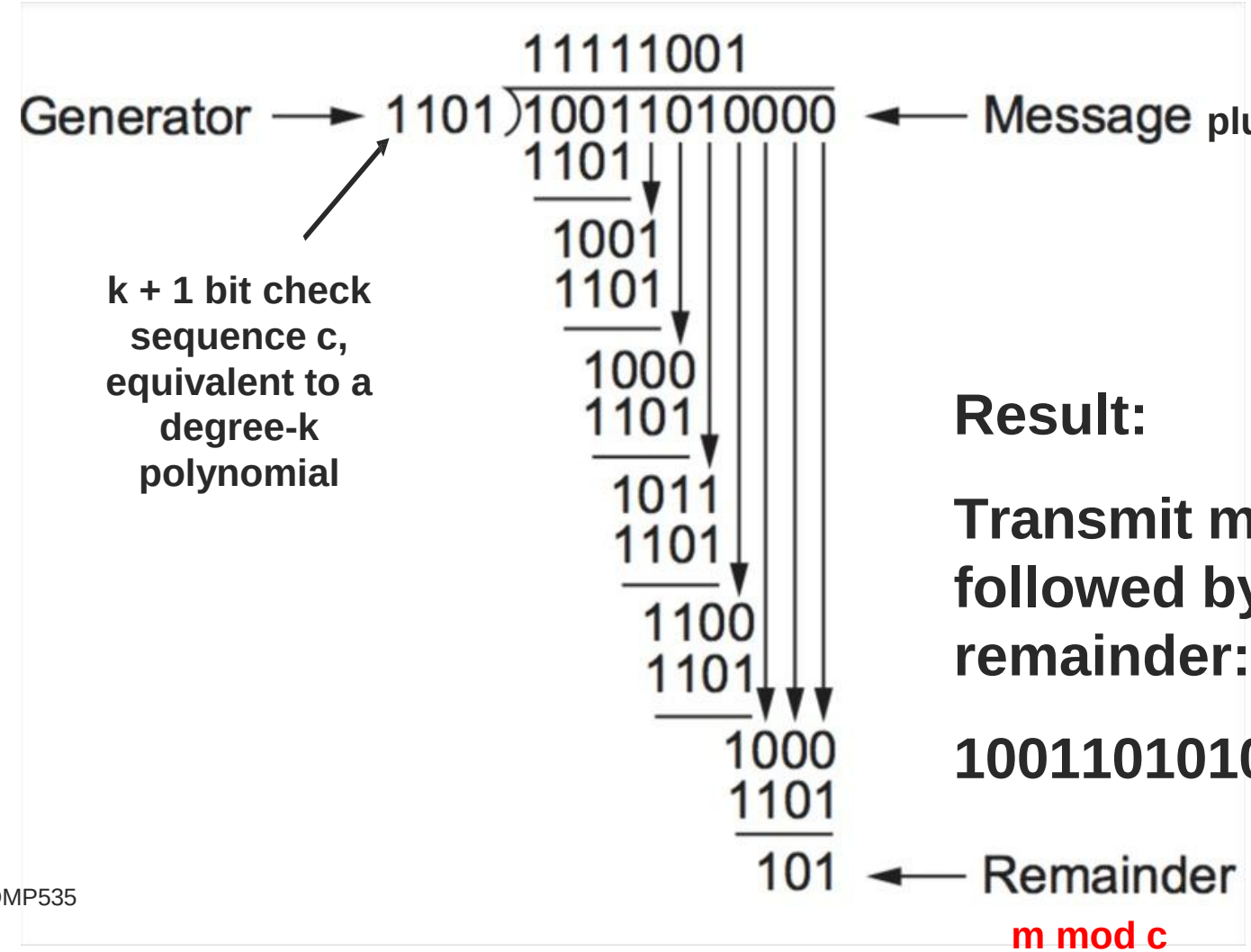
CRC

- Select a divisor polynomial C(x) with degree k
  - Example with k = 3:
    - $C(x) = x^3 + x^2 + 1$
- Transmit a polynomial P(x) that is evenly divisible by C(x)
  - P(x) = M(x) + k bits
  - How to get these k bits to make P(x) divisible by C(x)?

CRC - Sender

- Steps
  - T(x) = Multiply M(x) by x^k (zero extending)
  - Find remainder, R(x), from T(x)/C(x)
  - P(x) = T(x) – R(x) $\Rightarrow$ M(x) followed by R(x)
- Example
  - M(x) = 10011010 = $x^7 + x^4 + x^3 + x$
  - C(x) = 1101 = $x^3 + x^2 + 1$
  - T(x) = 10011010000 (K=3)
  - R(x) = 101
  - P(x) = 10011010101
  - (The minus operation in polynomial arithmetic is the logical XOR operation, so we actually send 10011010101.)

Calculation using polynomial long division



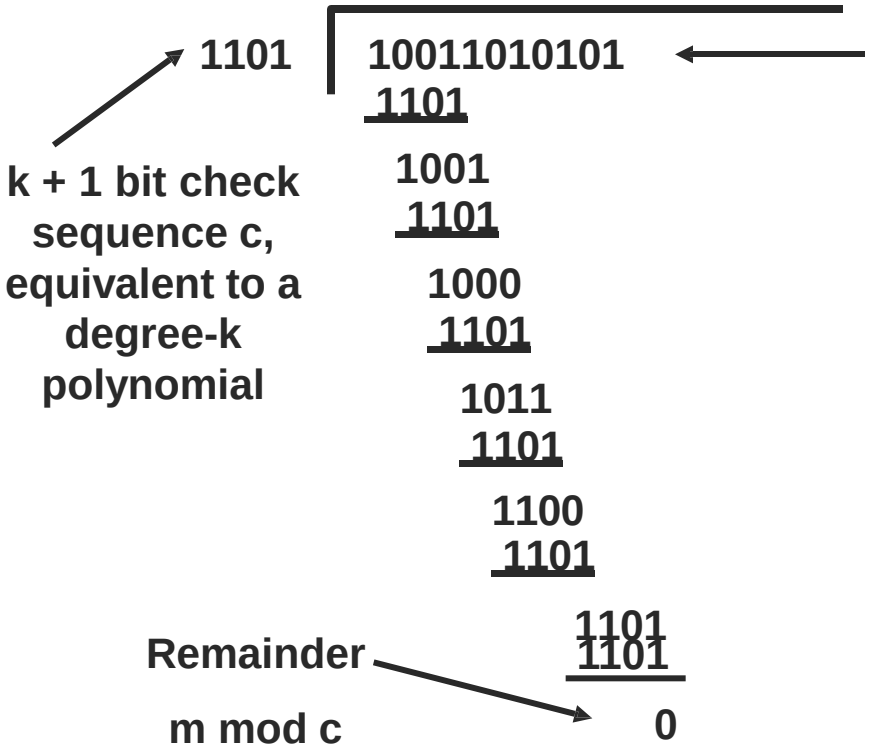
CRC - Receiver

- Receive Polynomial P(x) + E(x)
  - E(x) represents errors
  - E(x) = 0, implies no errors
- Divide (P(x) + E(x)) by C(x)
  - If result = 0, either
    - No errors (E(x) = 0, and P(x) is evenly divisible by C(x))
    - (P(x) + E(x)) is exactly divisible by C(x), error will not be detected

CRC – Example Decoding – No Errors

C(x) = $x^3 + x^2 + 1$ = 1101 Generator

P(x) = $x^{10} + x^7 + x^6 + x^4 + x^2 + 1$ = 10011010101 Received Message



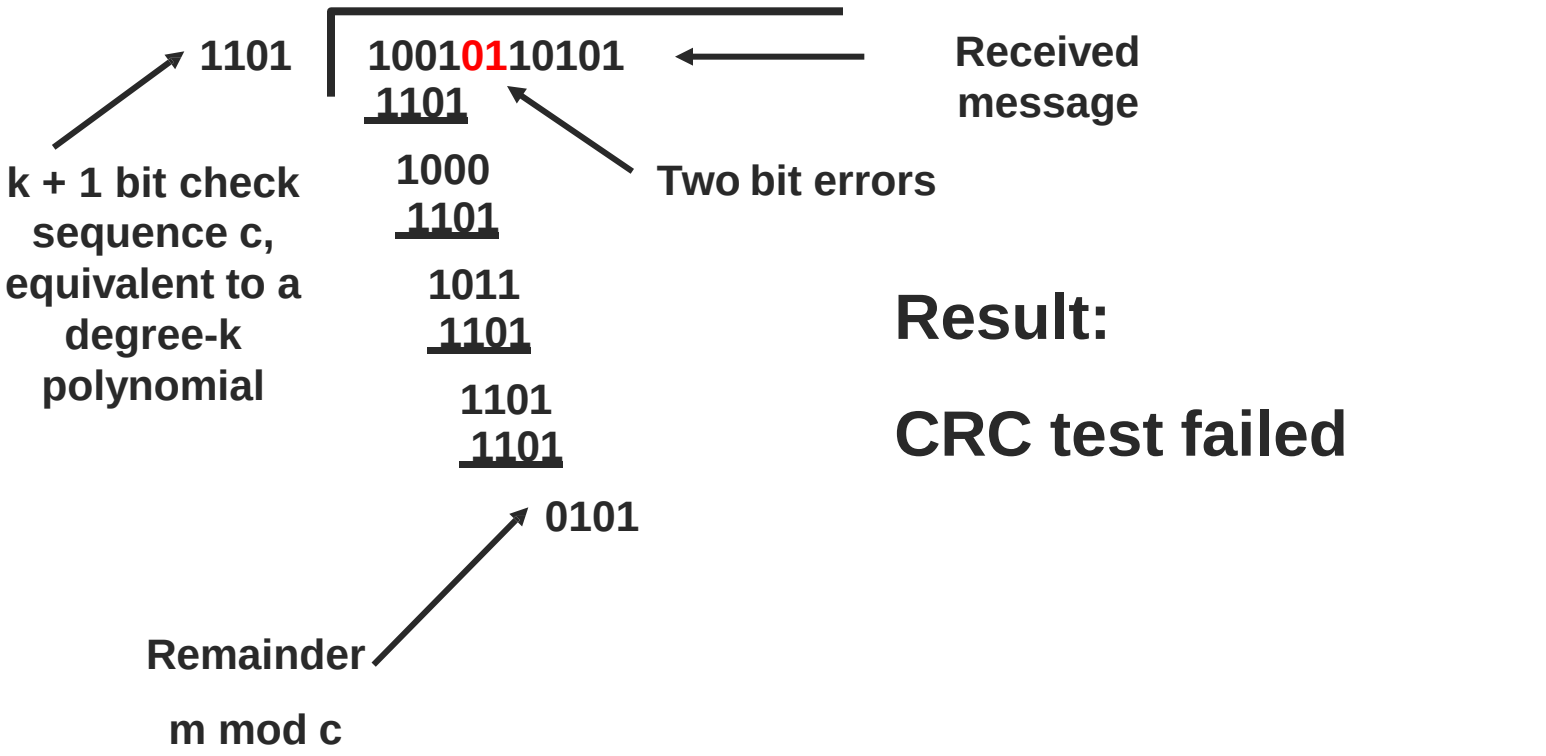
Result:

CRC test is passed

CRC – Example Decoding – with Errors

C(x) = $x^3 + x^2 + 1$ = 1101 Generator

P(x) = $x^{10} + x^7 + x^5 + x^4 + x^2 + 1$ = 10010110101 Received Message



Result:

CRC test failed

Common CRC Generators

CRC-8	$x^8 + x^2 + x^1 + 1$
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$

CRC Error Detection

- Properties
  - Characterize error as E(x)
  - Error detected unless C(x) divides E(x)
    - (*i.e.*, E(x) is a multiple of C(x))
- How to choose C(x)?, or What errors can we detect?
  - All single-bit errors, if x^k and x^0 have non-zero coefficients
    - E(x)= x^i , when it affects bit position i
    - Two-term polynomial that cannot divide evenly into the one term
  - All double-bit errors, if C(x) has at least three terms
  - Any “burst” error (i.e., sequence of consecutive errored bits) for which the length of the burst is less than k bits (Most burst errors of length greater than k bits can also be detected.)