Chapter 5: Link Layer and LAN

Error Detection and Correction

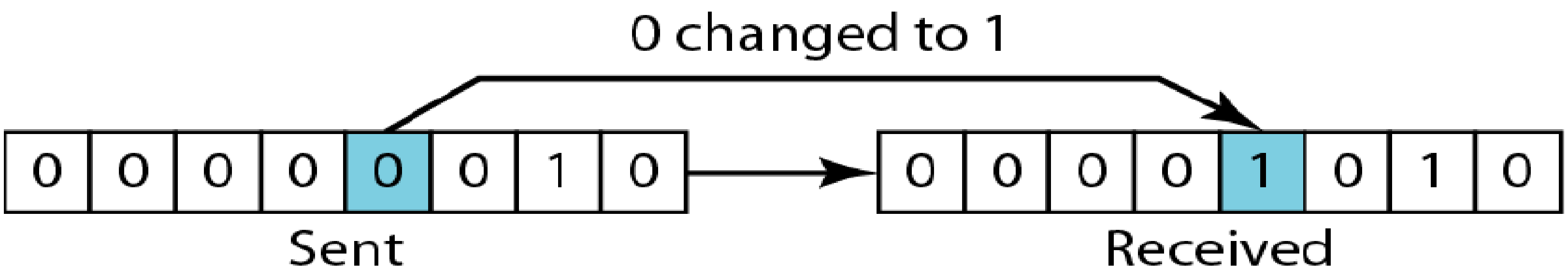
Instructor: HOU, Fen

2025

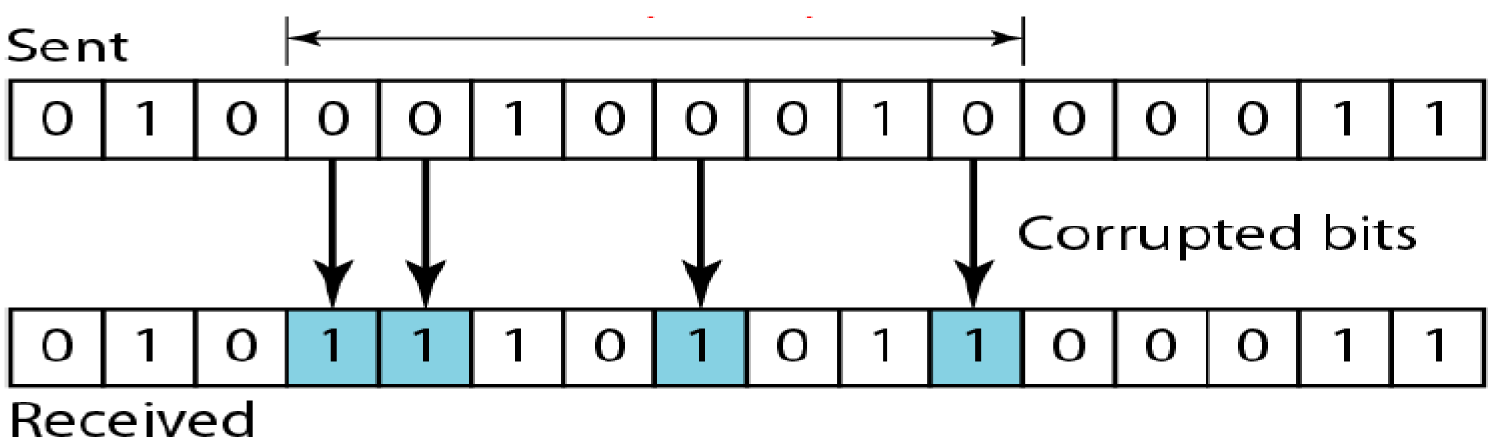
Error Detection and Error Correction Technologies

Transmission Error

- Single-bit error: only one bit in the data unit has been changed.



- Multiple bit error (a burst error): two or more bits in the data unit have been changed.



Error Detection

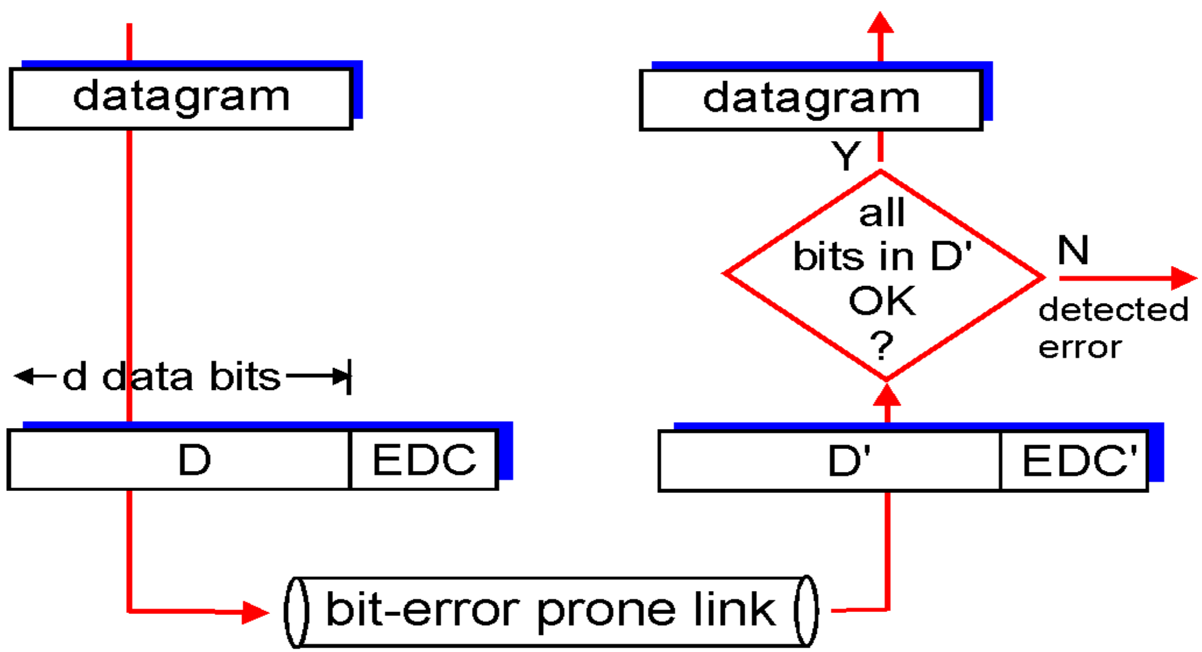
To detect or correct errors, we need to send extra (redundant) bits with the original data.

EDC= Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

Error detection not 100% reliable!

- protocol may miss some errors
- larger EDC field yields better detection and correction



Error Detection and Correction Technique

- Two error detection techniques
  - Parity checking
  - Checksum
  - CRC (Cyclic Redundancy Check)
- Forward error correction (FEC)
  - FEC denotes the ability of the receiver to both detect and correct error.
  - FEC technique allows for immediate correction of errors at the receiver, which avoids the delay needed for the sender to receive a NAK and the following retransmission.

Parity Check

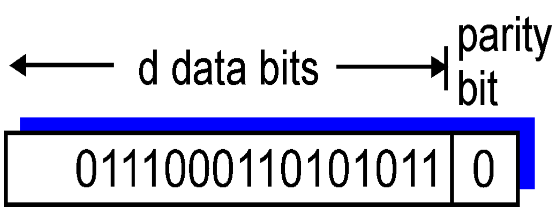
Single Bit Parity:

Detect single bit errors

Two Schemes:

- Even parity check : by adding extra bit called as parity bit to make the total number of "1" in the formed data (original data + parity bit) to be an even number.

- Odd parity check: by adding extra bit called as parity bit to make the total number of "1" in the formed data (original data + parity bit) to be an odd number.



Parity Check

Example:

- The original data unit is 1 0 0 1 0 0 1. Even parity check is used, the parity bit should be
  A. 1
  B. 0

- Answer: A; and the transmitted data unit is: 1 0 0 1 0 0 1 1

Parity Check

Example:

- What would the parity bit be for each of the following data unit. We assume that odd parity check is used
  - (1). 1 0 1 1 0 0 0
  - (2). 1 1 1 0 1 0 1
  - (3). 1 0 1 0 0 1 1
  - (4). 1 0 1 1 0 0 0

Limitation of Parity Check

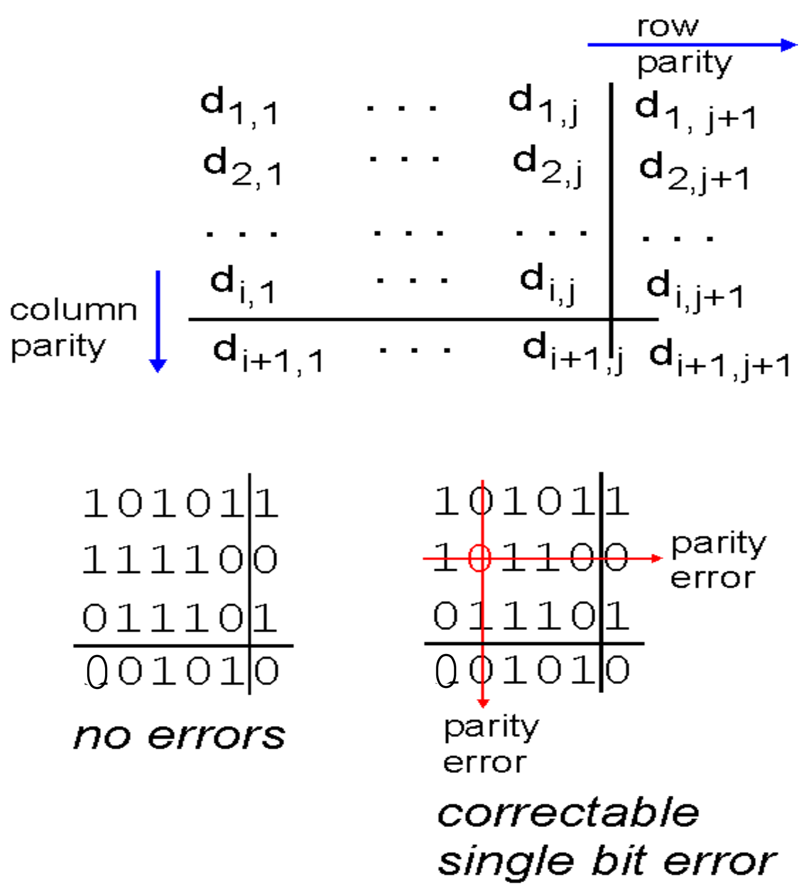
- This technique cannot detect an even number of bit errors (two, four, etc.).
- This technique cannot determine the location of the error bit.

- Example:
  - Consider the data unit to be transmitted is 10010001 and even parity is used.
  - Then, code word transmitted to the receiver = 100100011
  - Consider during transmission, code word is modifies as 101100111. (2 bits flip)
  - On receiving the modified data, the receiver finds the number of 1 is even and even parity checking is used.
  - So, the receiver assumes that no error occurred in the data during transmission although the data is corrupted.

Parity Check

Two Dimensional Bit Parity:

Detect *and correct* single bit errors



Parity Check

Example:

- Calculate the two-dimensional parity check for the data as follows. We assume even parity checking is used.

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

Parity Check

Example:

- Calculate the two-dimensional parity check for the following data. (Assume the two-dimensional parity check unit is 8 bits long. You will need to calculate the simple parity check bits as well as the parity check unit)

101101011100000000111

1 0 1 1 0 1 0

1 1 1 0 0 0 0

0 0 0 0 1 1 1

Parity Check

Exercise

- Suppose the information content of a packet is the bit pattern 1110 1011 1001 1101 and two-dimensional even parity check is being used. What would the value of the field containing the parity bits?
- Answer:

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

Parity Check

Exercise

- Suppose we begin with the initial two-dimensional parity matrix:

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

- When a single bit error

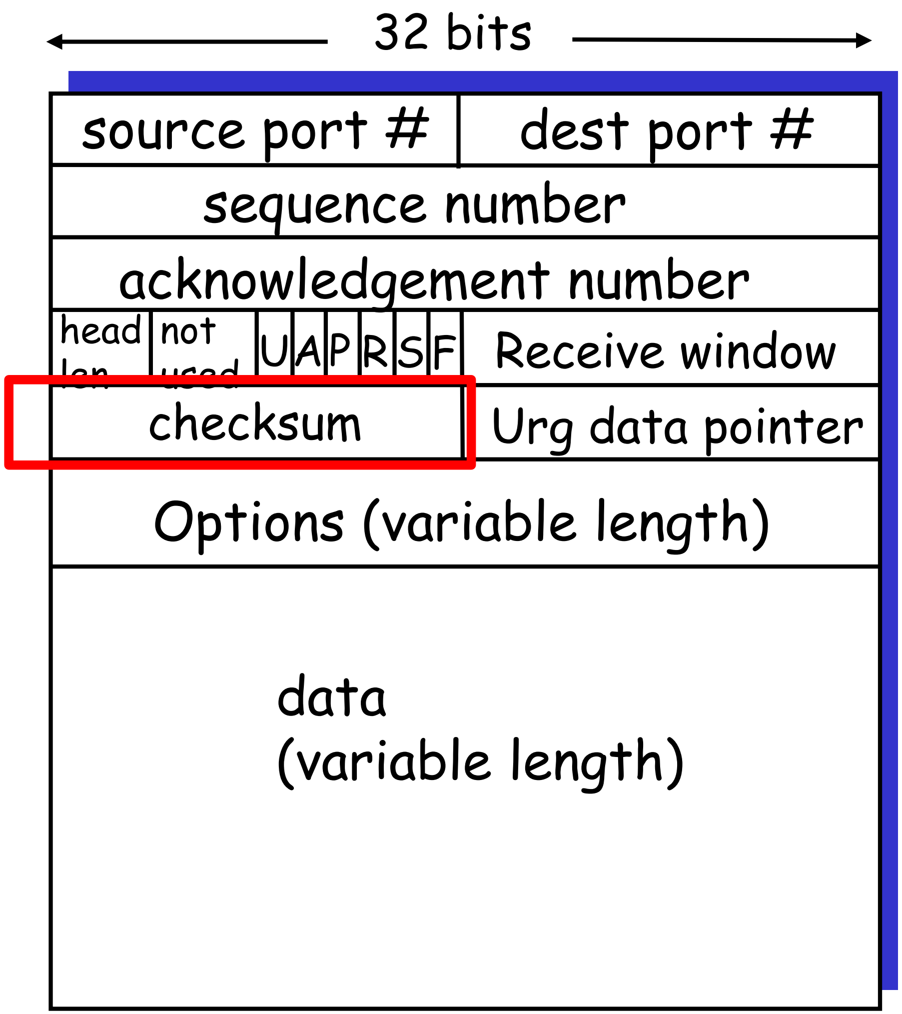
0	0	0	0
1	1	0	1
0	1	0	1
1	0	1	0

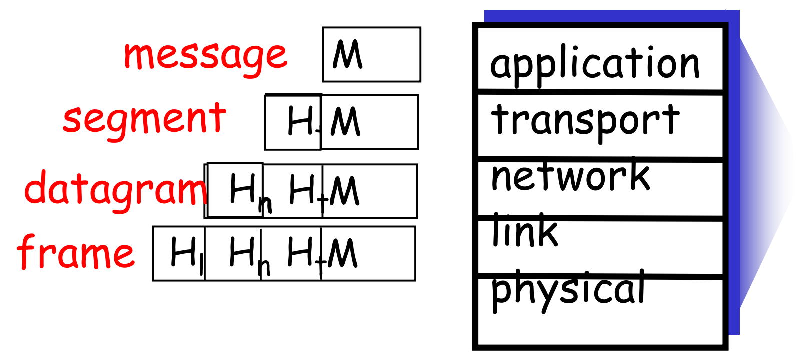
- When a double-bit error

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

Checksum

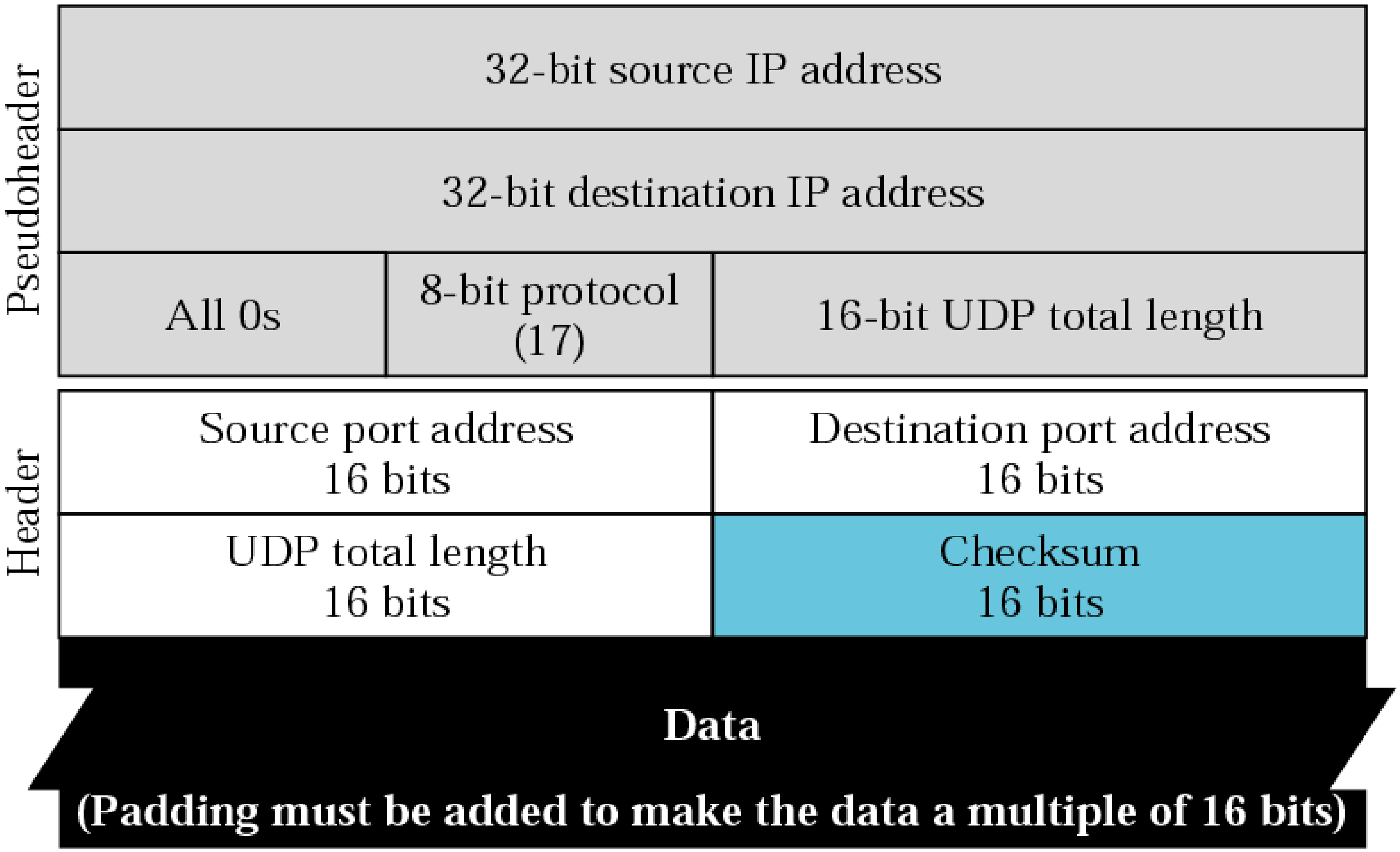
Checksum in TCP segment structure





Checksum in UDP Datagram Structure

- The checksum calculation includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.



Binary Addition

- You can add two binary numbers one column at a time starting from the right, just as you add two decimal numbers
- But remember that it's binary. For example, 1 + 1 = 10 and you have to carry!
- what we really need to do is add *three* bits: the augend and addend, and the *carry in* from the right.

The initial carry in is implicitly 0

	1	1	1	0		Carry in	（进位）
		1	0	1	1	Augend	（被加数）
+		1	1	1	0	Addend	（加数）
	1	1	0	0	1	Sum	

most significant bit, or MSB

least significant bit, or LSB

2' s Complement Sum and 1 's Complement Sum

- 2's complement sum is done by summing the numbers and discarding the carry.
- 1's complement sum is done by summing the numbers and adding the carry to the result。

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
2's complement sum		1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
1's complement sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0

Checksum

Goal---Error detection: detect "errors" (e.g., flipped bits) in transmitted data

How to implement: treat data to be protected as a sequence of k-bit integers and use the resulting sum as the error-detection bits

Sender:

- treat data to be protected as sequence of k-bit words
- checksum: make 1's complement sum over all of the k-bit words, and then for the sum, convert all 0s to 1s, and all 1s to 0s. The result is the checksum
- sender puts checksum value into the checksum field

Receiver:

- treat data as sequence of k-bit words
- Make 1's complement sum over all the k-bit words and the checksum.
- check if computed result is all 1s:
  - YES - no error detected
  - NO - error detected

Checksum

How to implement: treat data to be protected as a sequence of segments of k-bit and use the resulting sum as the error-detection bits

Sender:

- treat data to be protected as sequence of k-bit words
- checksum: make 1's complement sum over all of the k-bit words, and then for the sum, convert all 0s to 1s, and all 1s to 0s. The result is the checksum
- sender puts checksum value into checksum field

Receiver:

- treat data as sequence of k-bit words
- Make 1's complement sum over all the k-bit words and the checksum.
- check if computed result is all 1s:
  - YES - no error detected
  - NO - error detected

Checksum

- Note: When doing 1's complement sum, a carryout from the most significant bit needs to be added to the result
- Example: add two 16-bit integers
- Weak error protection. Easy to implement in software.

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
1's complement sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
Checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Example

- Checksum value of 1001001110010011 and 1001100001001101 of 16 bit segment is
  - A. 1010101000011111
  - B. 1011111000100101
  - C. 1101010000011110
  - D. 1101010000111111

- Answer: C

Example

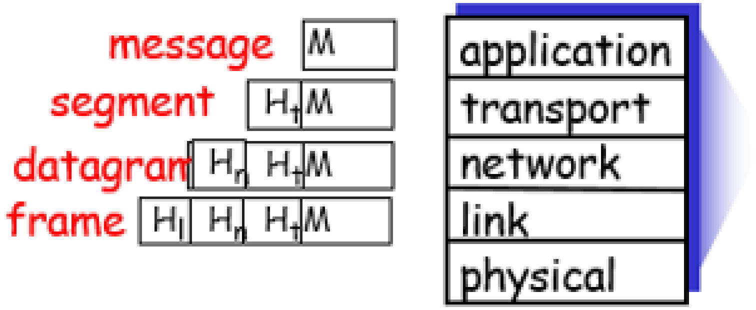
- Consider 8 bit checksum is used at the sender side, the data unit to be transmitted is : 10011001111000100010010010000100
- Please calculate the checksum.
- Answer: 11011010
  - All the segments are added and the result is

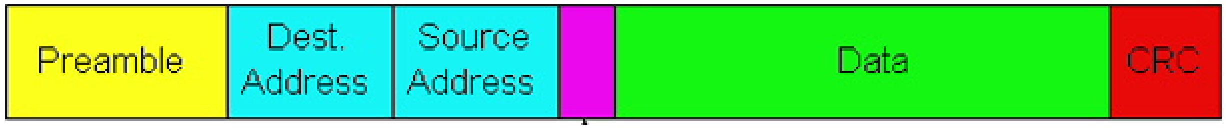
  10011001 + 11100010 + 00100100 + 10000100 = 1000100011

  - 1's complement sum: since the result consists of 10 as carry in bits, so these extra 2 bits are wrapped around. The result of 1's complement sum is 00100101.
  - Checksum is obtained as 11011010

Ethernet Frame Structure

- Frame and MAC Address: 6 byte MAC address



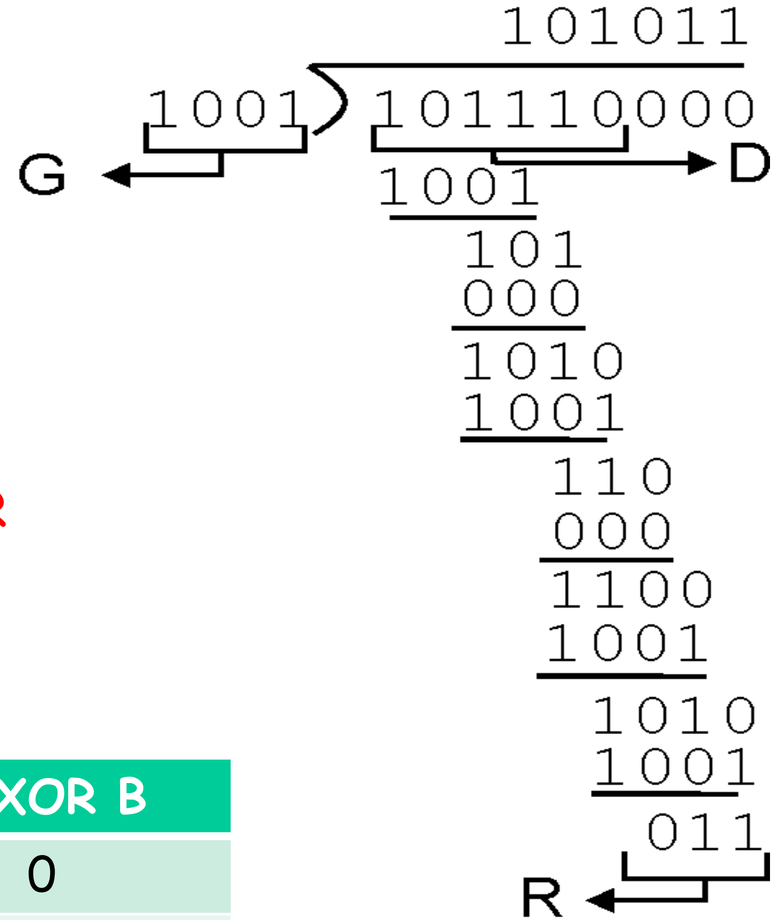


Cyclic Redundancy Check (CRC)

- CRC is an error detection technique widely used in computer networks.
- In CRC method, a certain number of check bits (called as CRC bits) are appended to the message being transmitted.
- CRC codes are also known as polynomial codes
  - Treat bit strings as representations of polynomials with coefficients 1's and 0's
  - A (n+1)-bit message can be represented as a polynomial of degree n, where the exponents are the order of the bits that are sent. For example
    - X = 10011010 is represented by a polynomial
    - $M(X) = x^7+x^4+x^3+x^1$
  - So, a sender and receiver can be considered to exchange polyns.
- A message is sent with a calculated CRC code such that it may be verified by receiver

Cyclic Redundancy Check (CRC)

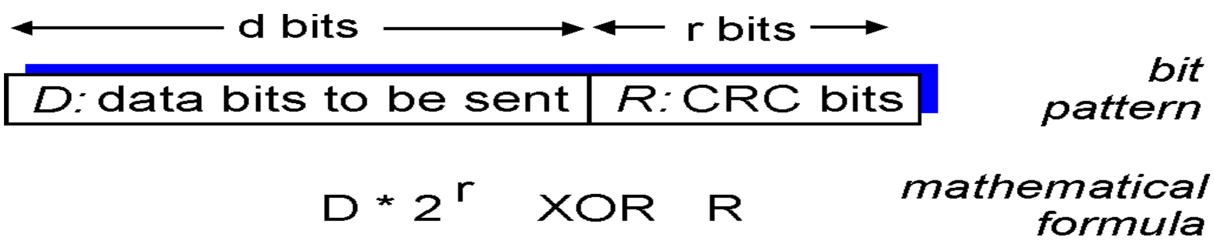
- CRC calculation is realized by a division operation applied to binary numbers.
- There is a difference in the division: subtraction operations are replaced by eXclusive OR logical operations (XOR)
- "XOR Divide" means use bitwise XOR instead of subtraction.
- Remembering XOR operation



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Cyclic Redundancy Check (CRC)

- A transmitter wishes to send a message D with d data bits (called as dataword)
- A CRC code R with r bits of length must be generated and appended to dataword before it is sent. The transmitted data is called as codeword.



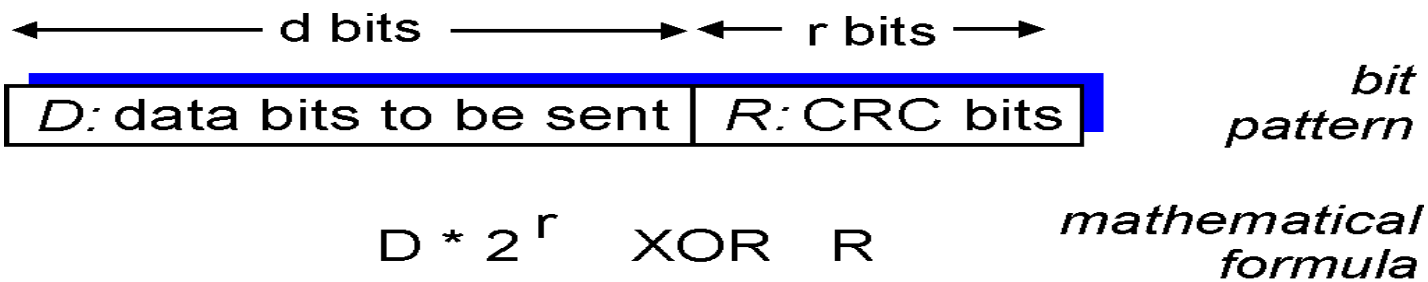
- CRC "Generator" G has (r+1) bits of length. The first and last bits are 1's, the others are mixed o's and 1's.
- G is known by both transmitter and receiver. It is a basis of CRC calculation.

Cyclic Redundancy Check (CRC)

- CRC code calculation is based on dataword D of d-bit length and generator G of (r+1)-bit length. The basis for CRC code calculation is the formula:

  R=remainder of $D*2^r/G$

- $D*2^r$ is the appending of r bits 0 to data bits. For example D=$(100101)_2$, and r=3, therefore, $D*2^r=(100101000)_2$
- The reminder R is the r-bit CRC code.
- Sender side: send the codeword which is composed of d-bit dataword D with r-bit CRC code appended, i.e., $D*2^r$ XOR R
- Receiver side: calculating a division for the total received data (including CRC code) by the generator G. If zero reminder, no error ! if non-zero remainder: error detected!



Cyclic Redundancy Check (CRC): Example

- Transmitter wants to send dataword D=101110
- Generator G are the (r+1) bits G=1001
- Transmitter side:
  - Calculate CRC code R (r-bits) using the formula

    the remainder of $D*2^r/G$
  - Send the codeward which is composed of data bits with CRC code appended
- Receiver side:
  - Verify the correct transmission by calculating a division for the total received data (including CRC code) by the generator.
    - zero remainder: No error!
    - non-zero remainder: error detected!

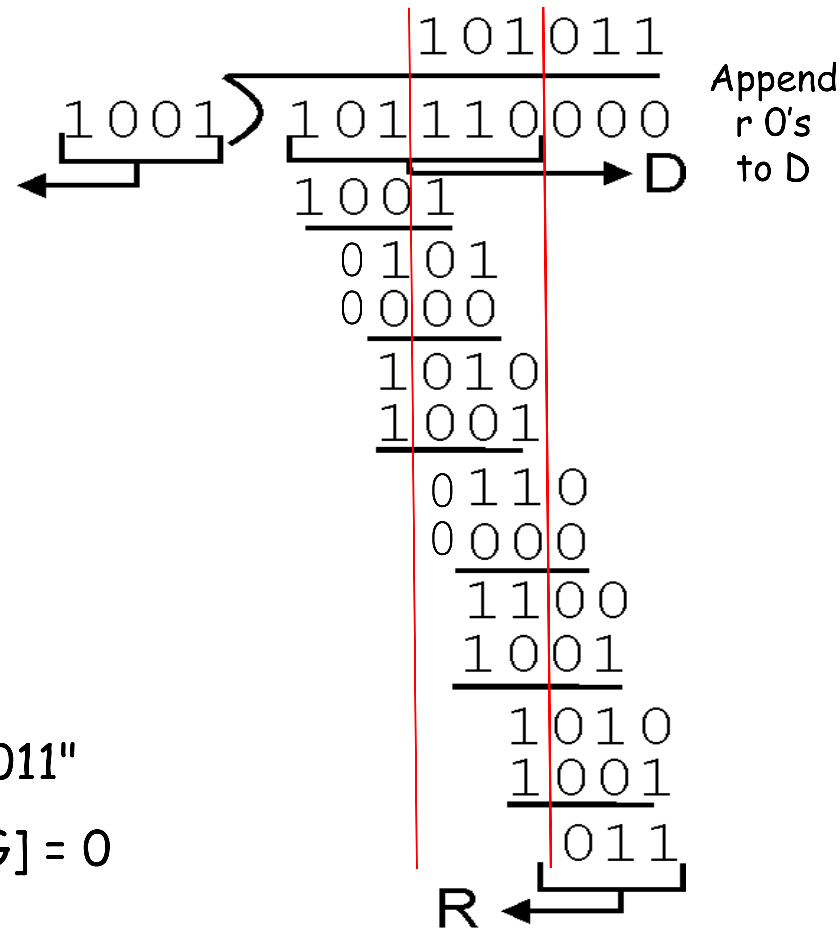
CRC Example (r-bit CRC)

CRC "Generator" G has r+1 bits. The first and last are 1's, the others are mixed o'0 and 1's.

"XOR Divide" the data $D*2^r$(r 0's added to right) by G until there is a 3-bit remainder R.

"XOR Divide" means use bitwise XORing instead of subtraction.

The "CRC" = R, may have leading 0's. Keep them.



Transmit: $D*2^r$ XOR R e.g., "101110 011"

At receiver: remainder[($D*2^r$ XOR R)/G] = 0

CRC Example (r-bit CRC)

"XOR Divide" means use bitwise XORing instead of subtraction.

The "CRC" = R, may have leading 0's. Keep them.

Transmit: $D*2^r$ XOR R   e.g., "101110 011"

At receiver: remainder[($D*2^r$ XOR R)/G] = 0

```
          1 0 1 0 1 1
        _____________________
1 0 0 1 ) 1 0 1 1 1 0 0 1 1
          1 0 0 1
            0 1 0 1
            0 0 0 0
              1 0 1 0
              1 0 0 1
                0 1 1 0
                0 0 0 0
                  1 1 0 1
                  1 0 0 1
                    1 0 0 1
                    1 0 0 1
```

Cyclic Redundancy Check (CRC)

Exercise

- Consider the 4-bit generator, G=1001, and suppose that dataword D has the value 10101010.

  (a)What is the value of R?

  (b)What is the transmitting codeword at the sending side

- Answer:
  - We divide 1001 into 10101010 000, we get, with 10111101, and the remainder of R=101.
  - The transmitting data at the sending side is 10101010101

Cyclic Redundancy Check (CRC): Example

- Transmitter wants to send data bits D=111100101
- Generator G are the (r+1) bits G=101101.
- Transmitter side:
  - Calculate CRC code R (r-bits) using the formula

    the remainder of $D*2^r/G$
  - Send data bits with CRC code appended
- Receiver side:
  - Verify the correct transmission by calculating a division for the total received data (including CRC code) by the generator.
    - zero remainder: No error!
    - non-zero remainder: error detected!

Ethernet Frame Structure

- Addresses: 6 byte MAC address
  - if an adapter receives a frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to network-layer protocol
  - otherwise, the adapter discards the frame