Data Link Layer (2)

COMP90007 Internet Technologies

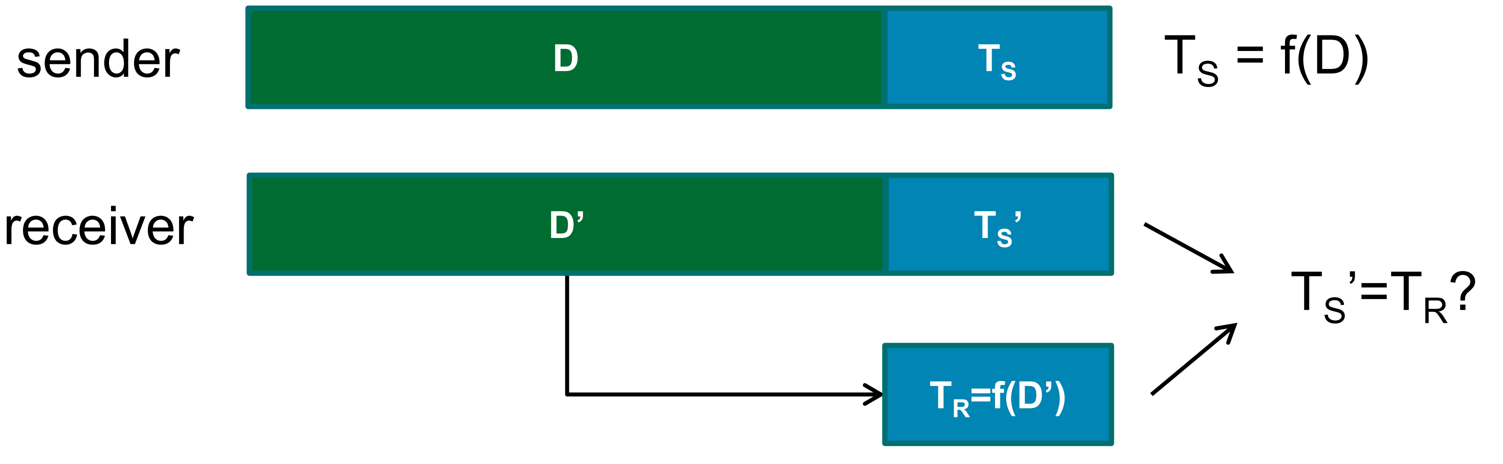
Lecturer: Ling Luo

Semester 2, 2024

Framing (1)

- Framing: breaks raw bit stream into discrete units

- Primary purpose: provide some level of reliability over the unreliable physical layer
- Example: checksums



Framing (2)

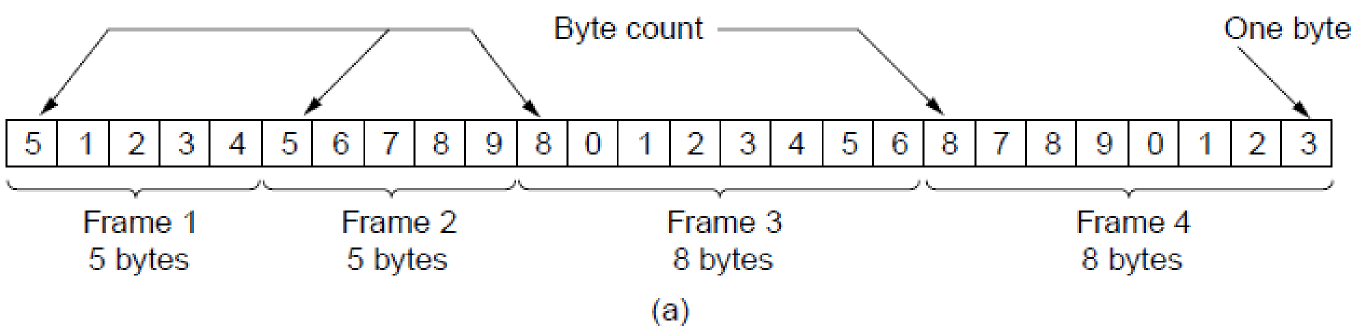
- Methods:
  - Character (Byte) count
  - Flag bytes with byte stuffing
  - Start and end flags with bit stuffing

- Most data link protocols use a combination of character count and one other method

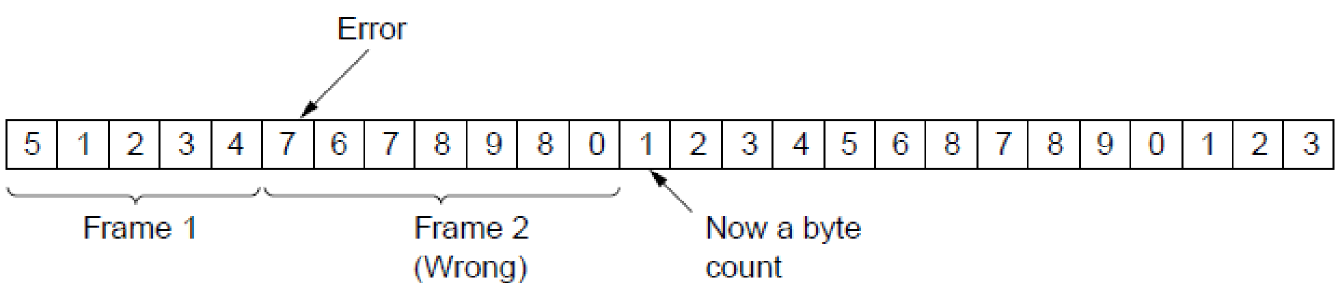
Character Count

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

No error

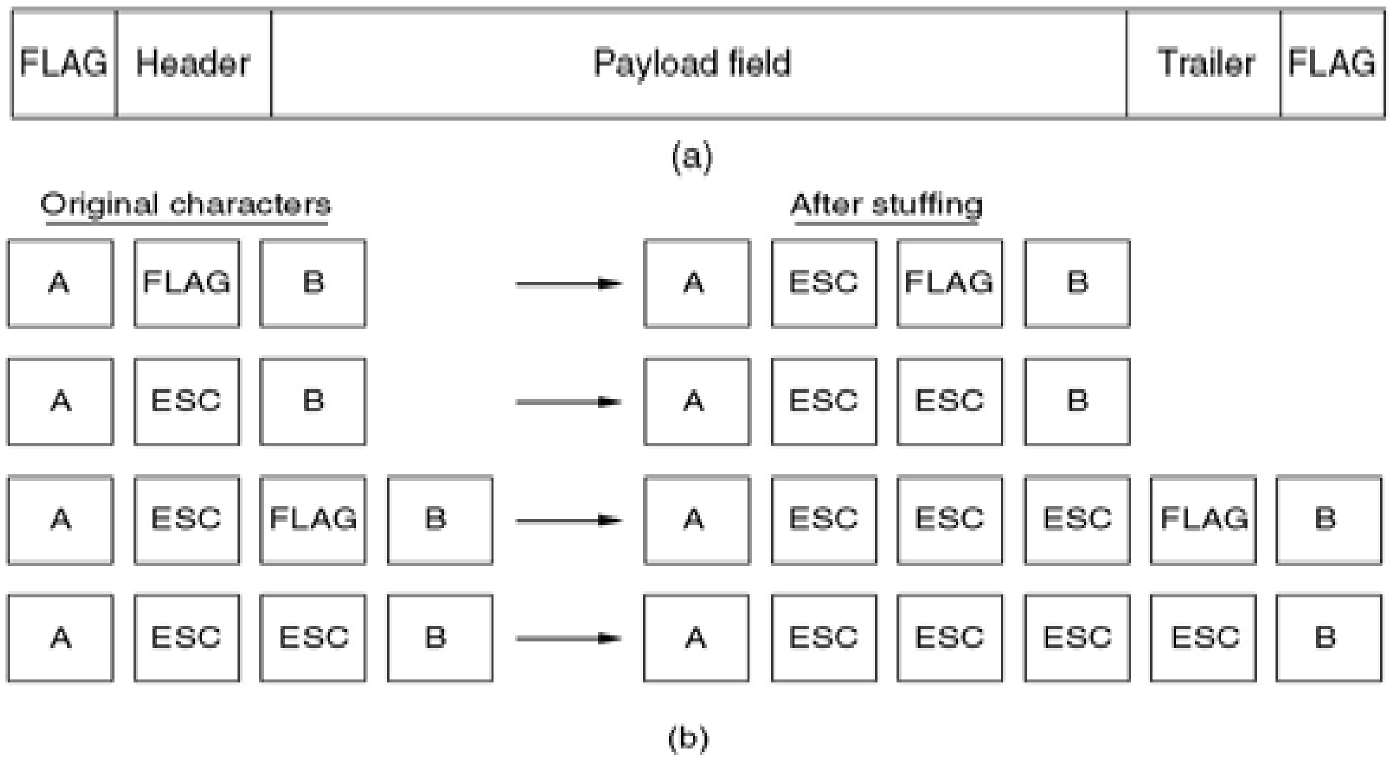


Case with error



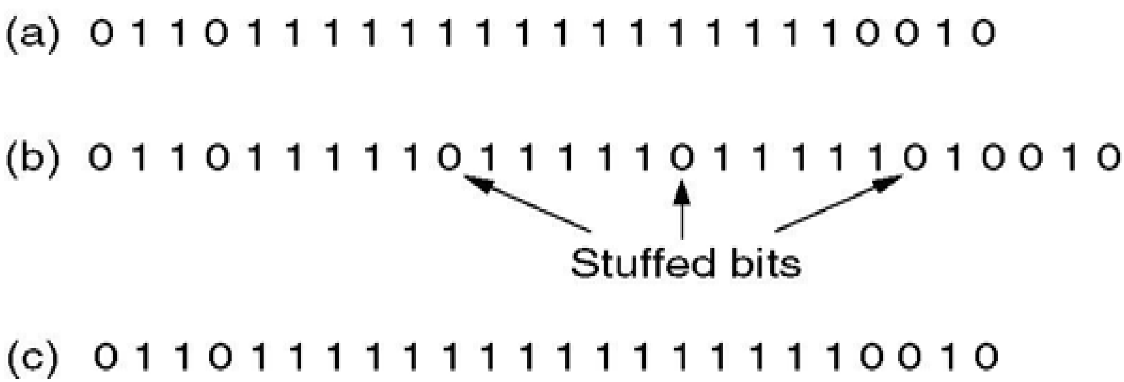
Flag Bytes with Byte Stuffing

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



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

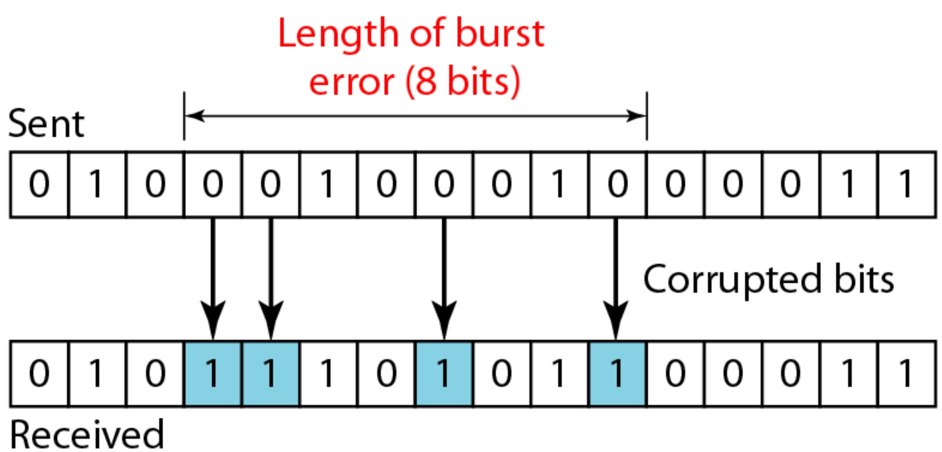
Insert 0 after five ones (11111)

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
  - **Detecting** the error, and **retransmitting**
  - **Correcting** the error

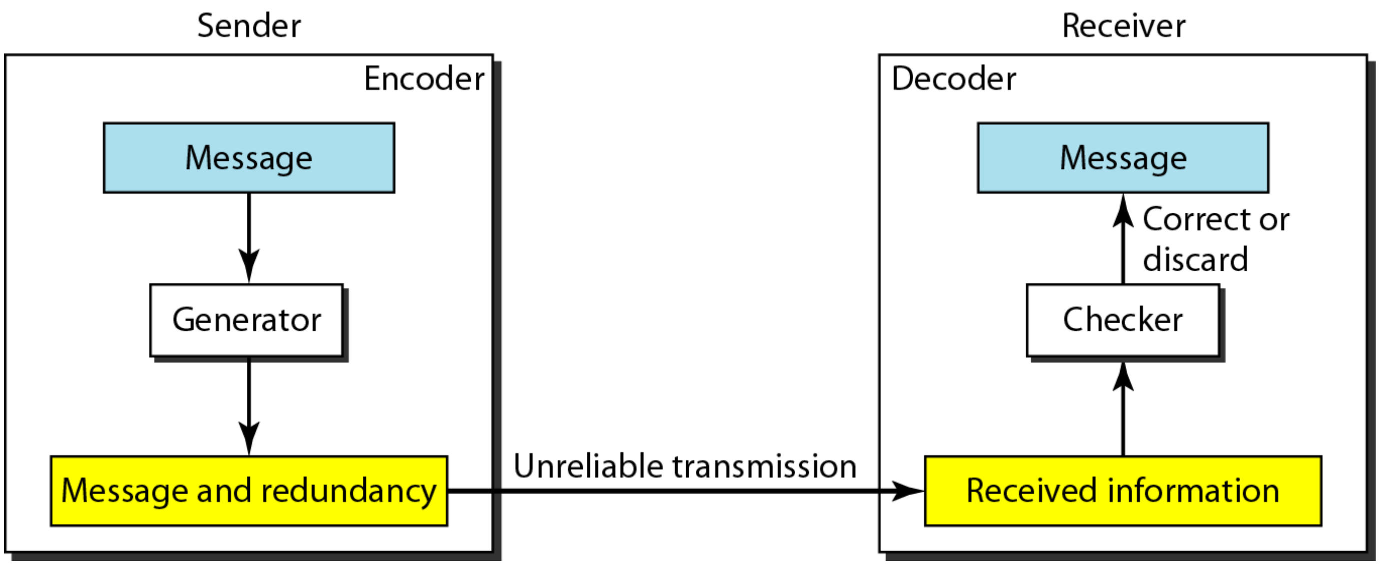
Error Detection and Correction (1)

- Physical media may be subject to errors, which may occur randomly or in bursts
  - Single-bit error
  - Burst error: two or more bits have changed. Easier to detect but harder to resolve



Error Detection and Correction (2)

- Resolution needs to occur before handing data to network layer



- Desirable features
  - **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 from 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?

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 incorrect.

Error Control Methods

- Error Detection
  - Parity Bit
  - Internet Checksum
  - Cyclic Redundancy Check (CRC)
- Error Correction
  - Hamming Code

Parity Bit

- Given data 10001110, count the number of 1s

  Sender: Add parity bit → 10001110**0** (for even parity)

  10001110**1** (for odd parity)

  Receiver: Check the received data for errors.

- Hamming distance = 2
  - 2 – 1 = **1 bit error** can be **detected**
  - (2 – 1) / 2 = ½ not even 1 bit error can be corrected

Internet Checksum

- Checksum: a group of check bits for a message
- There are different variations of checksum

- Internet Checksum (16-bit word):

  Sum modulo 2^{16} and add any overflow of high order bits back into low-order bits

Example of Checksum

Calculate checksum (5-bit word) for data

00110 10001 11001 01011

1. 00110
   +10001 = $\boxed{10111}$

2. 11001
   +10111 = 10000 +1 = $\boxed{10001}$

3. 01011
   +10001 = $\boxed{11100}$

4. The checksum is *one's complement* of 11100 which is $\boxed{00011}$

Data sent: 00110 10001 11001 01011 **00011**

Cyclic Redundancy Check

- Based on a generator polynomial G(x)
  - e.g. $G(x) = x^4+x+1$ (10011)
  - Steps:
    - Let r be the degree of $G(x)$ (r=4). **Append *r* zero bits to the low-order end of the frame** so it now contains $m + r$ bits and corresponds to the polynomial $x^rM(x)$.
    - **Divide the bit string corresponding to $G(x)$** into the bit string corresponding to $x^rM(x)$, using modulo 2 division.
    - **Subtract the remainder** (which is always r or fewer bits) from the bit string corresponding to $x^rM(x)$ using modulo 2 subtraction.
    - The result is the checksummed frame to be transmitted. Call its polynomial $T(x)$.

Example

Data: **1101001** and $G(x) = x^4+x+1$ (**10011**)

5 bits polynomial add **4** bits as the checksum – so add 0000

```
10011 | 11010010000
        10011
        ------
        010010
         10011
         ------
         000011000
              10011
              ------
              010110
               10011
              ------
               00101
```

Data sent:
11010010101

Error Detection Codes

- **Parity Bit** (1 bit): (Hamming distance=2)
- **Internet Checksum** (16 bits): (Hamming distance=2)
- **Cyclic Redundancy Check** (CRC) (Standard 32-bit CRC: Hamming distance=4)

Error Correction: Hamming Code

- How many check bits are required for *n* bits of data?

  At least *k* check bits

  $$n \leq 2^k - k - 1$$

  Example: data 0101 → requires 3 check bits

  $$4 = (2^3) - 3 - 1$$

- Put check bits in **positions *p* that are power of 2**, starting with position 1
- Check bit in **position *p* is parity of positions with a *p* term in their value**

Example

Put check bits in positions p that are power of 2, starting with position 1

- Data: 0101 → requires 3 check bits

111

Position	P1	P2	P3	P4	P5	P6	P7
Data	?	?	0	?	1	0	1

1. Calculate the parity bits for P1, P2, P4 (rule: even parity)

P1 + P3 + P5 + P7 = ?+0+1+1 (even) → P1 = 0

P2 + P3 + P6 + P7 = ?+0+0+1 (odd) → P2 = 1

P4 + P5 + P6 + P7 = ?+1+0+1 (even) → P4 = 0

Data sent: 0100101

error

Example 1: At the receiver: 010010**0**

P1 + P3 + P5 + P7 = 0+0+1+0= 1 ×

P2 + P3 + P6 + P7 = 1+0+0+0= 1 ×

P4 + P5 + P6 + P7 = 0+1+0+0= 1 ×

Error bit: P1, P2, P4 → P(1+2+4)=P7

error

Example 2: At the receiver: 0**0**00101

P1 + P3 + P5 + P7 = 0+0+1+1= 0

P2 + P3 + P6 + P7 = 0+0+0+1= 1 ×

P4 + P5 + P6 + P7 = 0+1+0+1= 0

Error bit: P2

Error Control Discussion

- Error Correction: More efficient in noisy transmission media, e.g., wireless
- Error Detection: More efficient in the transmission media with low error rates, e.g., quality wires
- Require assumption on a specific number of errors occurring in transmission. Errors can occur in the check bits.