CS 302 Data Communication and Networks

Lecture 14: Data Link Layer

Data Link Layer

- The main responsibilities of the data link layer are:
  - Error detection & error correction
  - Data link control (DLC)
    - Framing
    - Flow control & error control
  - Media access control (MAC)

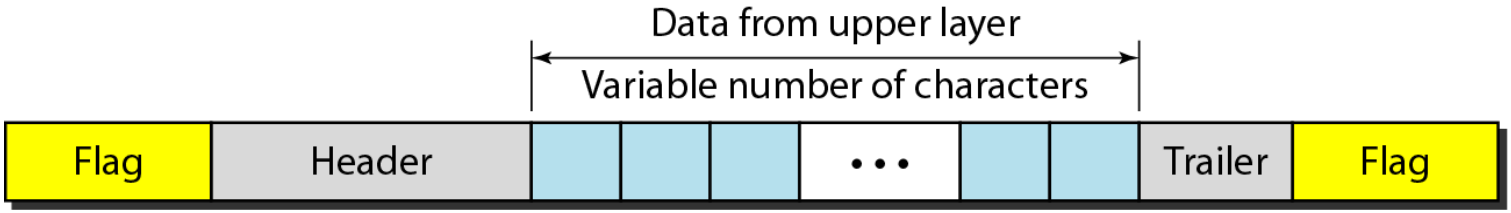
DLC: Framing

- The data link layer needs to pack bits into frames, so that each frame is distinguishable from another
- Framing in the data-link layer separates a message from one source to a destination by adding a sender address and a destination address
- Frames can be of fixed or variable size
- **Fixed-size framing:** There is no need for defining the boundaries of the frames; the size itself can be used as a delimiter
- **Variable-size framing:** We need a way to define the end of one frame and the beginning of the next
  - Historically, two approaches were used for this purpose: a **character-oriented approach** and a **bit-oriented approach**

Character-Oriented Framing

- In character-oriented (or byte-oriented) framing, data to be carried are 8-bit characters from a coding system like ASCII
- The header and trailer are also multiples of 8 bits
- To separate one frame from the next, an 8-bit (1-byte) flag is added at the beginning and the end of a frame
- The flag, composed of protocol-dependent special characters, signals the start or end of a frame
- It was popular when only text was exchanged by the data-link layers and the flag could be selected to be any character not used for text communication

A frame in a character-oriented protocol



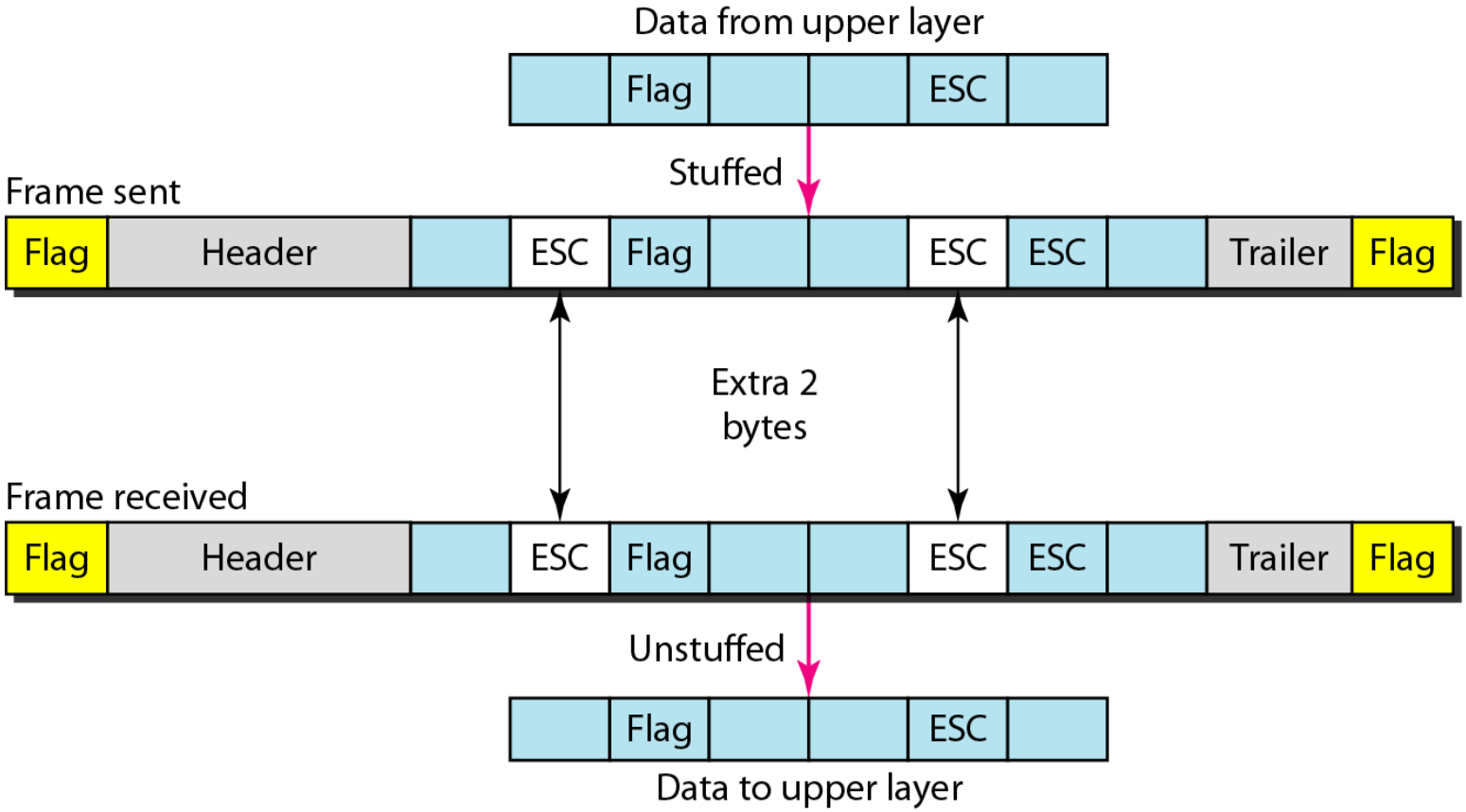
Character-Oriented Framing

- Now, however, we send other types of information like graphs, audio, and video; any character used for the flag could also be part of the information
- If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame
- To fix this problem, a **byte-stuffing** strategy was added to character-oriented framing
- In **byte stuffing (or character stuffing)**, a special byte is added to the data section of the frame when there is a character with the same pattern as the flag
- Means, the data section is stuffed with an extra byte
- This byte is usually called the *escape character (ESC)* and has a predefined bit pattern

Character-Oriented Framing

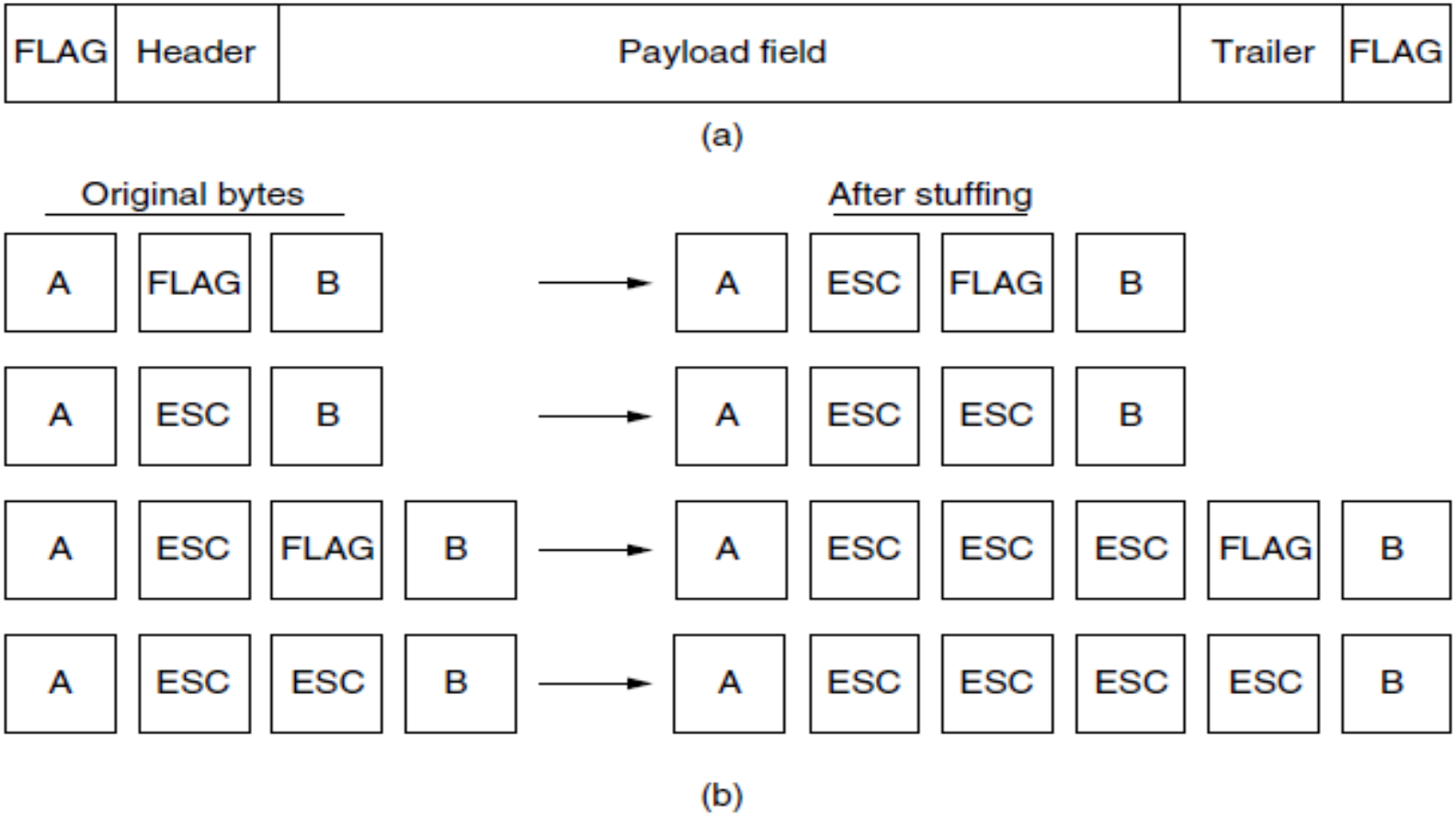
- Whenever the receiver encounters the *ESC* character, it removes it from the data section and treats the next character as data
- If the *ESC* character is part of the text, an extra one is added to show that the second one is part of the text
- **Problem:** The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters
- To overcome this we can move towards the bit-oriented protocols

Byte stuffing and unstuffing



Note: Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text

Examples: Byte stuffing

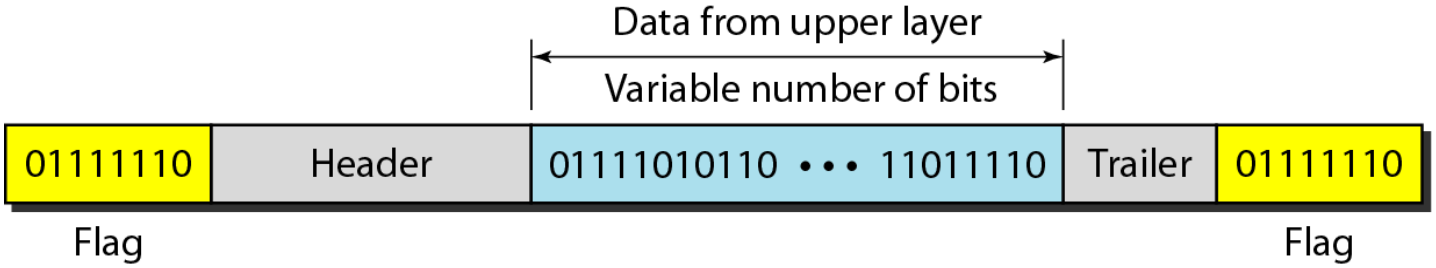


(a) A frame delimited by flag bytes. (b) Four examples of byte sequences before and after byte stuffing.

Bit-Oriented Framing

- In bit-oriented framing, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video etc.
- Most protocols use a special 8-bit pattern flag, 01111110, as the delimiter to define the beginning and the end of the frame
- This flag can create the same type of problem we saw in the character-oriented protocols
- This problem can be solved by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag
- The strategy is called **bit stuffing**

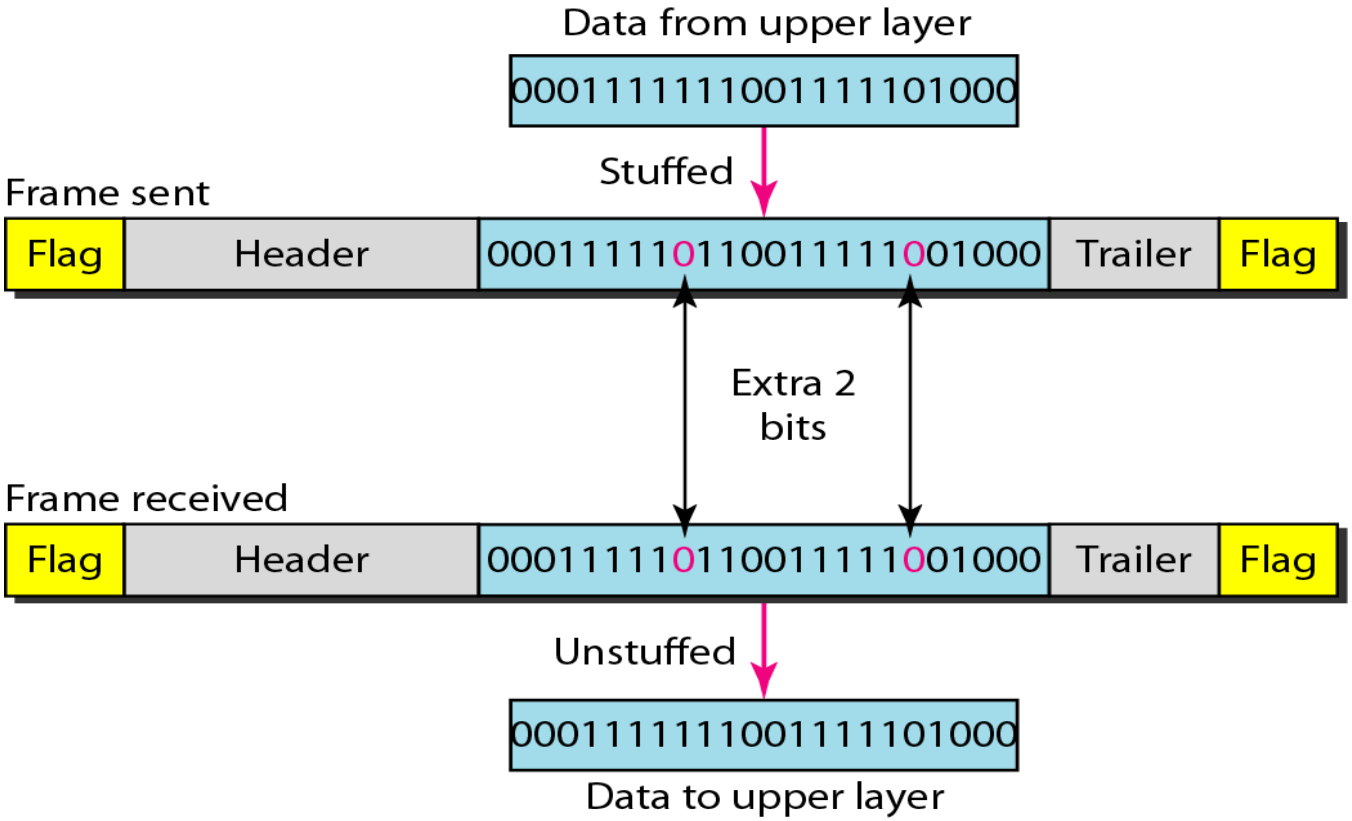
A frame in a bit-oriented protocol



Bit-Oriented Framing

- **Bit stuffing** is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not make any mistake to identify the flag pattern

Bit stuffing and unstuffing



End