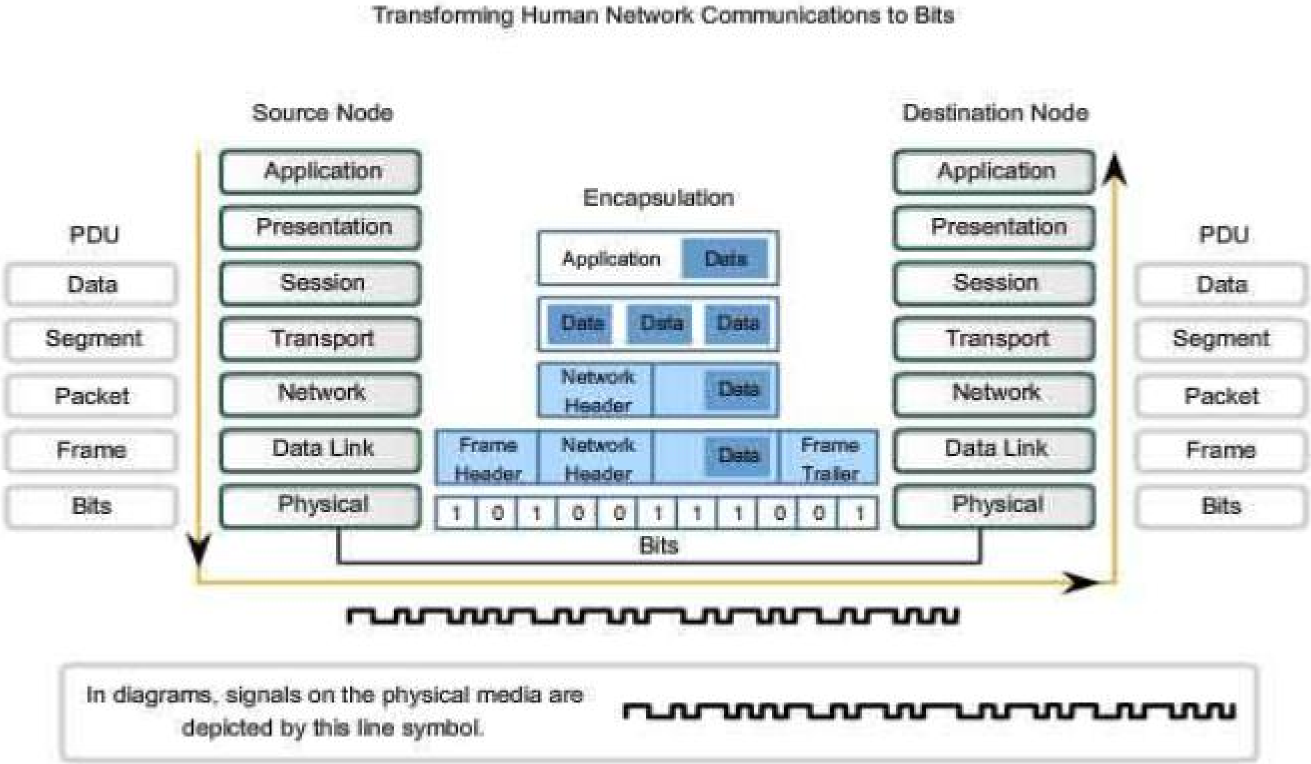
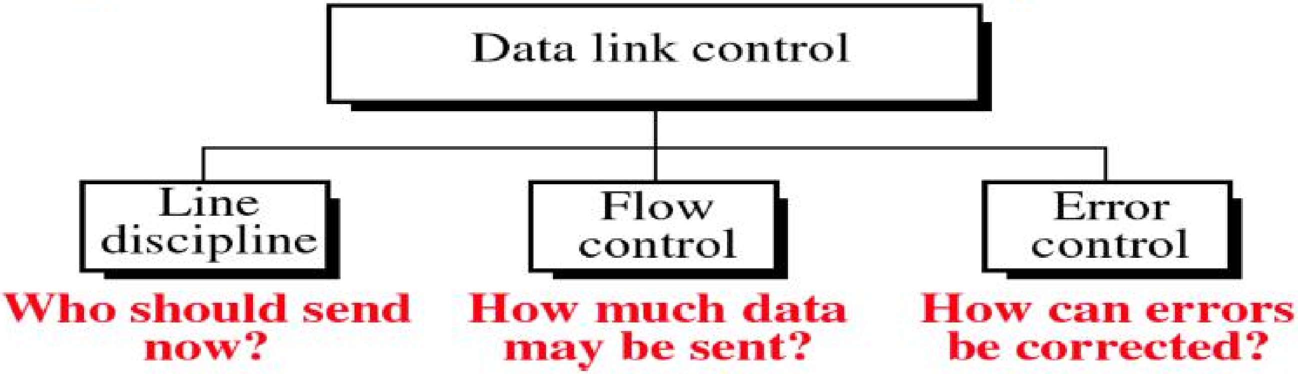
Data Link Layer: An Overview



Responsibilities of Data Link Layer



- Specific responsibilities of the data link layer include ***framing***, ***addressing***, ***flow control***, ***error control***, and ***media access control***.
- It provides service interface to the network layer.
- The data link layer adds a header to the frame to define the addresses of the sender and receiver of the frame.
- ***Framing:***

- Data link layer deals data in chunks generally called ***Frames.***
- Size of frame is typically a few hundreds/thousands of bytes.
- The data link layer is concerned with local delivery of frames between devices on the same LAN.
- It creates/recognizes frame boundaries i.e., start and stop of the frame in order to distinguish between them.

- ***Error Control***

- Data link layer provides a mechanism for error detection and correction.
- Thus, data link layer ensures a reliable transmission.

- ***Flow Control***

- If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, overflow occurs.
- The data link layer imposes a flow control mechanism to prevent transmitter from overrunning the receiver buffer.

- ***Address Information***

- Source and destination address are required, since a channel carries information for multiple users.
- Addressing information is added in each frame.

- ***Implements data link control protocols***

- It also has to implement several data link control protocol such as High Level Data Link Control (**HDLC**), **ATM**, **frame relay**, etc.

- ***Line Discipline***

- When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

Framing

Framing

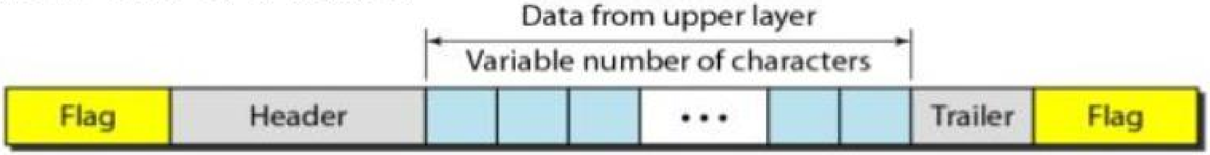
- Data link layer converts stream of bits (from physical layer) into frames.
- Each frame is made distinguishable from one another by appending source and destination address.
- Although, the whole message can be packed in one frame, that is not normally done. This is because, a very large frame will not make flow and error control very efficient.
- For e.g., even for a single bit error, we need to retransmit the complete frame.

- Size of the frame can be either fixed or variable.
- **Fixed-Size Framing:** In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the **ATM wide-area network**, which uses frames of fixed size called cells.
- **Variable-Size framing:** It is the most important as it is widely used in **local-area networks**. In variable-size framing, we need a way to defined boundaries, i.e., beginning and end of the frames.
  - Historically, two approaches were used for this purpose:

  1) Character-oriented approach   2) Bit-oriented approach

Variable-Size Framing: Character-Oriented Protocol

- In character oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII.
- The header which consists of source and destination addresses and other control information, and the trailer, which carries error detection and correction redundant bits, are also multiple 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 the frame.
- The flag composed of protocol dependent special characters, signals that start or end of a frame.



- **A challenge:** Flag can be any thing which is not used in the data to be sent using DLL. This is well when we use only text as a data to be sent. But in general, we are sending audio, video, images, etc. as data which can have the same pattern as used for flag.

Solution: To fix this problem, we use byte stuffing (or character stuffing).

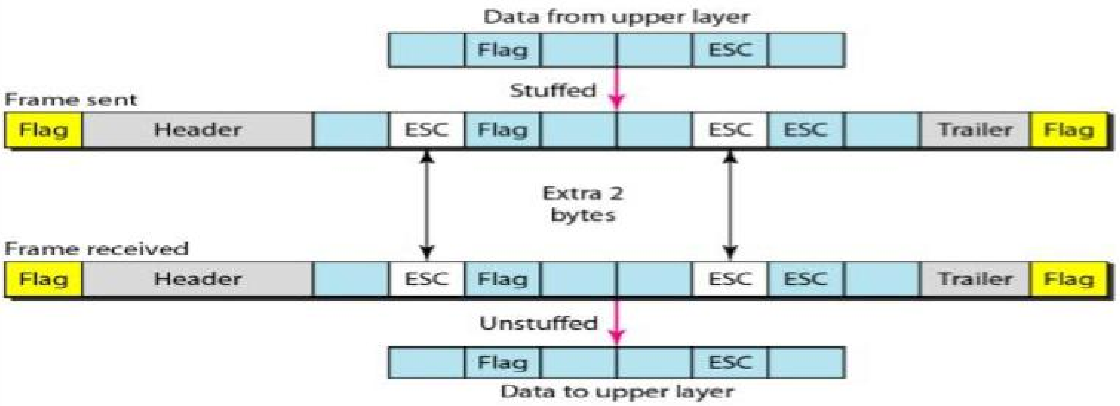
- **Byte stuffing:** The data section is stuffed with an extra byte, called escape character (ESC), whenever there is a character with the same pattern as flag in the data. ESC character has a predefined bit pattern.

Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

Byte Stuffing

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

Byte stuffing and unstuffing



E.g. of byte stuffing when data to be transmitted is: WE WILL NOT ESCAPE FROM THE QUIZ FOR THE FLAG HOSTING

Data to be sent

WE WILL NOT ESCAPE FROM THE QUIZ FOR THE FLAG HOISTING

Frame sent

FLAG	HEADER	WE WILL NOT **ESCAPE** ESCAPE FROM THE QUIZ FOR THE **ESCAPE** FLAG HOISTING	TRAILER	FLAG

Frame received

FLAG	HEADER	WE WILL NOT **ESCAPE** ESCAPE FROM THE QUIZ FOR THE **ESCAPE** FLAG HOISTING	TRAILER	FLAG

Data after unstuffing

WE WILL NOT ESCAPE FROM THE QUIZ FOR THE FLAG HOISTING

Variable-Size Framing: Bit-Oriented Protocol

- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, etc.
- Apart from header and trailer, we still need a delimiter, called flag to separate one frame from other.
- Most of the protocols use a special 8-bit pattern flag **01111110** as the delimiter to define beginning and end of the frame.

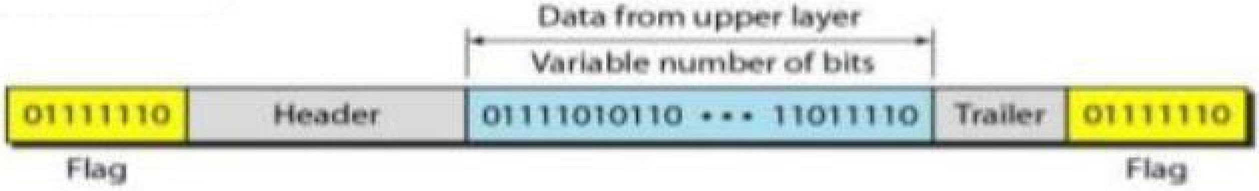
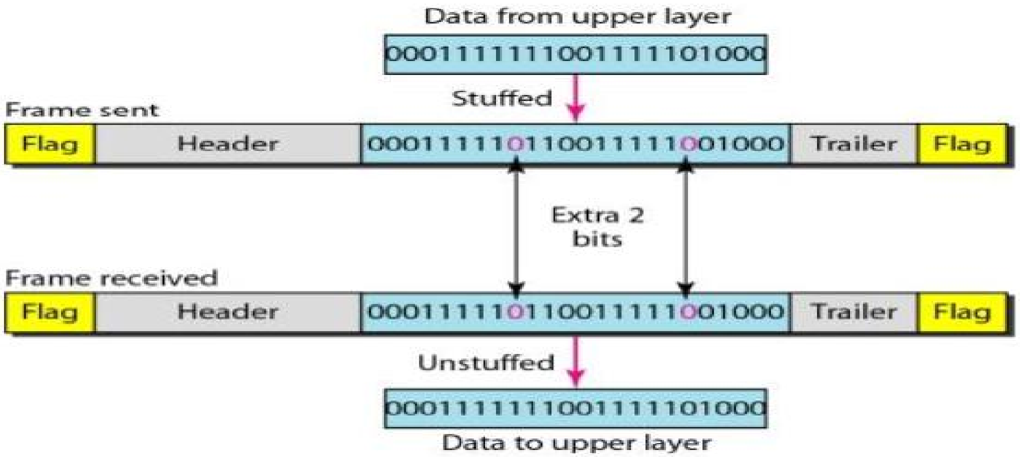


Fig. A frame in bit-oriented protocol

- **A Challenge:** The same type of problem (as happened in byte-oriented protocol) can occur here, if the flag pattern occurs in the data.
- **Sol:** *This problem is solved here by stuffing a single bit (instead of one byte in character oriented protocol) to prevent the pattern looking like a flag. This process is called bit-stuffing.*
- **Bit stuffing:** In this process, one extra 0 is added whenever five consecutive 1s follow a 0 in the data. This extra stuffed bit is eventually removed from the data by receiver.

Bit stuffing and unstuffing



Note: A bit-oriented protocol is actually implemented by using the High-level Data Link Control (**HDLC**) Protocol.
Whereas a popular byte-oriented protocol is implemented by using Point-to-Point Protocol (**PPP**).