Data Link Layer

Overview

- Performs the most reliable node to node delivery of data.
- The data link layer is divided into two sub-layers :

1. Logical Link Control Sub-layer (LLC) –
   - Provides the logic for the data link , Thus it controls the synchronization , flow control , and error checking functions of the data link layer. Functions are –

   (i) Error Recovery.

   (ii) It performs the flow control operations.

   (iii) User addressing.

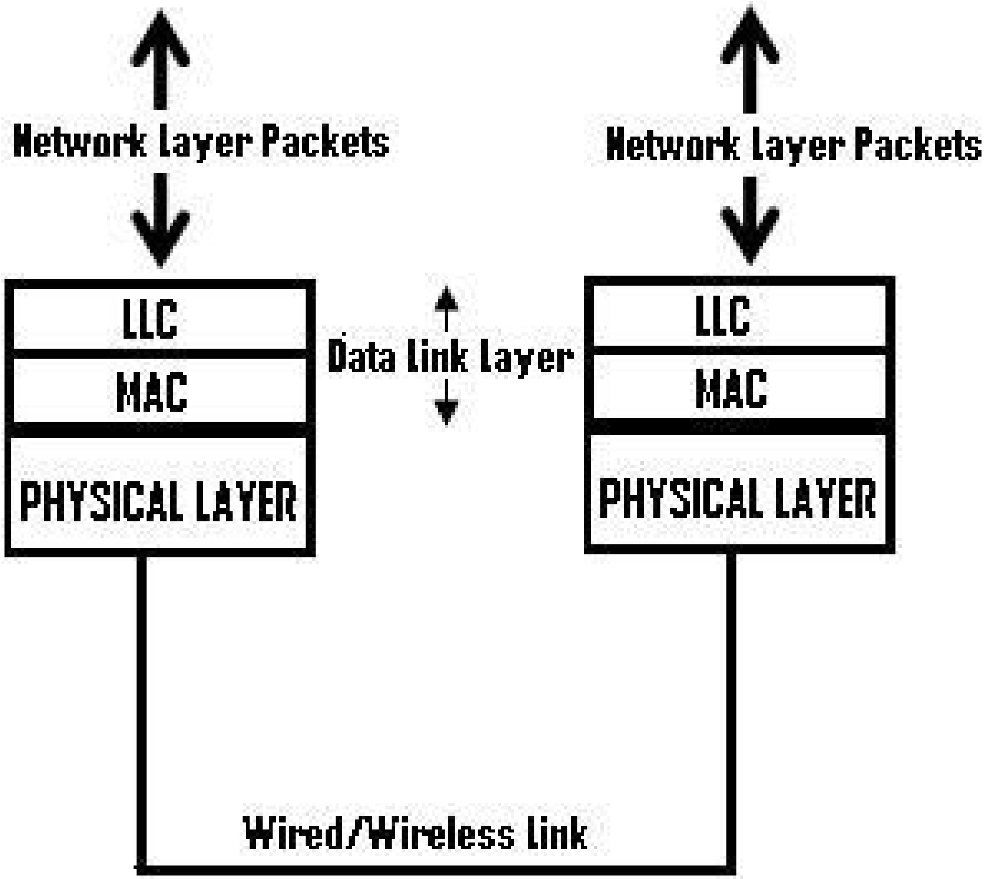
Overview

2. Media Access Control Sub-layer (MAC) –
   - It is the second sub-layer of data-link layer. It controls the flow and multiplexing for transmission medium. Transmission of data packets is controlled by this layer. This layer is responsible for sending the data over the network interface card. Functions are –

(i) To perform the control of access to media.

(ii) It performs the unique addressing to stations directly connected to LAN.

(iii) Detection of errors.


Network Layer Packets
Network Layer Packets
LLC
MAC
PHYSICAL LAYER
Data Link Layer
LLC
MAC
PHYSICAL LAYER
Wired/Wireless Link

Functions of Data Link Layer

- **Framing**: Frames are the streams of bits received from the network layer into manageable data units. This division of stream of bits is done by Data Link Layer.
- **Physical Addressing**: The Data Link layer adds a header to the frame in order to define physical address of the sender or receiver of the frame, if the frames are to be distributed to different systems on the network.
- **Flow Control**: A flow control mechanism to avoid a fast transmitter from running a slow receiver by buffering the extra bit is provided by flow control. This prevents traffic jam at the receiver side.

Functions of Data Link Layer

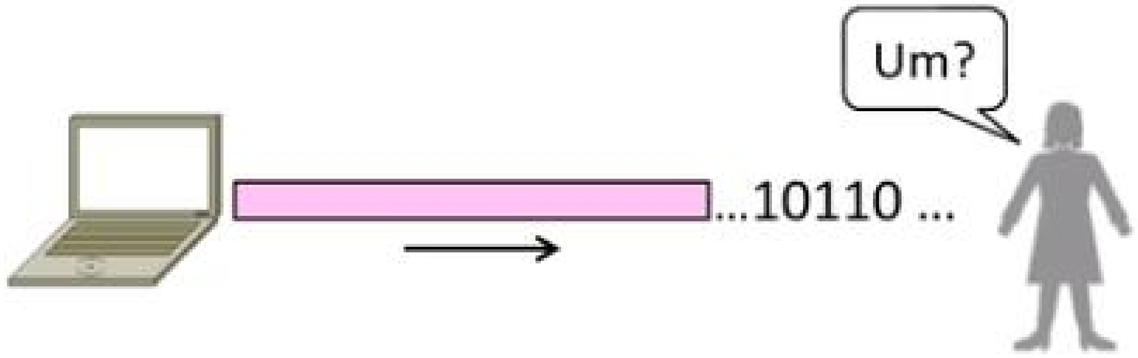
- **Error Control**: Error control is achieved by adding a trailer at the end of the frame. Duplication of frames are also prevented by using this mechanism. Data Link Layers adds mechanism to prevent duplication of frames.
- **Access Control**: Protocols of this layer determine which of the devices has control over the link at any given time, when two or more devices are connected to the same link.

Design Issues with Data Link Layer

- The issue that arises in the data link layer(and most of the higher layers as well) is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism is often needed to let the transmitter know how much buffer space the receiver has at the moment. Frequently, the flow regulation and the error handling are integrated.
- Broadcast networks have an additional issue in the data link layer: How to control access to the shared channel. A special sublayer of the data link layer, the Medium Access Control(MAC) sublayer, deals with this problem.

Framing

- The physical layer gives us a stream of bits. How do we interpret it as a sequence of frames.



Problems in Framing

- **Detecting start of the frame**: When a frame is transmitted, every station must be able to detect it. Station detect frames by looking out for special sequence of bits that marks the beginning of the frame i.e. SFD (Starting Frame Delimiter).
- **How do station detect a frame**: Every station listen to link for SFD pattern through a sequential circuit. If SFD is detected, sequential circuit alerts station. Station checks destination address to accept or reject frame.
- **Detecting end of frame**: When to stop reading the frame.

Types of framing

- **Fixed size** – The frame is of fixed size and there is no need to provide boundaries to the frame, length of the frame itself acts as delimiter.
    - Drawback: It suffers from internal fragmentation if data size is less than frame size
    - Solution: Padding 2.

Types of framing

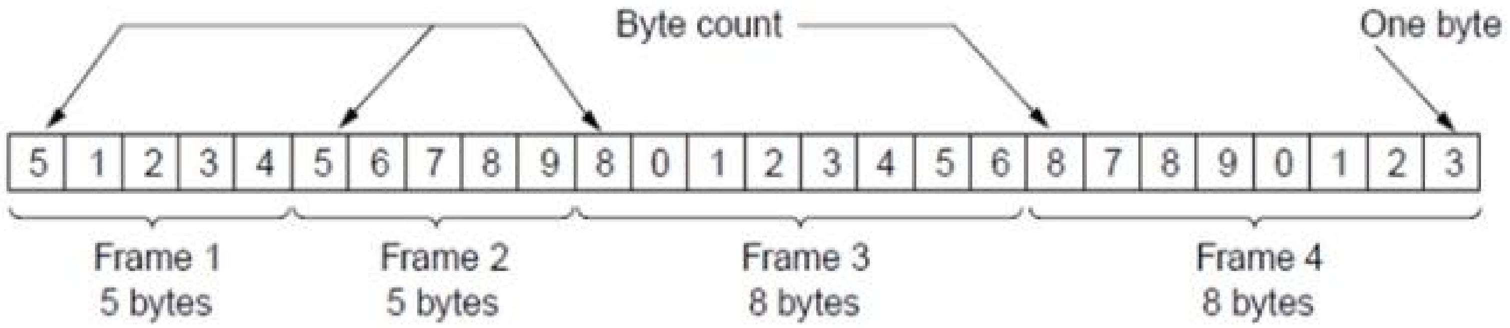
- **Variable size** – In this there is need to define end of frame as well as beginning of next frame to distinguish. This can be done in two ways:
  - Length field – We can introduce a length field in the frame to indicate the length of the frame. Used in Ethernet (802.3). The problem with this is that sometimes the length field might get corrupted.
  - End Delimiter (ED) – We can introduce an ED (pattern) to indicate the end of the frame. Used in Token Ring. The problem with this is that ED can occur in the data.

Framing Methods

- We’ll look at:
  - Byte Count
  - Byte stuffing
  - Bit stuffing
- In practice, the physical layer often helps to identify frame boundaries.
  - E.g., Ethernet, 802.11

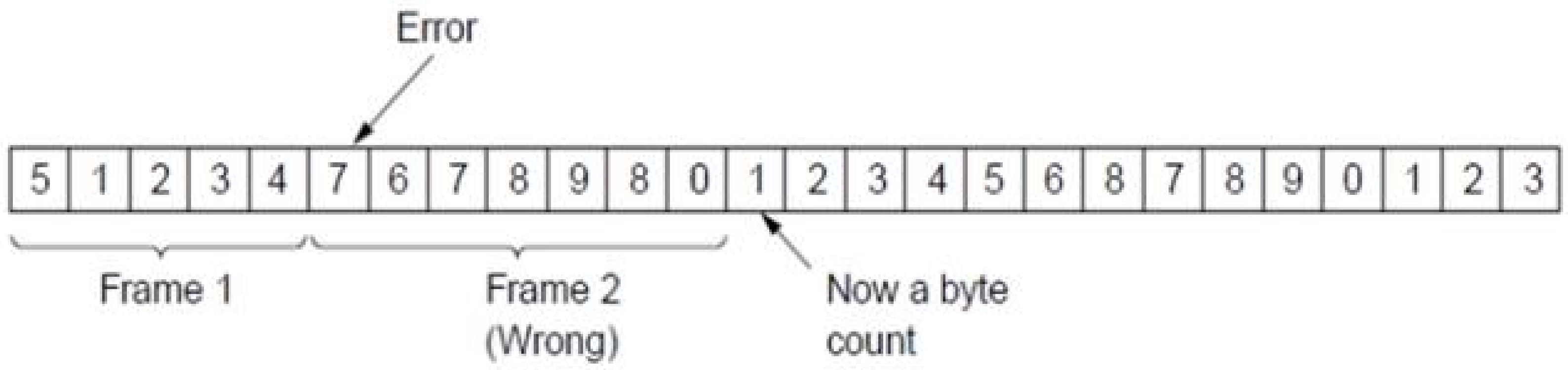
Byte Count

- First try:
  - Let's start each frame with a length field!
  - It's simple, and hopefully good enough …



Byte Count

- Difficult to re-synchronize after framing error.

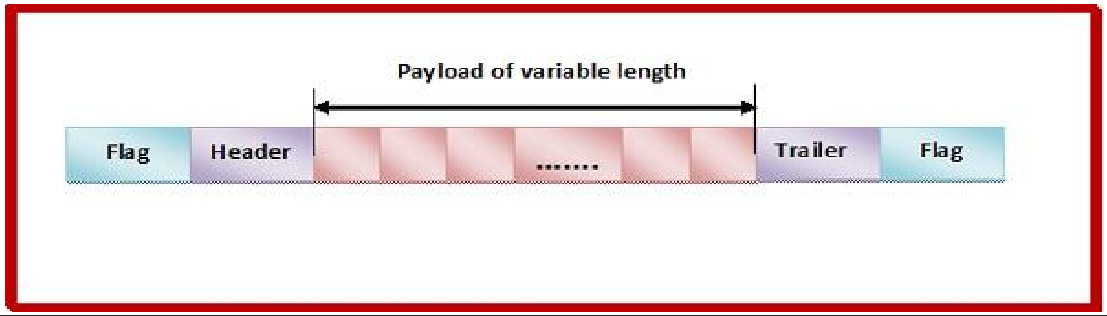


Byte Stuffing

- A byte is stuffed in the message to differentiate from the delimiter. This is also called character-oriented framing.
- Better idea:
  - Have a special flag byte value that means start/end of frame.
  - Replace the flag inside the frame with an escape code.
  - Complication: have to escape the escape code too!

Frame in a Character – Oriented Framing

- In character – oriented protocols, the message is coded as 8-bit characters, using codes like ASCII codes.
- A frame has the following parts –
  - Frame Header – It contains the source and the destination addresses of the frame.
  - Payload field – It contains the message to be delivered.
  - Trailer – It contains the error detection and error correction bits.
  - Flags – 1- byte (8-bits) flag at the beginning and at end of the frame. It is a protocol – dependent special character, signalling the start and end of the frame.



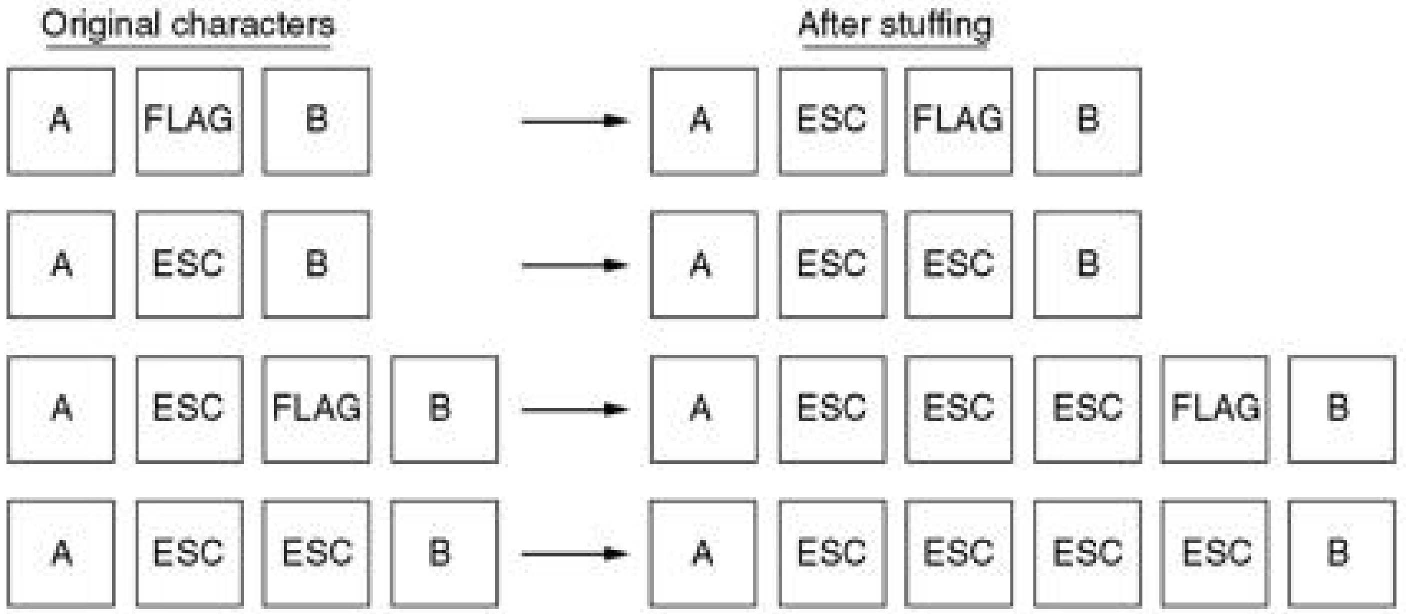
Byte Stuffing Mechanism

- If the pattern of the flag byte is present in the message byte, there should be a strategy so that the receiver does not consider the pattern as the end of the frame. In character – oriented protocol, the mechanism adopted is byte stuffing.

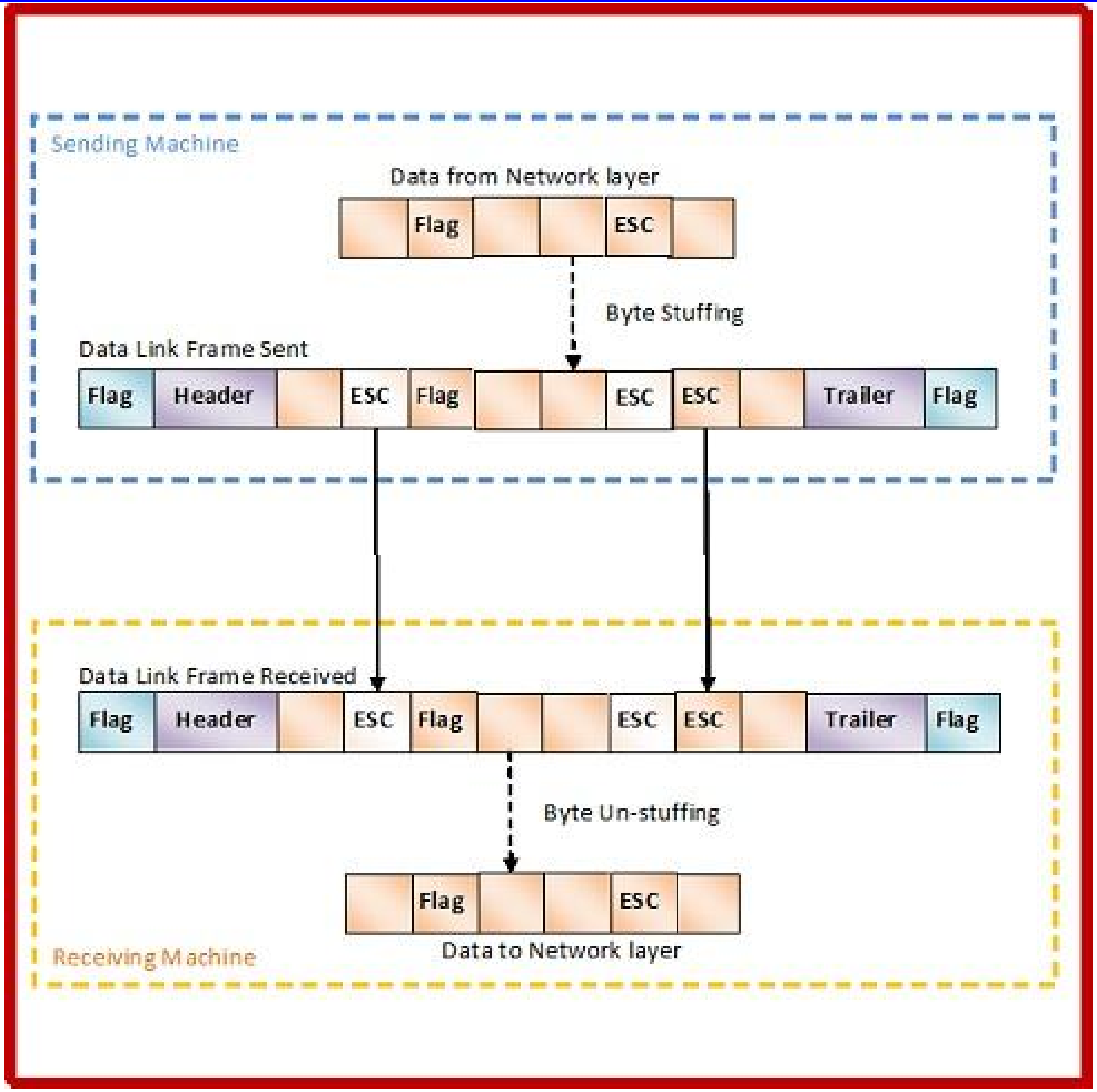
- In byte stuffing, a special byte called the escape character (ESC) is stuffed before every byte in the message with the same pattern as the flag byte. If the ESC sequence is found in the message byte, then another ESC byte is stuffed before it.

Byte Stuffing

- Rules:
  - Replace each FLAG in data with ESC FLAG.
  - Replace each ESC in data with ESC ESC.



Byte Stuffing

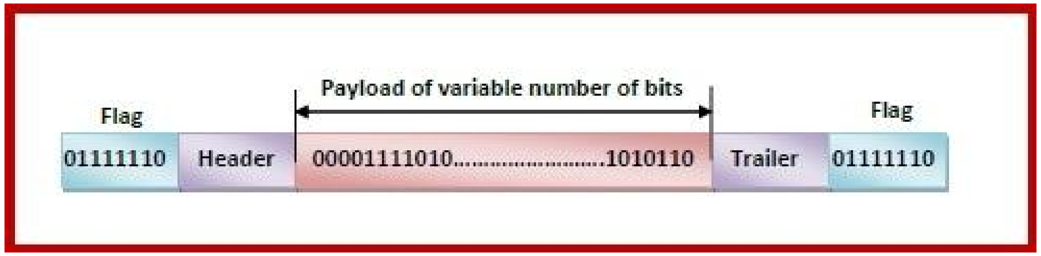


Bit Stuffing

- A pattern of bits of arbitrary length is stuffed in the message to differentiate from the delimiter. This is also called bit - oriented framing.

Frame in a Bit - Oriented Protocol

- In bit-oriented protocols, the message is coded as a sequence of bits, which are interpreted in the upper layers as text, graphics, audio, video etc. A frame has the following parts –
  - Frame Header – It contains the source and the destination addresses of the frame.
  - Payload field – It contains the message to be delivered.
  - Trailer – It contains the error detection and error correction bits.
  - Flags – A bit pattern that defines the beginning and end bits in a frame. It is generally of 8-bits. Most protocols use the 8-bit pattern 01111110 as flag.

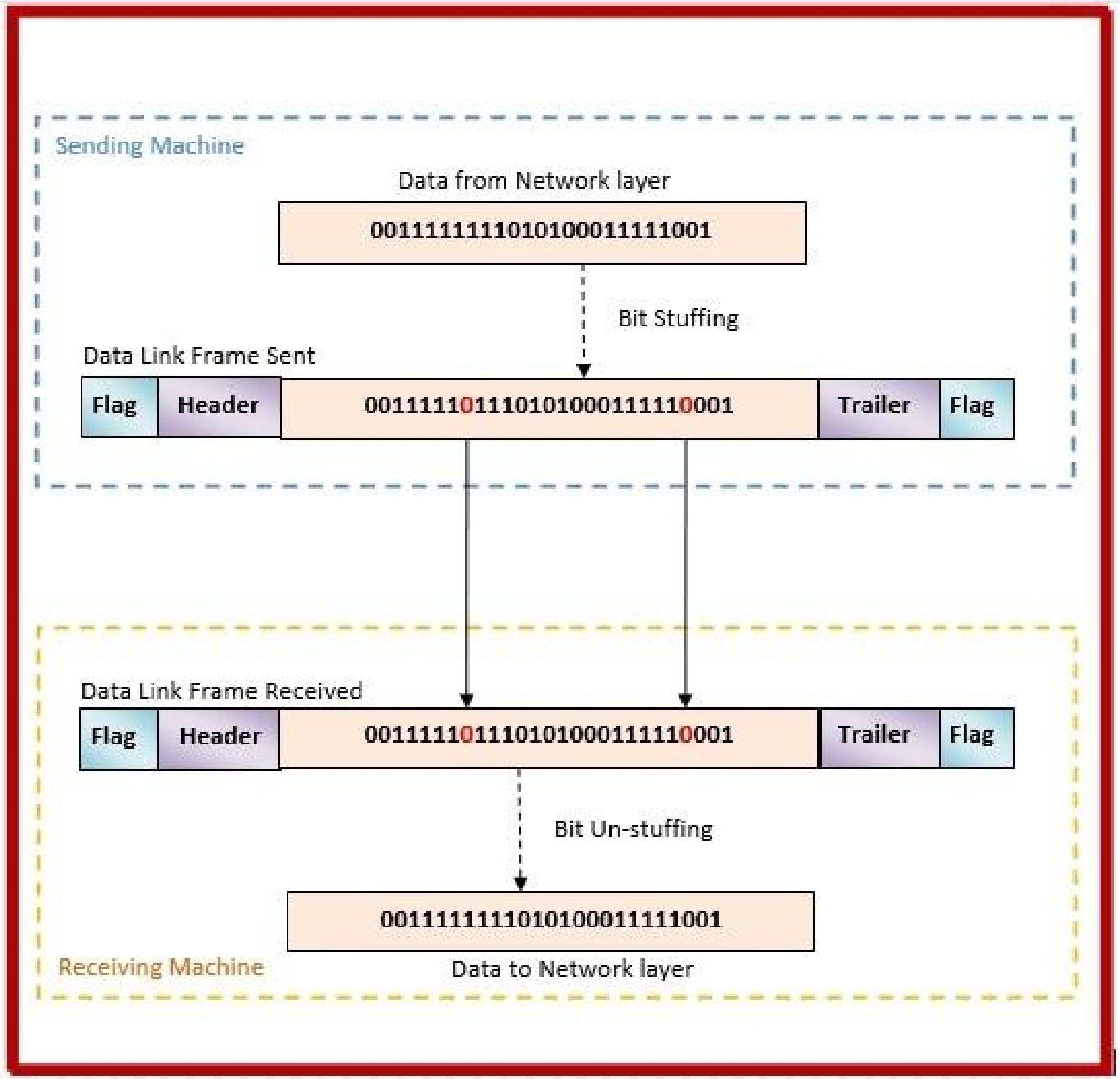


Bit Stuffing Mechanism

- In a data link frame, the delimiting flag sequence generally contains six or more consecutive 1s. In order to differentiate the message from the flag in case of the same sequence, a single bit is stuffed in the message. Whenever a 0 bit is followed by five consecutive 1bits in the message, an extra 0 bit is stuffed at the end of the five 1s.

- When the receiver receives the message, it removes the stuffed 0s after each sequence of five 1s. The un-stuffed message is then sent to the upper layers.

Bit Stuffing

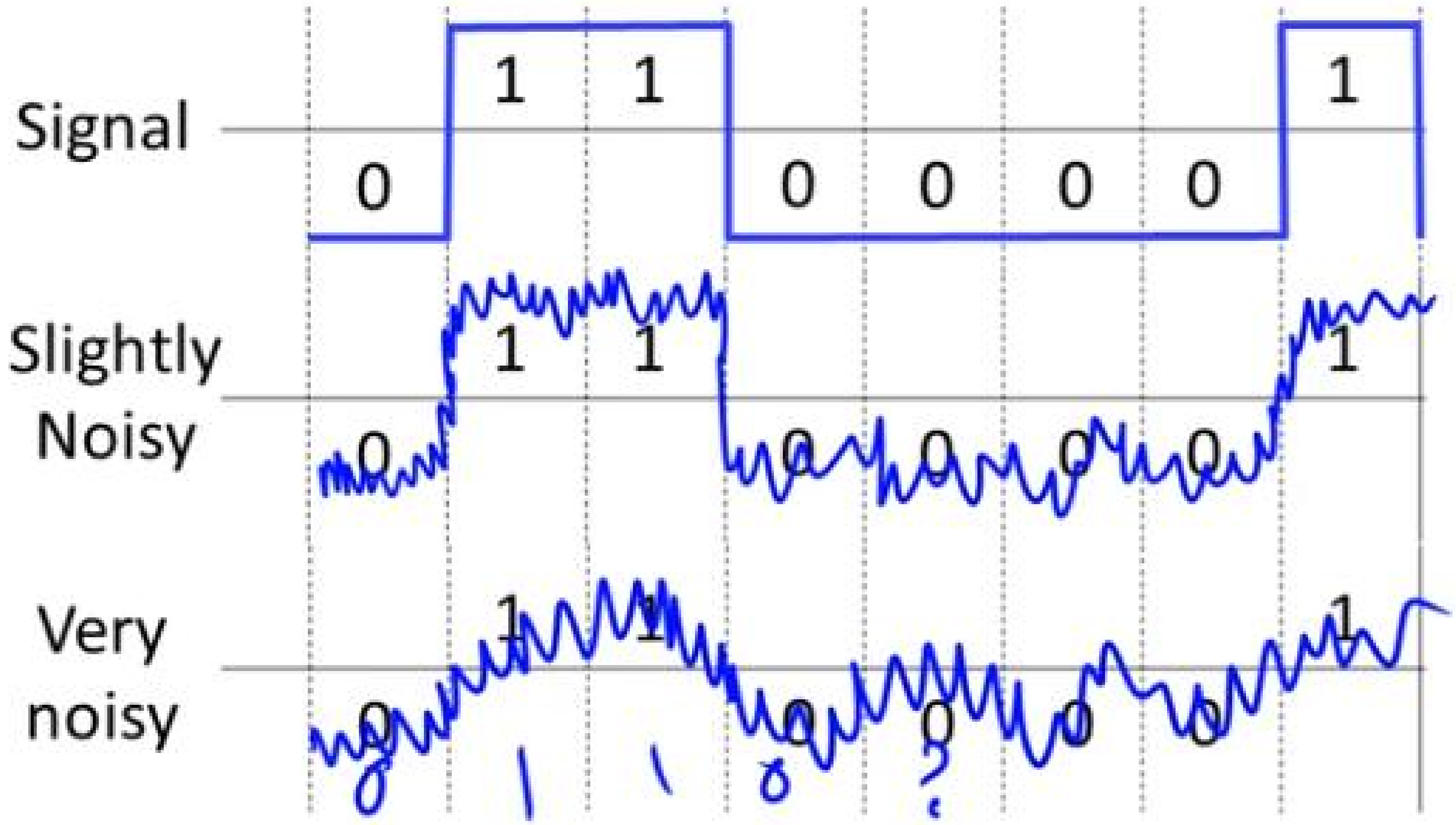


Error Coding Overview

- Some bits will be received in error due to noise. What can we do?
  - Detect errors with codes
  - Correct error with codes
  - Retransmit lost frames (Later)
- Reliability is a concern that cuts across the layers- we'll see it again.

Problem

- Noise may flip received bits



Approach- Add Redundancy

- Error detection codes
  - Add check bits to the message bits to let some errors be detected
- Error correction codes
  - Add more check bits to let some errors be corrected
- Key issue is now to structure the code to detect many errors with few check bits and modest computation.

Motivating Example

- Add a simple code to handle errors:
  - Send two copies! Error if different.

- How good is this code?
  - How many errors can it detect/correct?
  - How many error will make it fail?

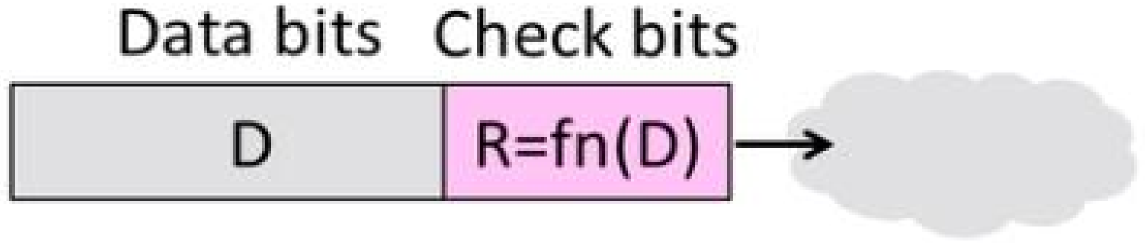
Motivating Example(2)

- We want to handle more errors with less overhead
  - Will look at better codes; they are applied mathematics
  - But, they can't handle all errors
  - And the focus on accidental errors

    (will look at secure hashes later)

Using Error Codes

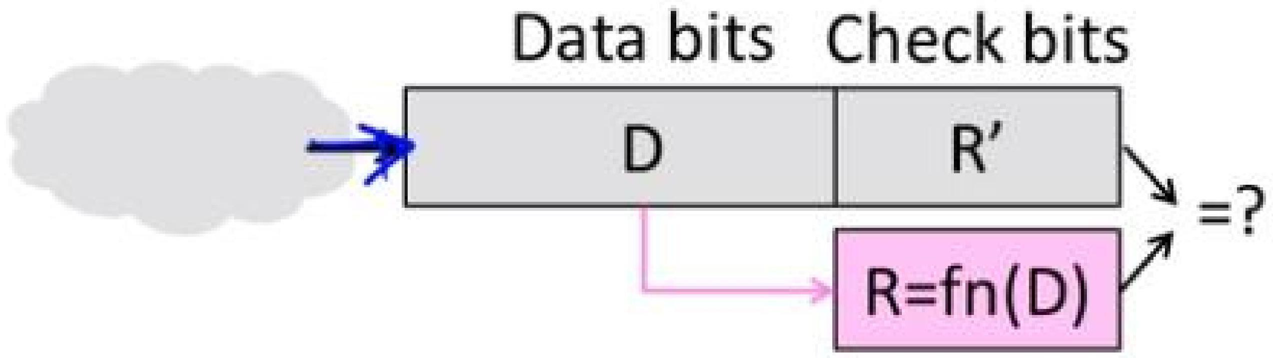
- Codeword consist of D data plus R check bits(=systematic block codes)



- Sender
  - —Compute R check bits based on the data d bits; send the codeword of D+R bits.

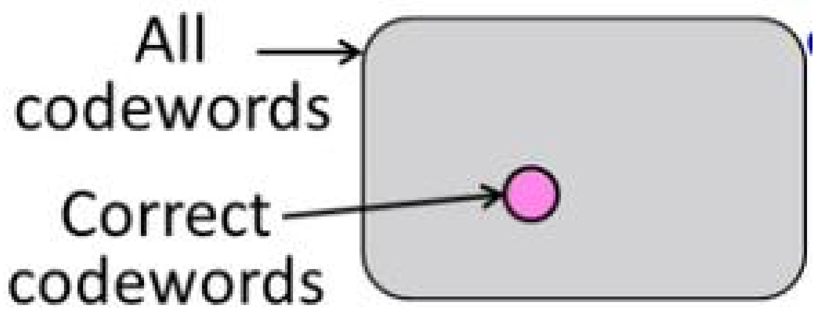
Using Error Codes(2)

- Receiver:
- Receive D+R with unknown errors
- Recompute R check bits based on the D data bits; error if R doesn’t match R’



Intuition for Error Codes

- For D data bits, R check bits:



- Randomly chosen codeword is unlikely to be correct; overhead is low

R.W Hamming (1915-998)

- Much early work on codes:
  - “Error Detecting and Error Correcting Codes”, 1986
- See also:
  - “You and Your Research”, 1986

Hamming Distance

- Distance is the number of bit flips needed to change D1 to D2

- Hamming distance of a code is the minimum distance between any **pair** of codewords.

Hamming Distance(2)

- Error Detection:
  - For a code of distance d+1, up to d errors will always be detected

Hamming Distance(3)

- Error correction:
  - For a codeword of distance 2d+1, up to d errors can always be corrected by mapping to the closest codeword.