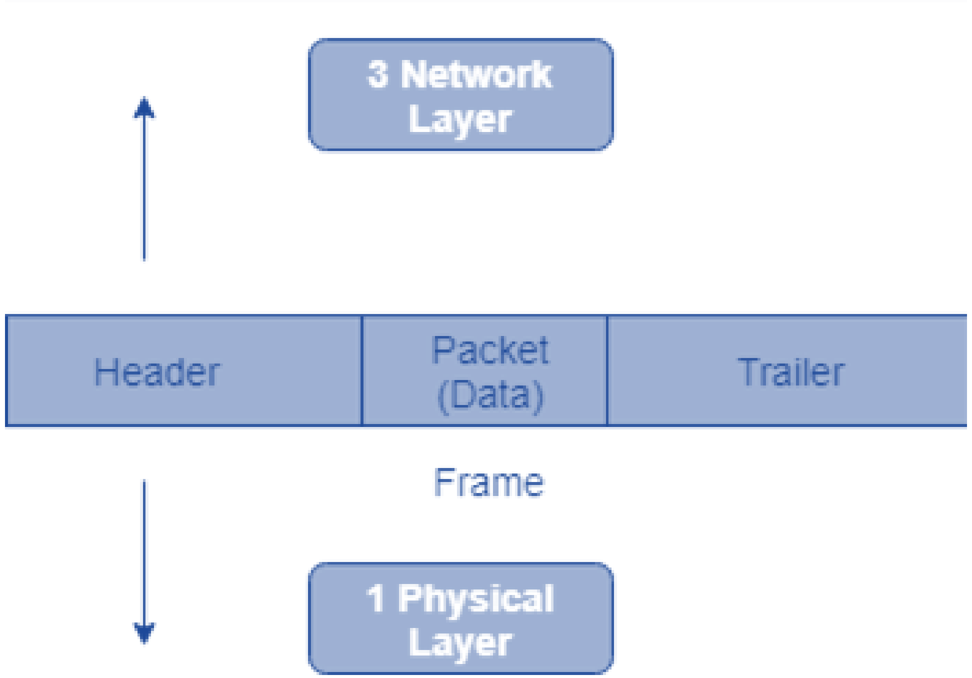
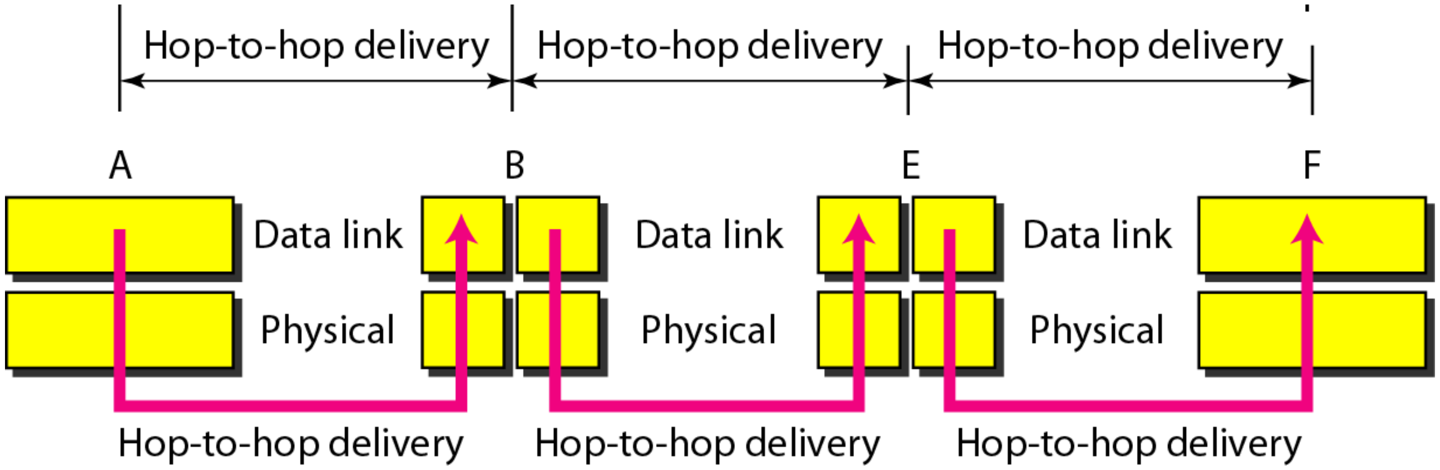


Data link Layer Framing

Data Link Layer

- Data link achieving reliable, efficient communication between two adjacent machines.
- Machines are connected by a communication channel that acts conceptually like wire (coaxial cable, telephone line, or point-to-point wireless channel)

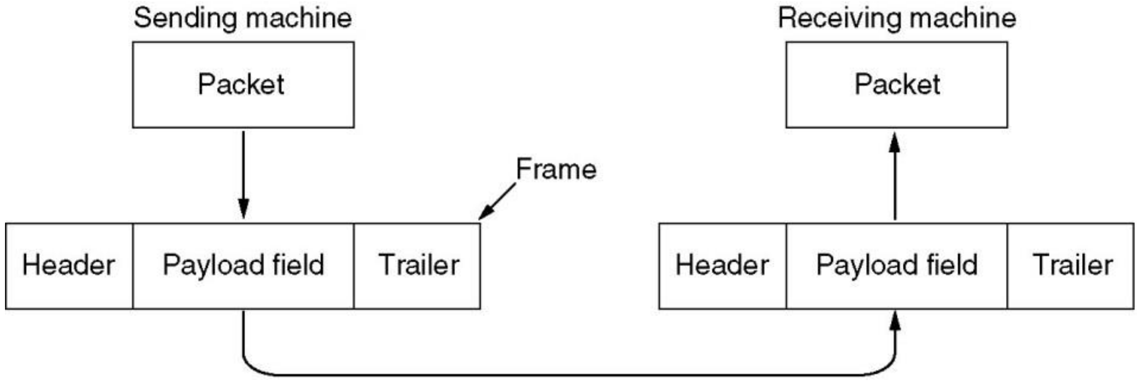


Data Link Layer – design issues

- Services Provided to the Network Layer
- Framing
- Error Control
- Flow Control

Data Link Layer – design issues

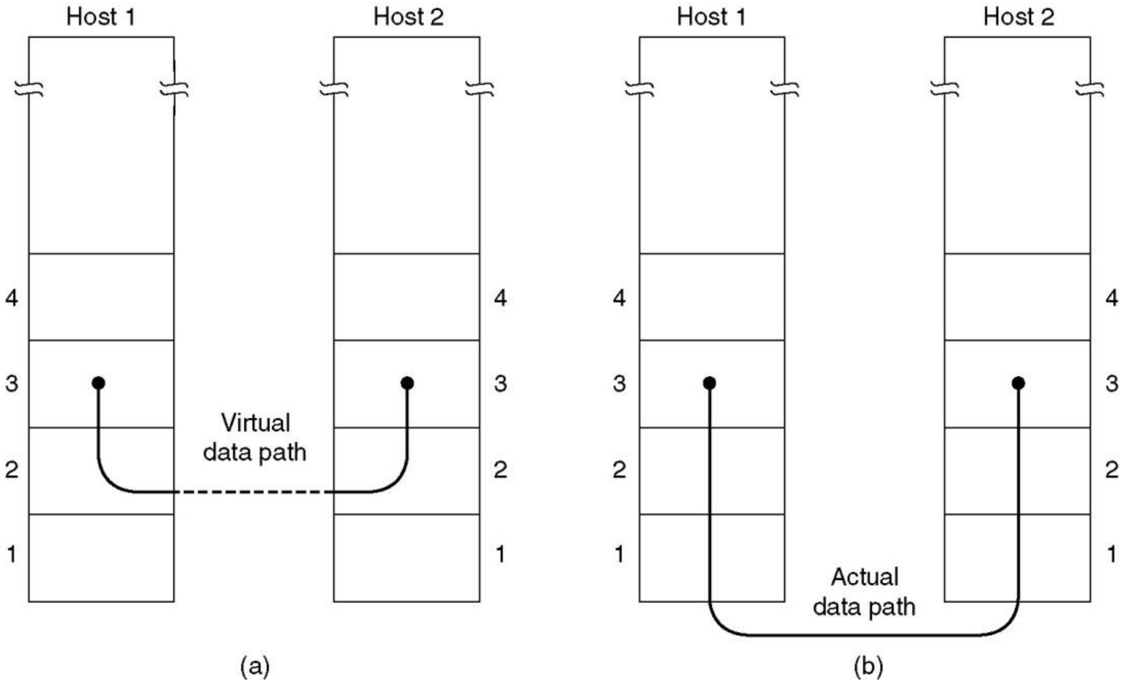
Services Provided to the Network Layer



The actual services offered can vary from system to system.

- Unacknowledged connectionless service.
- Acknowledged connectionless service.
- Acknowledged connection-oriented service.

(a) Virtual communication
(b) Actual communication



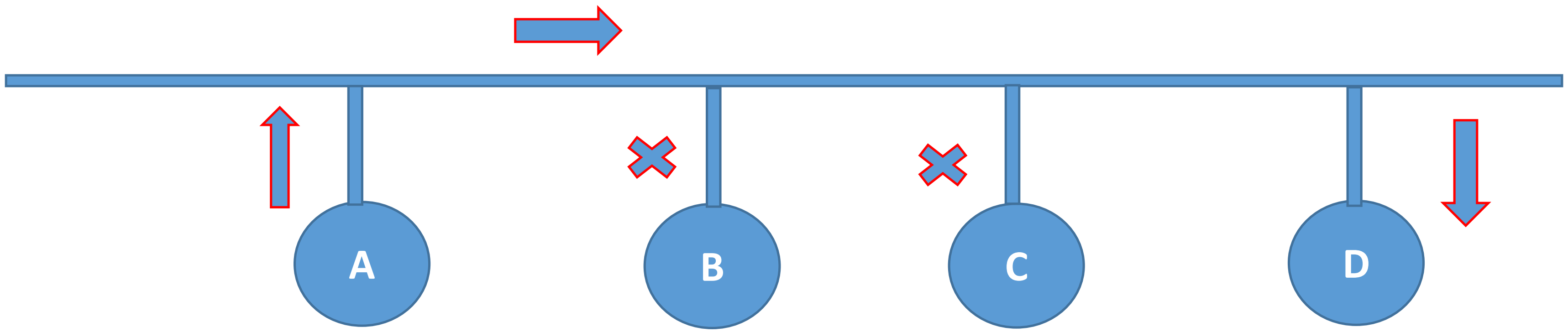
Framing

- Framing is a approach to break bit stream into discrete frames and compute the checksum for each frame.
- When a frame arrives at the destination, the checksum is recomputed.
- If the newly-computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it
- Discarding the bad frame and possibly also sending back an error report.

Fixed-Size Framing
Variable-Size Framing

Framing

Header	Payload field	Trailer

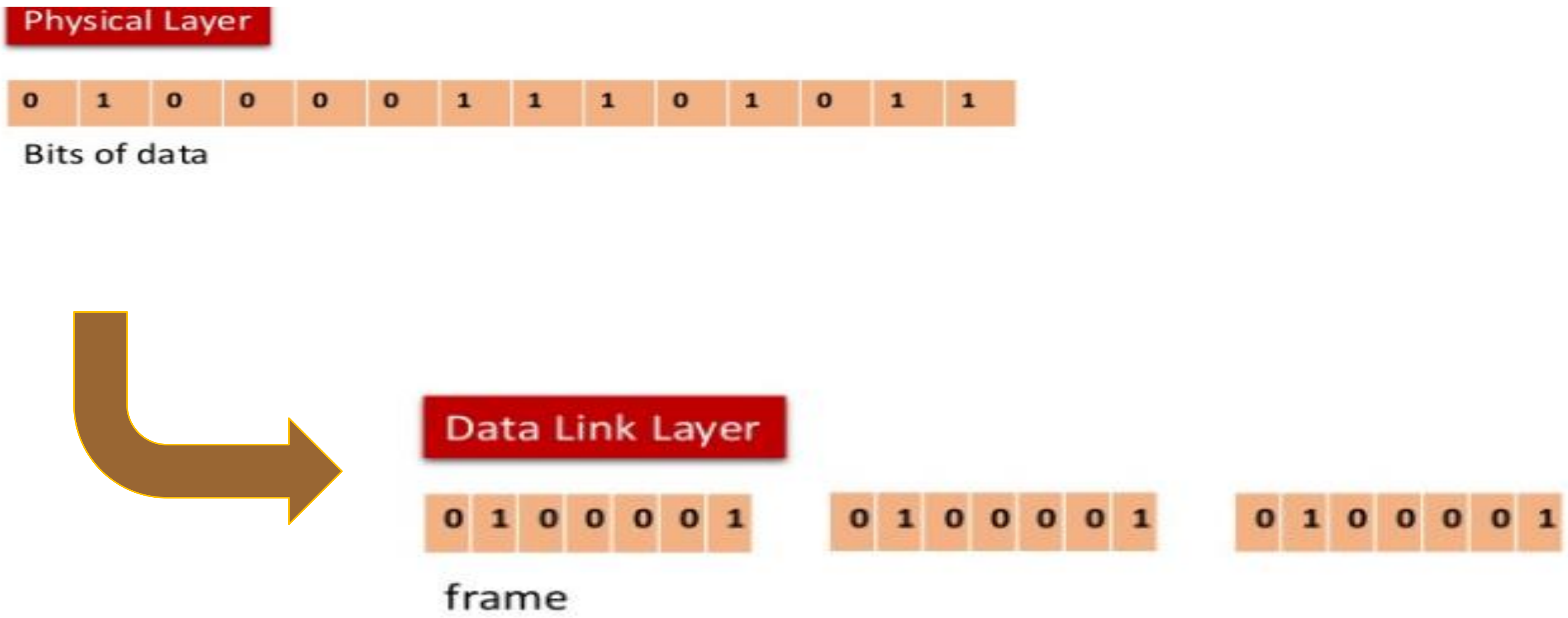


A,D	Data	Trailer

Framing

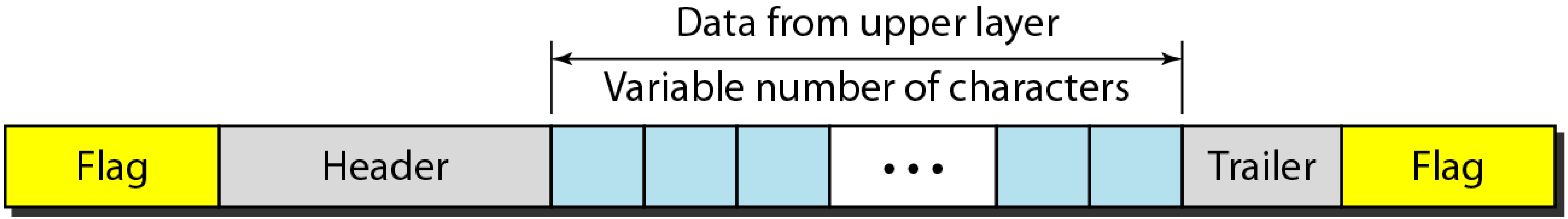
- Fixed –Size Framing
- Variable – Size framing

Fixed Size Framing



Problem: if data size is small then frame then we require to do padding

Variable Size Framing

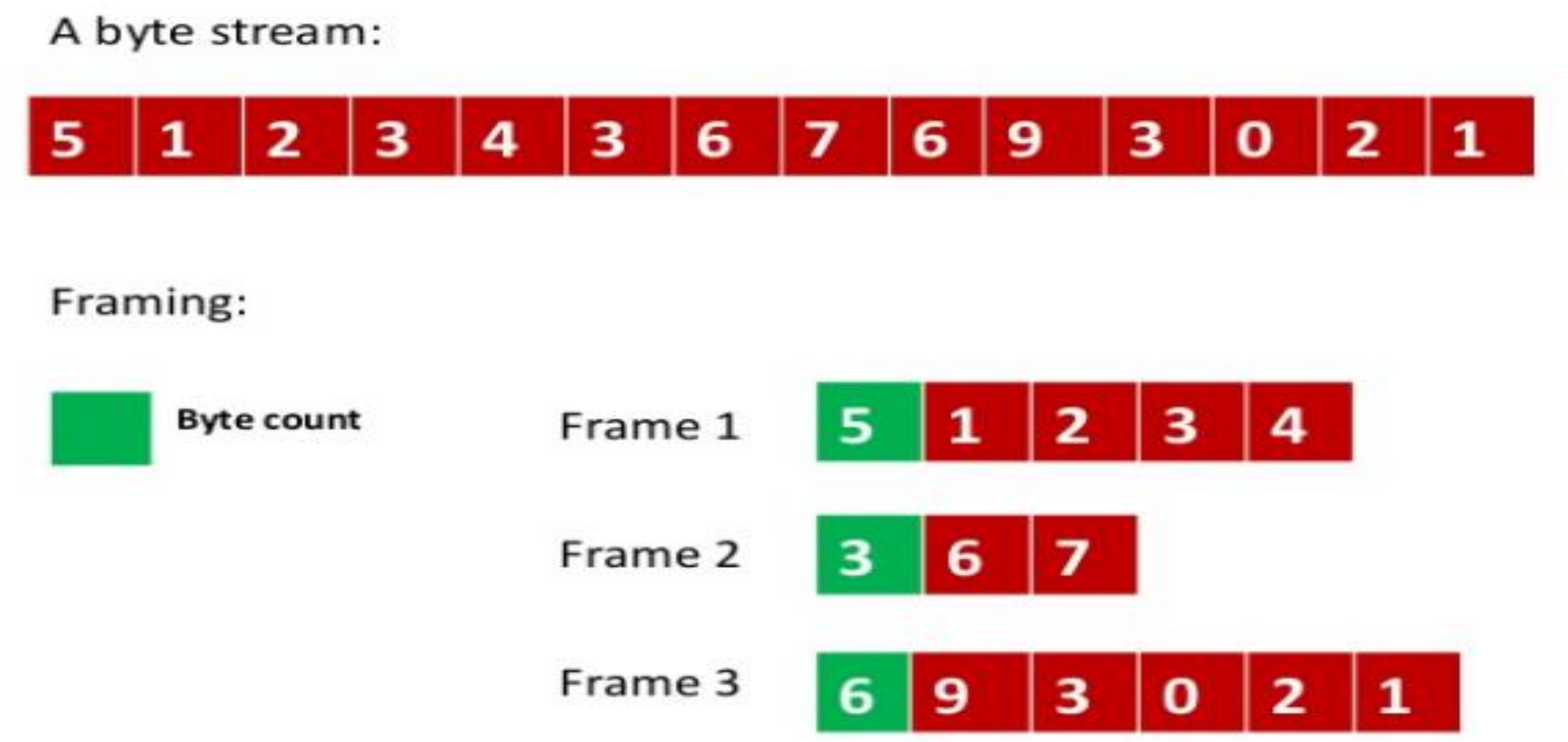


Framing Methods

- Character Count
- Byte stuffing
- Bit stuffing
- Physical layer coding violations

Character/Byte Count

- Uses a Field in a header to specify the number of bytes into frame



A byte stream:

5 1 2 3 4 3 6 7 6 9 3 0 2 1

Framing:

🟩 Byte count

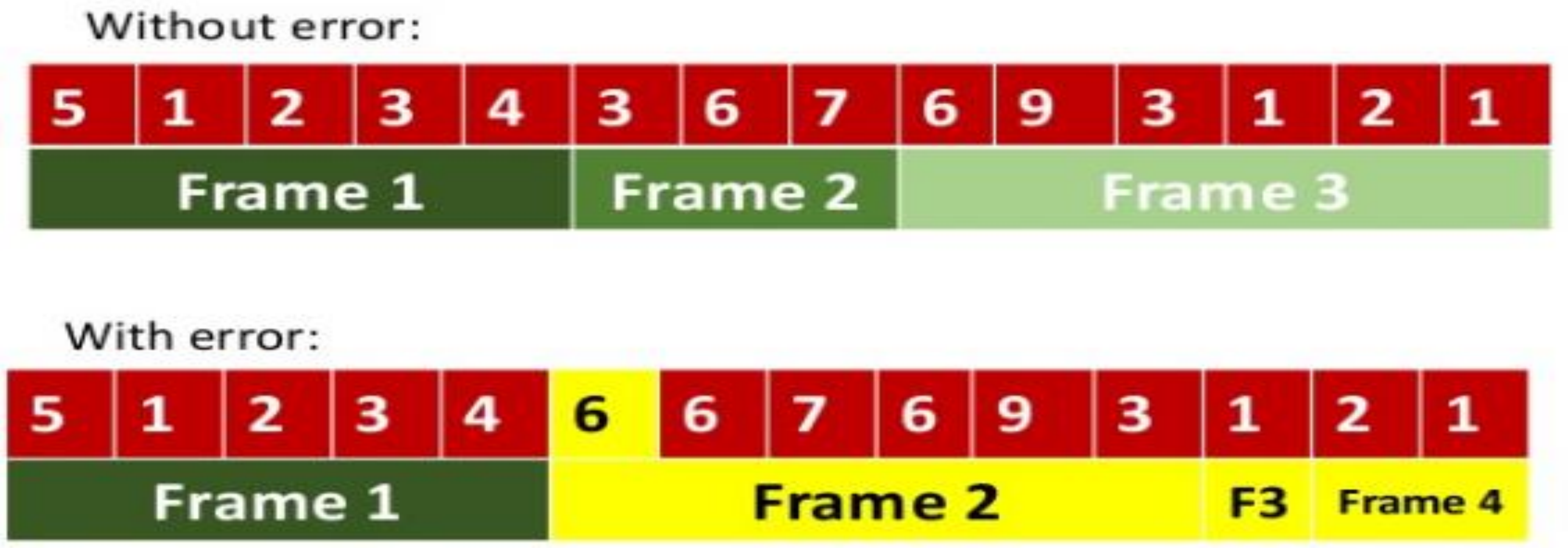
Frame 1: 5 1 2 3 4

Frame 2: 3 6 7

Frame 3: 6 9 3 0 2 1

Character/Byte Count

- Problem with byte count



Without error:

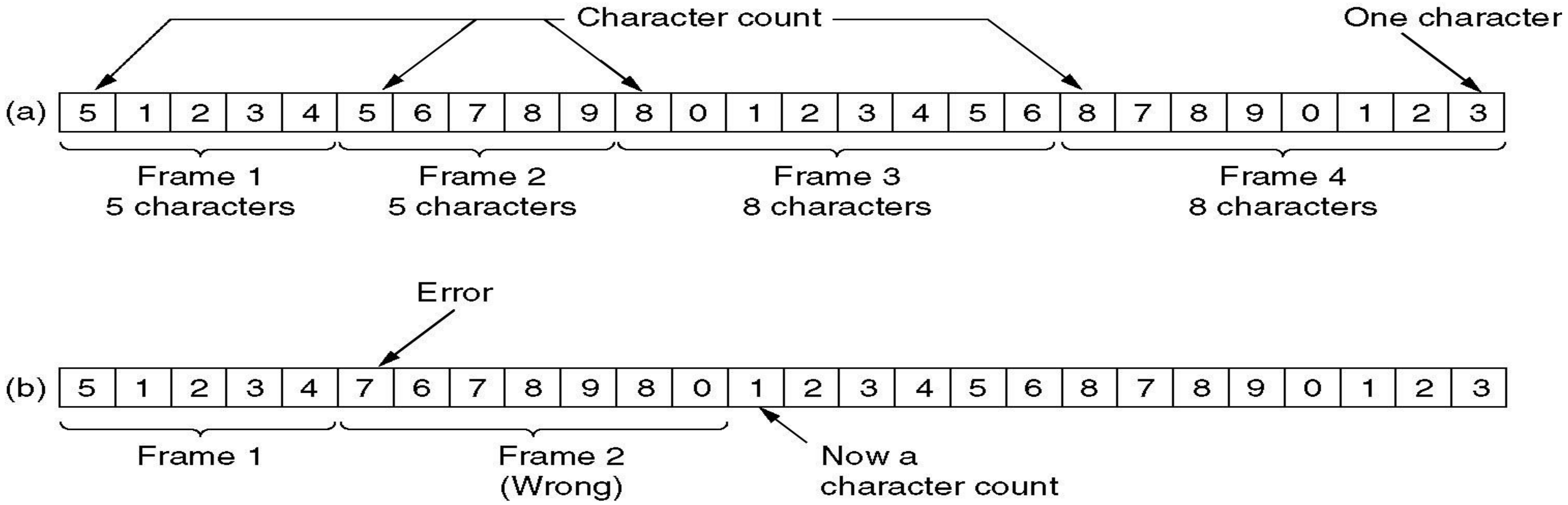
5	1	2	3	4	3	6	7	6	9	3	1	2	1
Frame 1					Frame 2			Frame 3					

With error:

5	1	2	3	4	6	6	7	6	9	3	1	2	1
Frame 1					Frame 2						F3	Frame 4	

Character/Byte Count

- Problem with byte count



Character count

One character

(a) 5 1 2 3 4 5 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3

Frame 1
5 characters

Frame 2
5 characters

Frame 3
8 characters

Frame 4
8 characters

Error

(b) 5 1 2 3 4 7 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3

Frame 1

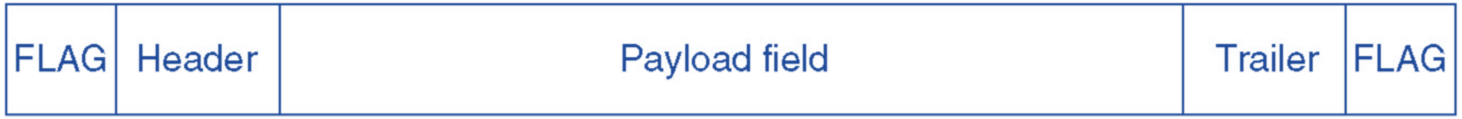
Frame 2
(Wrong)

Now a
character count

The trouble with this method is that the count can be garbled by a transmission error.

Parts of Frame

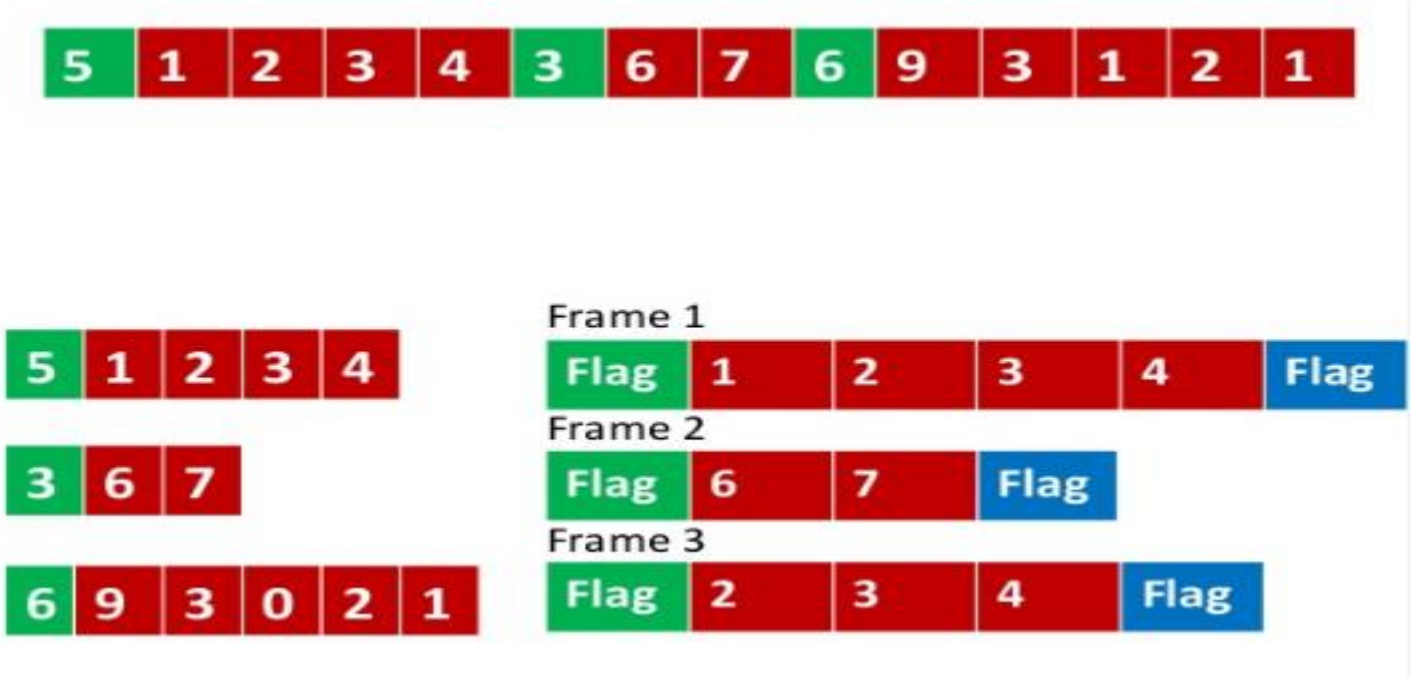
- **Frame Header** – It contains the source and the destination addresses of the frame.
- **Payload field** – It contains the message to be delivered.
- **Trailer** – It contains the error detection and error correction bits.
- **Flag** – It marks the beginning and end of the frame.



FLAG	Header	Payload field	Trailer	FLAG

Byte Stuffing (character oriented protocol)

- Insert FLAG byte before each frame staring and ending



5 1 2 3 4 3 6 7 6 9 3 1 2 1

5 1 2 3 4

3 6 7

6 9 3 0 2 1

Frame 1

Flag 1 2 3 4 Flag

Frame 2

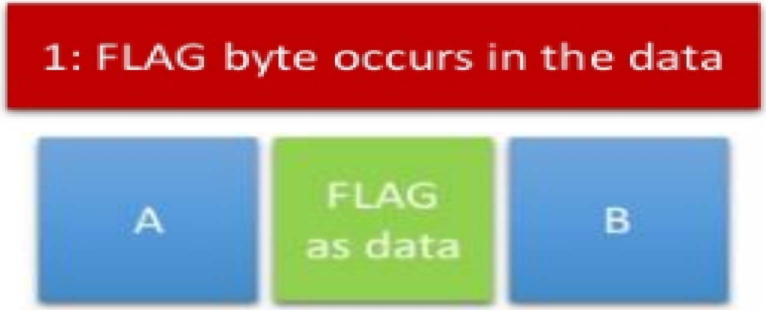
Flag 6 7 Flag

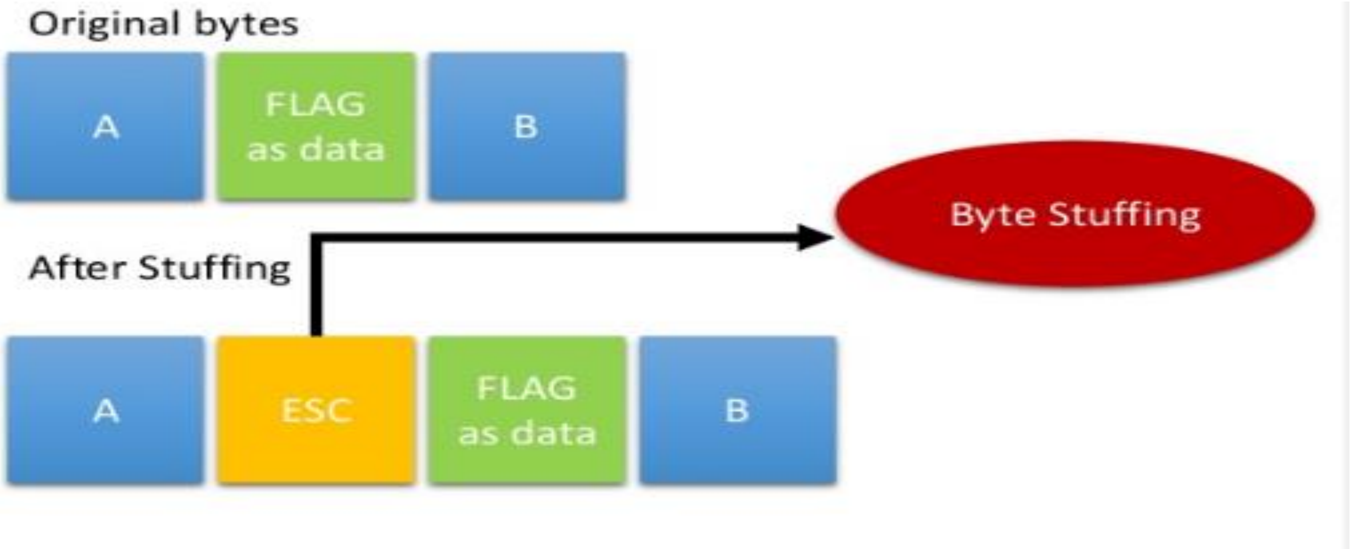
Frame 3

Flag 2 3 4 Flag

Byte Stuffing (character oriented protocol)

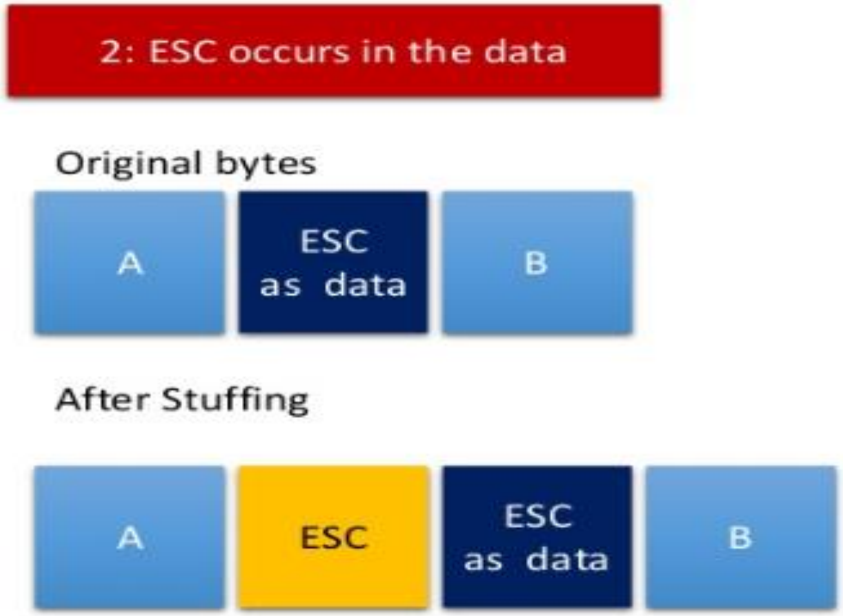
- Problem : FLAG byte is occur in data





Byte Stuffing the technique of inserting a special byte (ESC) just before each accidental flag byte in the data.

Byte Stuffing (character oriented protocol)



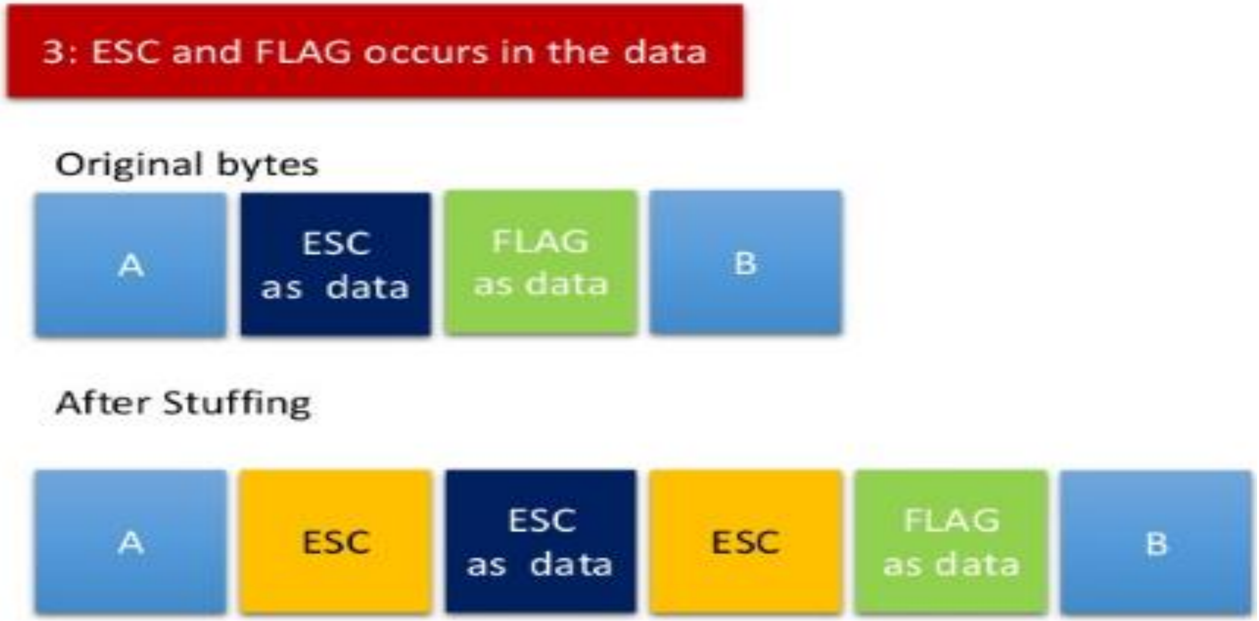
2: ESC occurs in the data

Original bytes

A | ESC as data | B

After Stuffing

A | ESC | ESC as data | B



3: ESC and FLAG occurs in the data

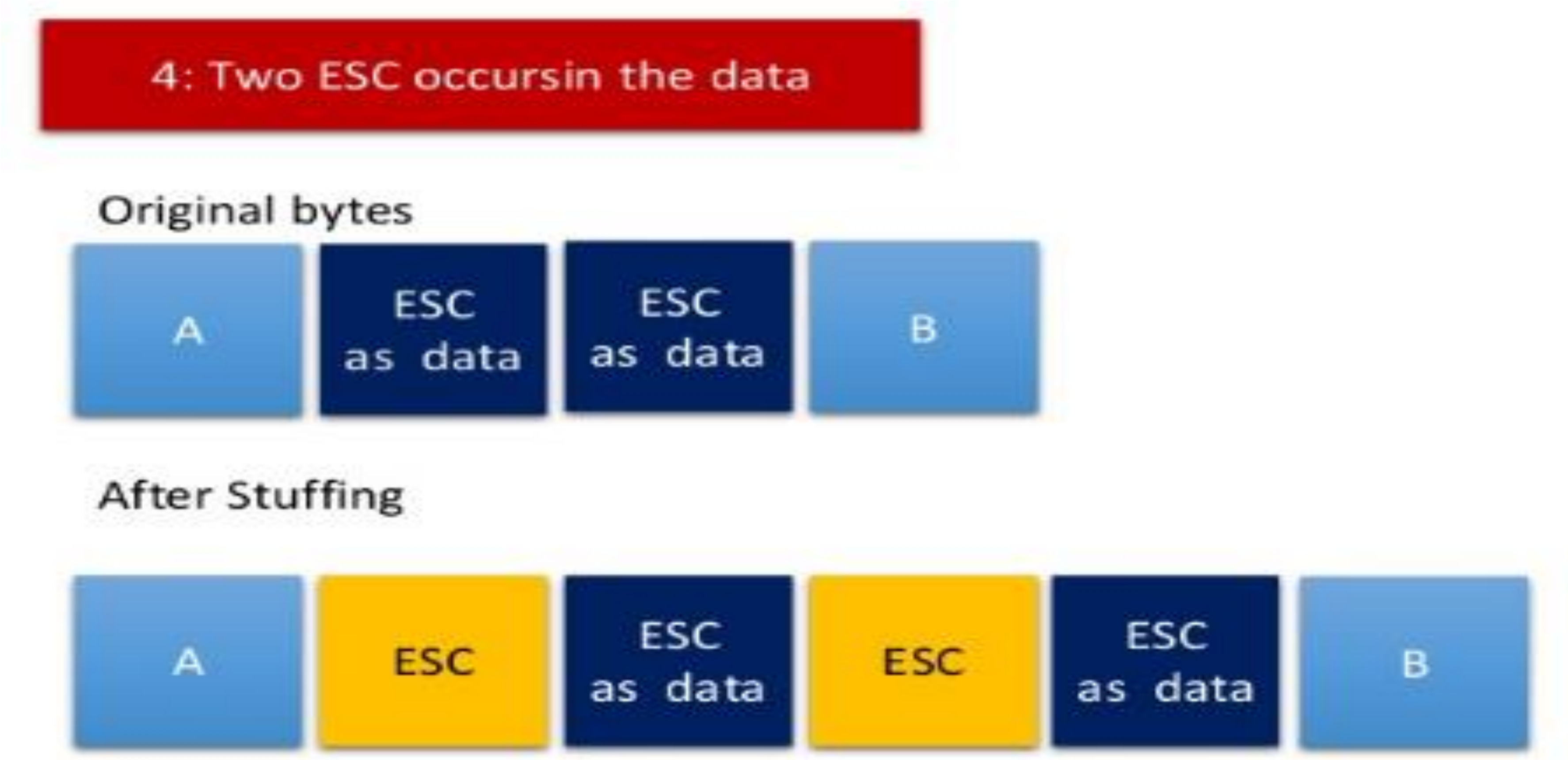
Original bytes

A | ESC as data | FLAG as data | B

After Stuffing

A | ESC | ESC as data | ESC | FLAG as data | B

Byte Stuffing (character oriented protocol)



4: Two ESC occursin the data

Original bytes

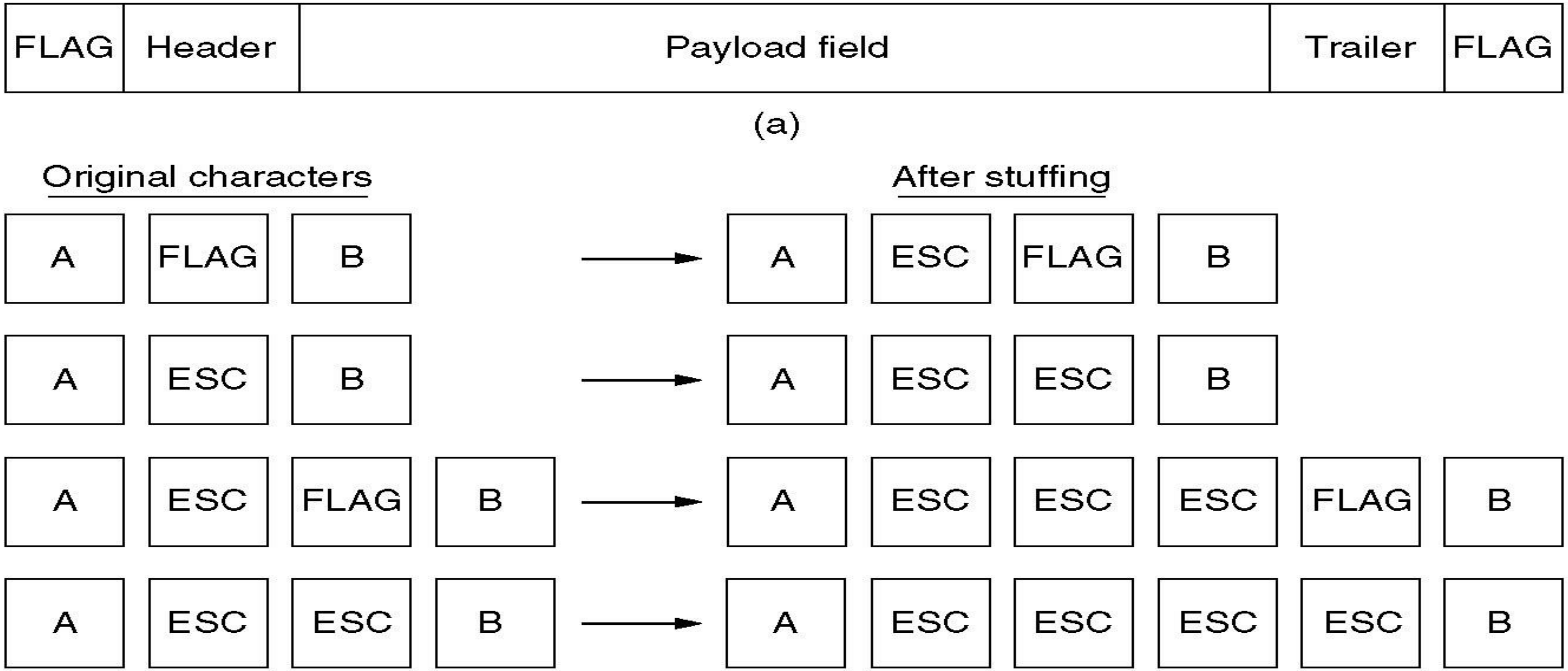
A | ESC as data | ESC as data | B

After Stuffing

A | ESC | ESC as data | ESC | ESC as data | B

Byte Stuffing (character oriented protocol)

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



FLAG Header Payload field Trailer FLAG

(a)

Original characters → After stuffing

A FLAG B → A ESC FLAG B

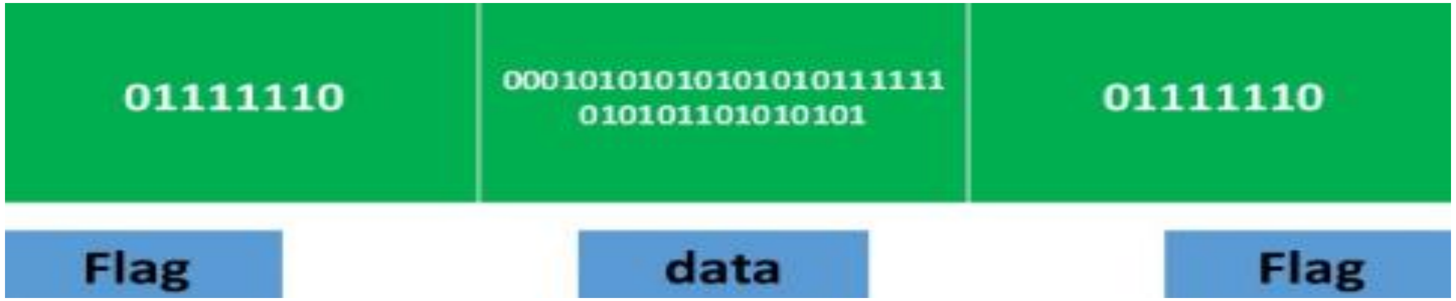
A ESC B → A ESC ESC B

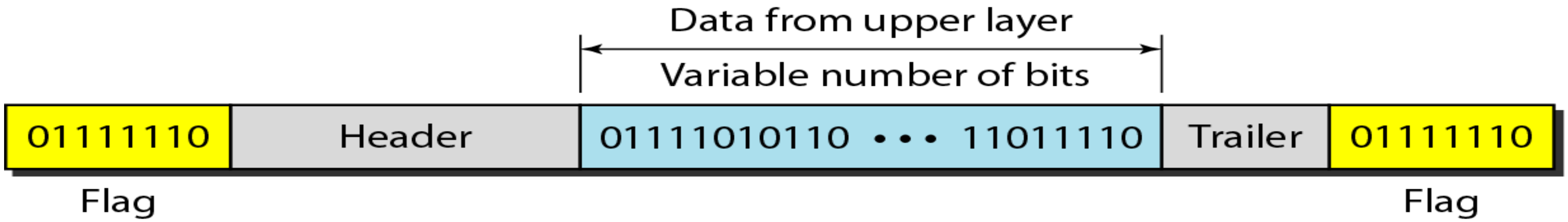
A ESC FLAG B → A ESC ESC ESC FLAG B

A ESC ESC B → A ESC ESC ESC ESC B

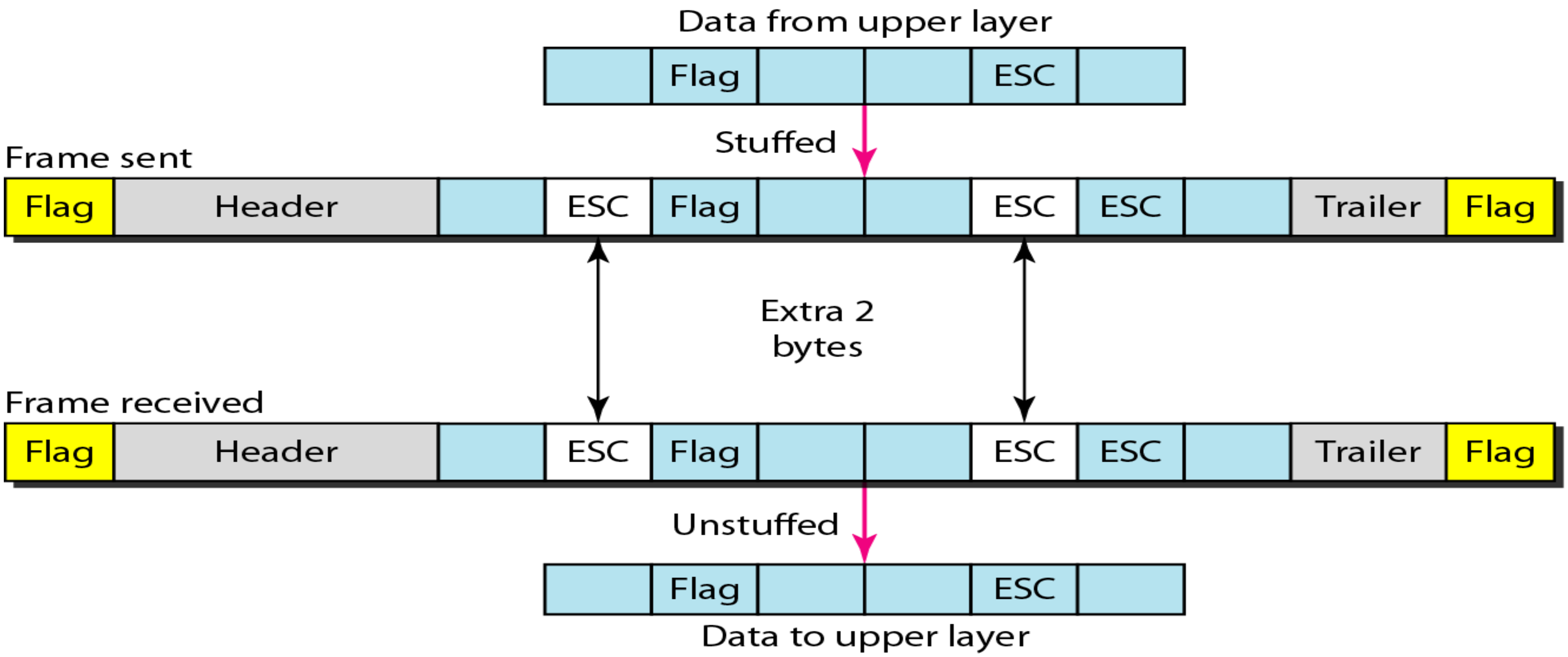
Byte Stuffing

- Uses a special 8- bit pattern flag 01111110 as the delimiter to define the beginning of the frame



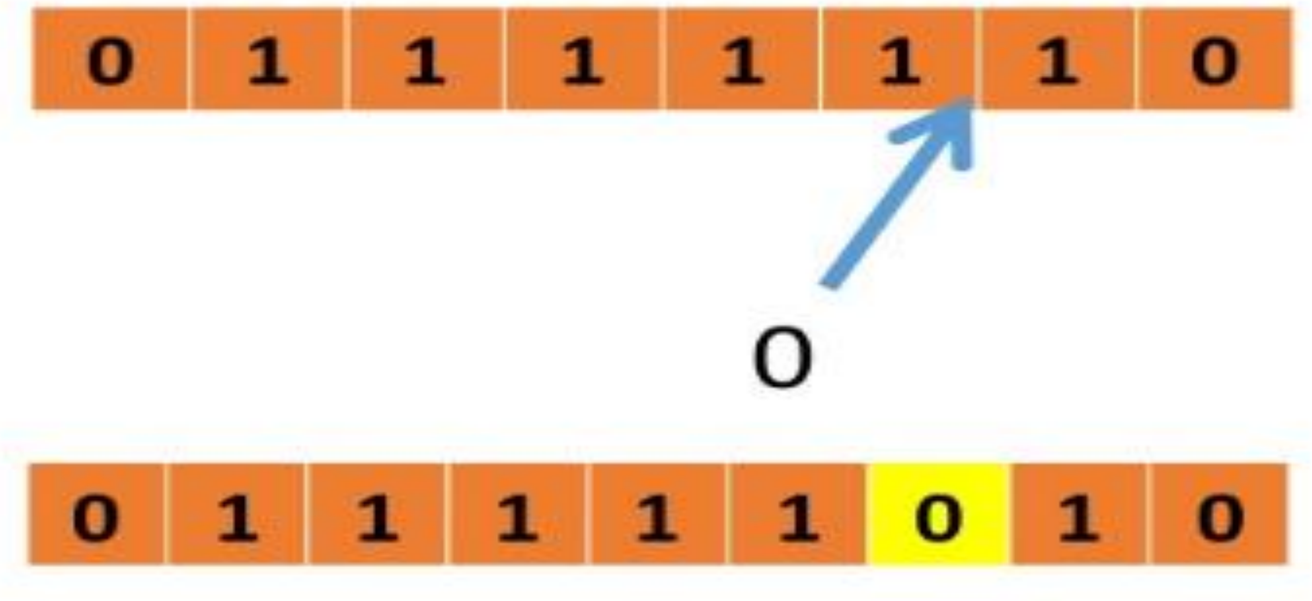


Byte Stuffing



Bit Stuffing (bit oriented protocol)

- 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 mistake the pattern 0111110 for a flag.

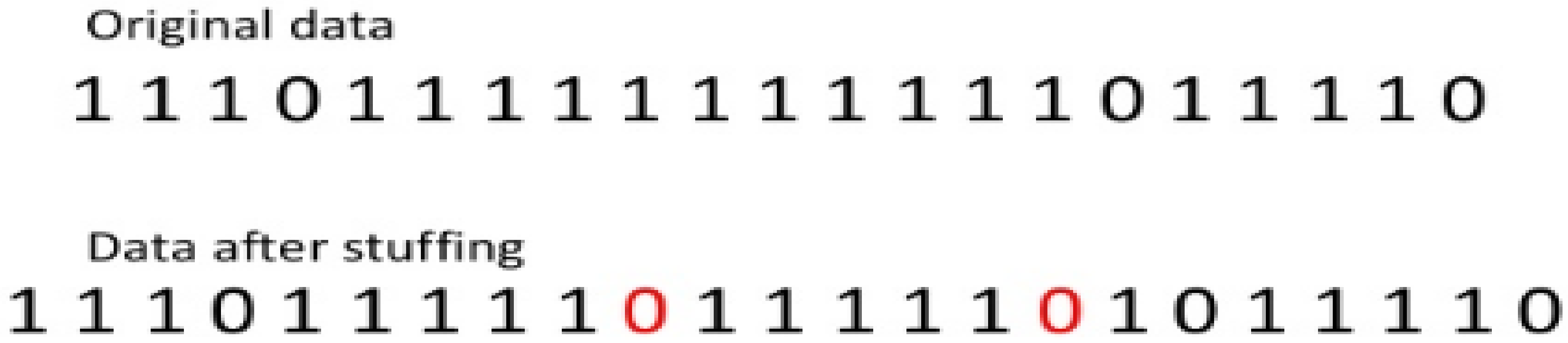


0 1 1 1 1 1 1 0

0

0 1 1 1 1 1 0 1 0

Bit Stuffing (bit oriented protocol)



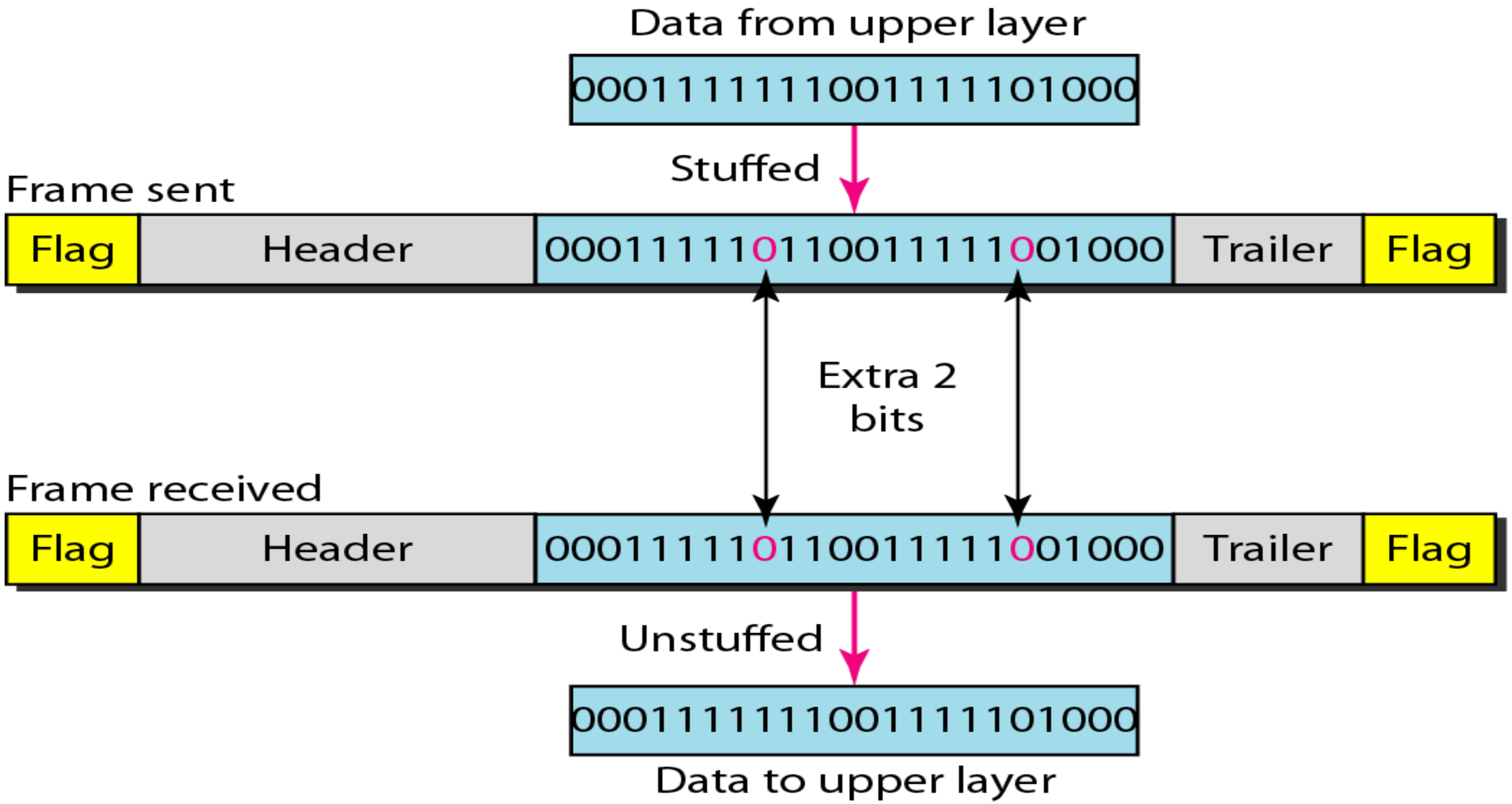
Original data

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

Data after stuffing

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

Bit Stuffing and unstuffing



Data from upper layer

0001111111001111101000

Stuffed

Frame sent

Flag	Header	000111110110011111001000	Trailer	Flag

Extra 2 bits

Frame received

Flag	Header	000111110110011111001000	Trailer	Flag

Unstuffed

0001111111001111101000

Data to upper layer

Physical layer coding violations

- It only applicable to network in which the encoding on the physical medium contains some redundancy
- Example
  - LANs encode 1 bit of data by using 2 physical bits.
  - Normally ,a 1 bit is high-low pair and 0 bit is a low-high pair
  - Easy for the receiver to locate the bit boundaries
  - High-High and Low-Low are not used for data but are used for delimiting frames in some protocols