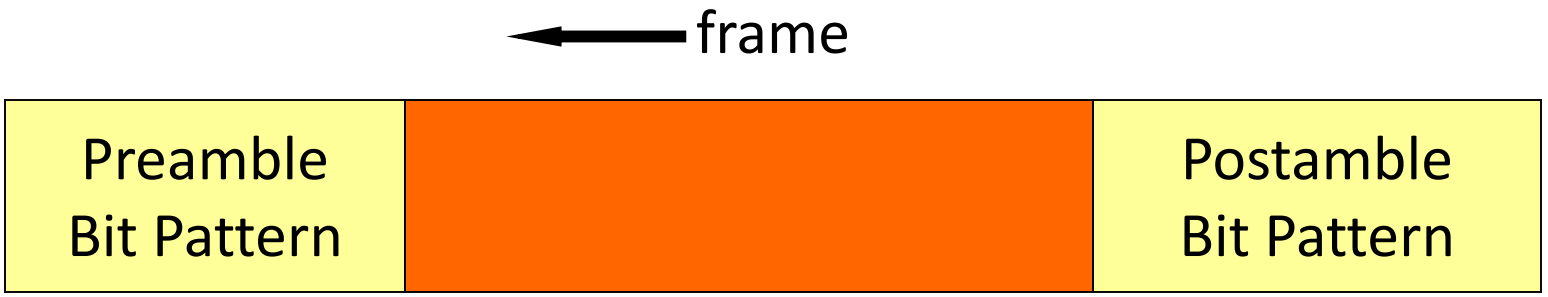
Bit and Byte Stuffing

Synchronous versus Asynchronous Transmissions

- There exists a hierarchy of synchronization tasks:
  - *Bit level* : recognizing the start and end of each bit
  - *Character or byte level* : recognizing the start and end of each character (or small unit of data)
  - *Block or message level* : recognize the start and end of each large unit of data (in networks this is a **frame**).

The contents of each frame are *encapsulated* between a pair of reserved characters or bytes for frame synchronization.



Byte Stuffing

[HDLC Example]

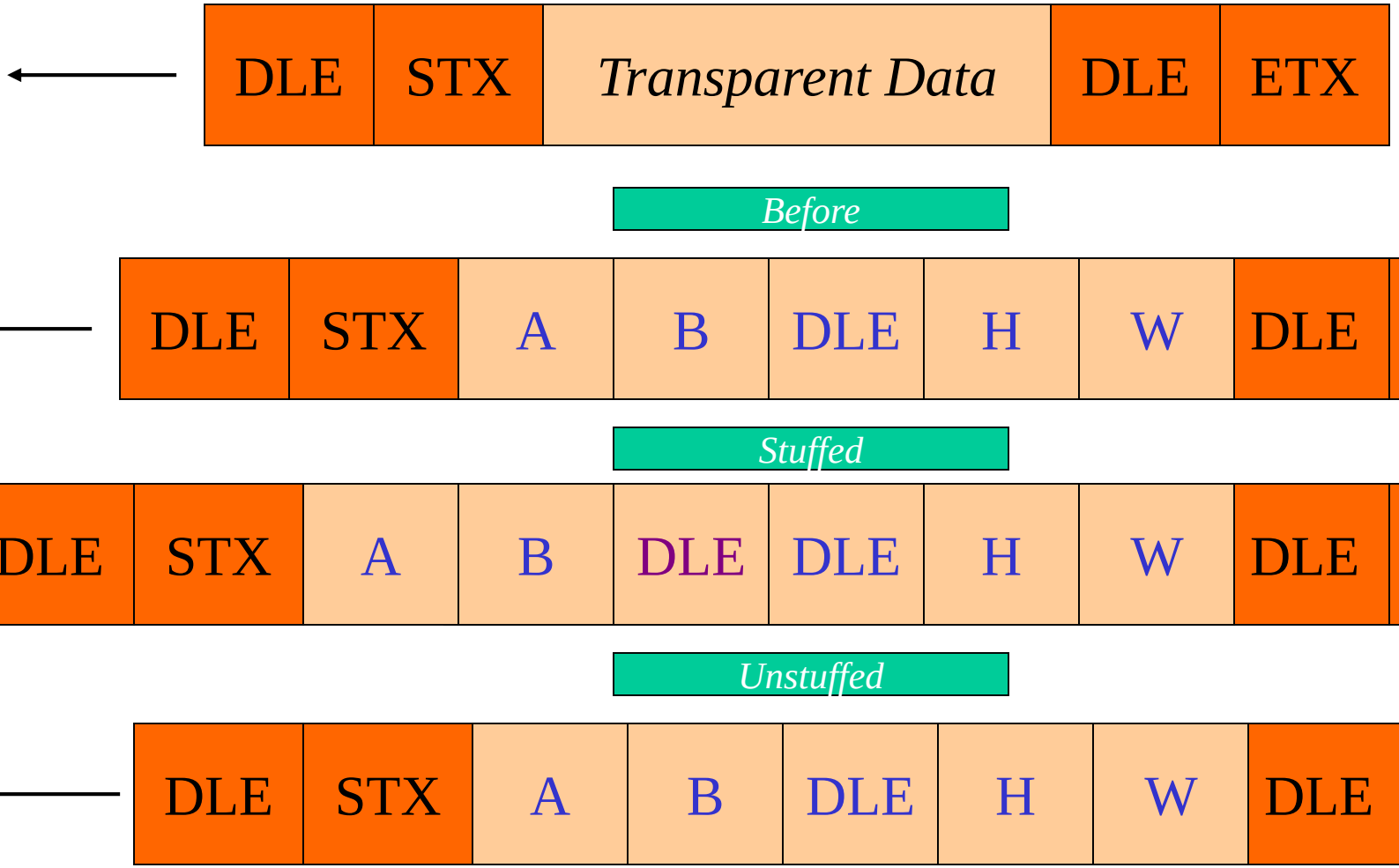
- *Also referred to as character stuffing.*
- ASCII characters are used as framing delimiters (e.g. DLE STX and DLE ETX)
- The problem occurs when these character patterns occur within the "transparent" data.

Solution: sender stuffs an **extra DLE** into the data stream just before each occurrence of an "accidental" DLE in the data stream.

The data link layer on the receiving end unstuffs the DLE before giving the data to the network layer.

Byte Stuffing

[HDLC Example]



Bit Stuffing

- Each frame begins and ends with a special bit pattern called a flag byte [01111110]. {Note this is **7E** in hex}
- Whenever sender data link layer encounters *five consecutive ones* in the data stream, it automatically stuffs a 0 bit into the outgoing stream.
- When the receiver sees *five consecutive incoming ones followed by a 0 bit*, it automatically destuffs the 0 bit before sending the data to the network layer.

Bit Stuffing