FRAMING

- The data link layer, needs to pack bits into frames, so that each frame is distinguishable from another.

- Although the whole message could be packed in one frame, that is not normally done. One reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message.

- When a message is divided into smaller frames, a single-bit error affects only that small frame.

- **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.

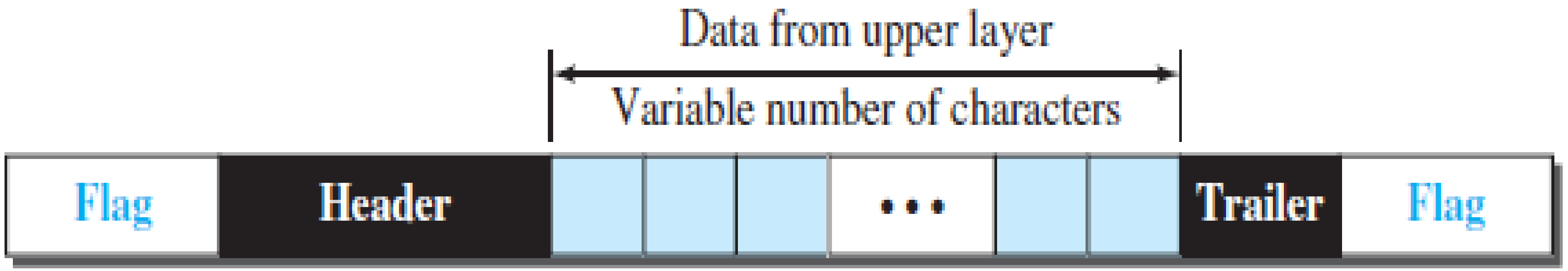
- **Variable-Size Framing**
  - In variable-size framing, we need a way to define the end of the frame and the beginning of the next. Historically, two approaches were used for this purpose: a character-oriented approach and a bit-oriented approach.

Break

Character-Oriented Protocols

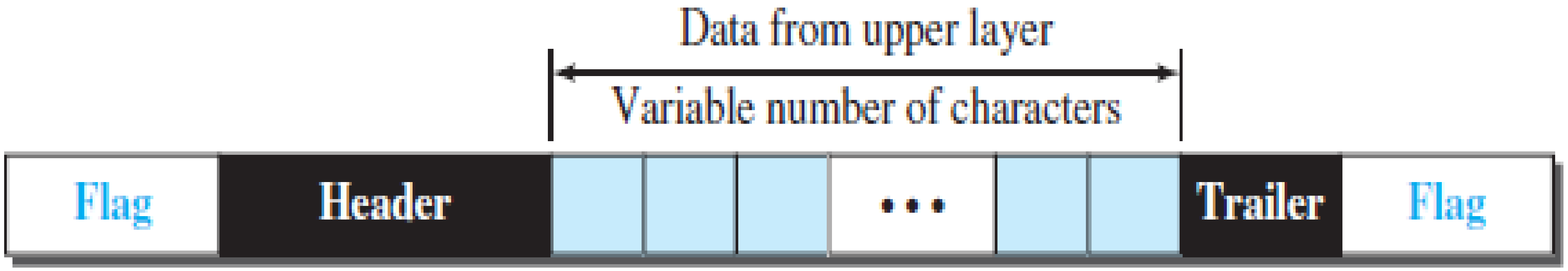
- In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII. The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are 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.



- Character-oriented framing was popular when only text was exchanged by the data link layers. The flag could be selected to be any character not used for text communication. Now, however, we send other types of information such as graphs, audio, and video. Any pattern used for the flag could also be part of the information.

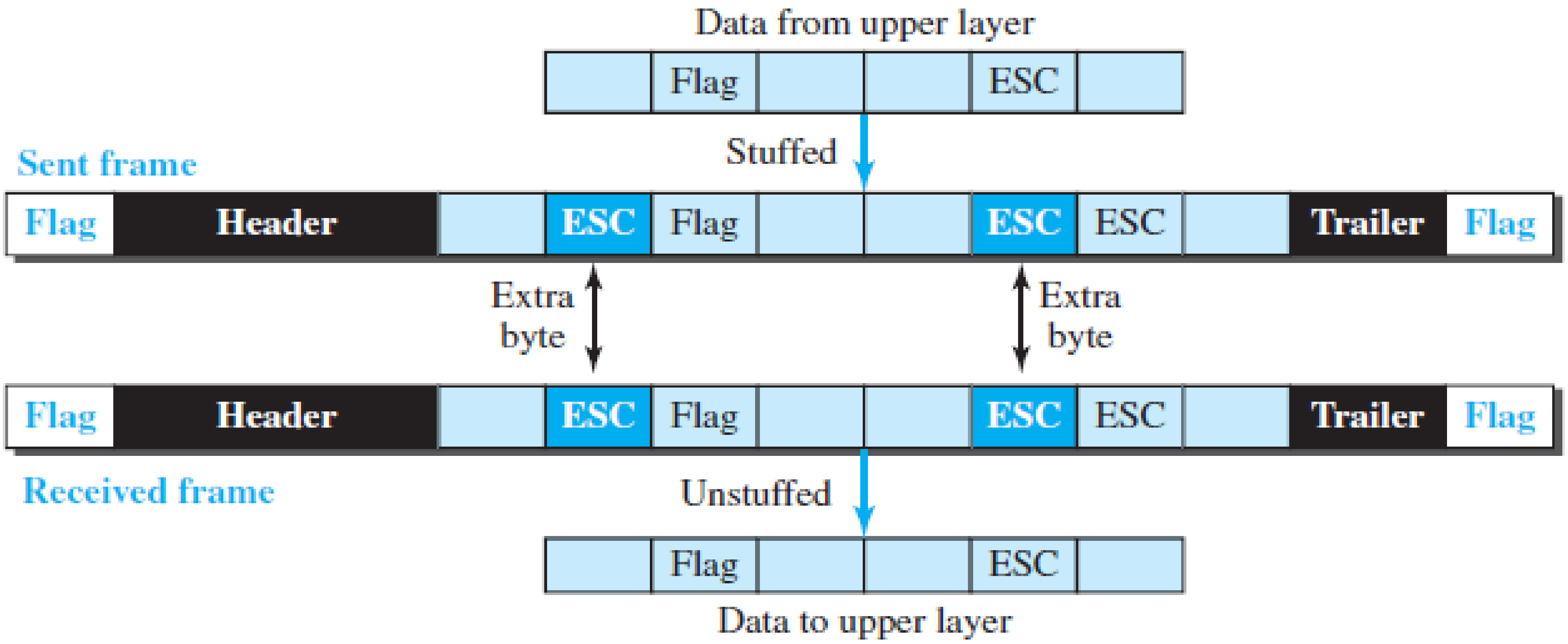
- If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame.



Break

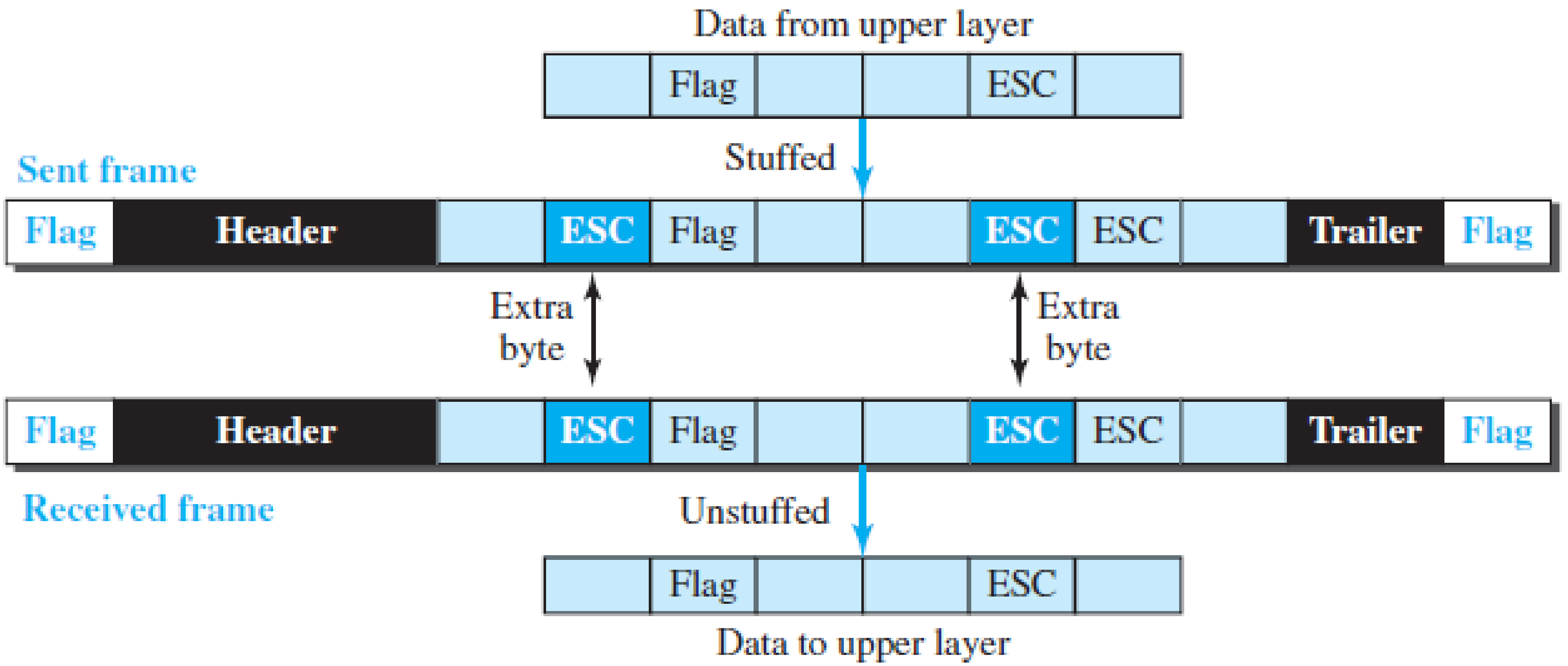
- To fix this problem, a byte-stuffing strategy was added to character-oriented framing. In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag.

- The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which 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 by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a flag? The receiver removes the escape character, but keeps the flag, which is incorrectly interpreted as the end of the frame.

- To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text.

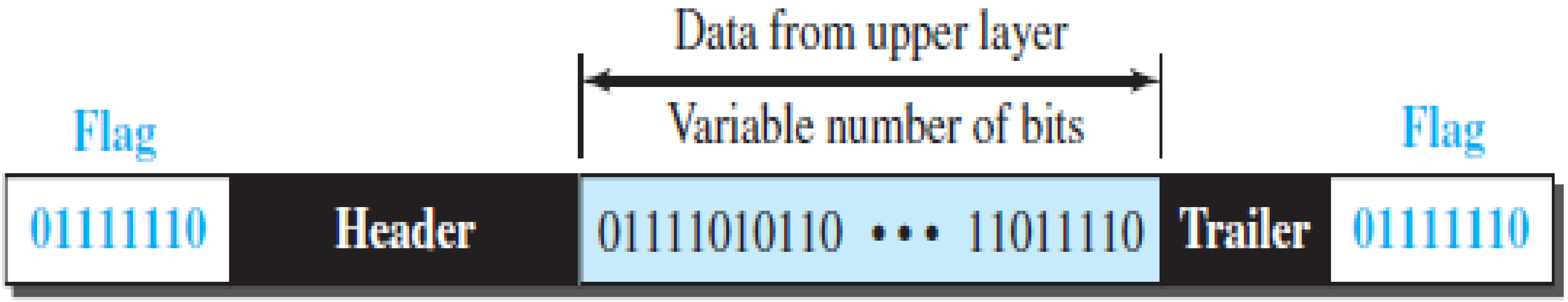


- Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters. We can say that in general, the tendency is moving toward the bit-oriented protocols that we discuss next.

Break

Bit-Oriented Protocols

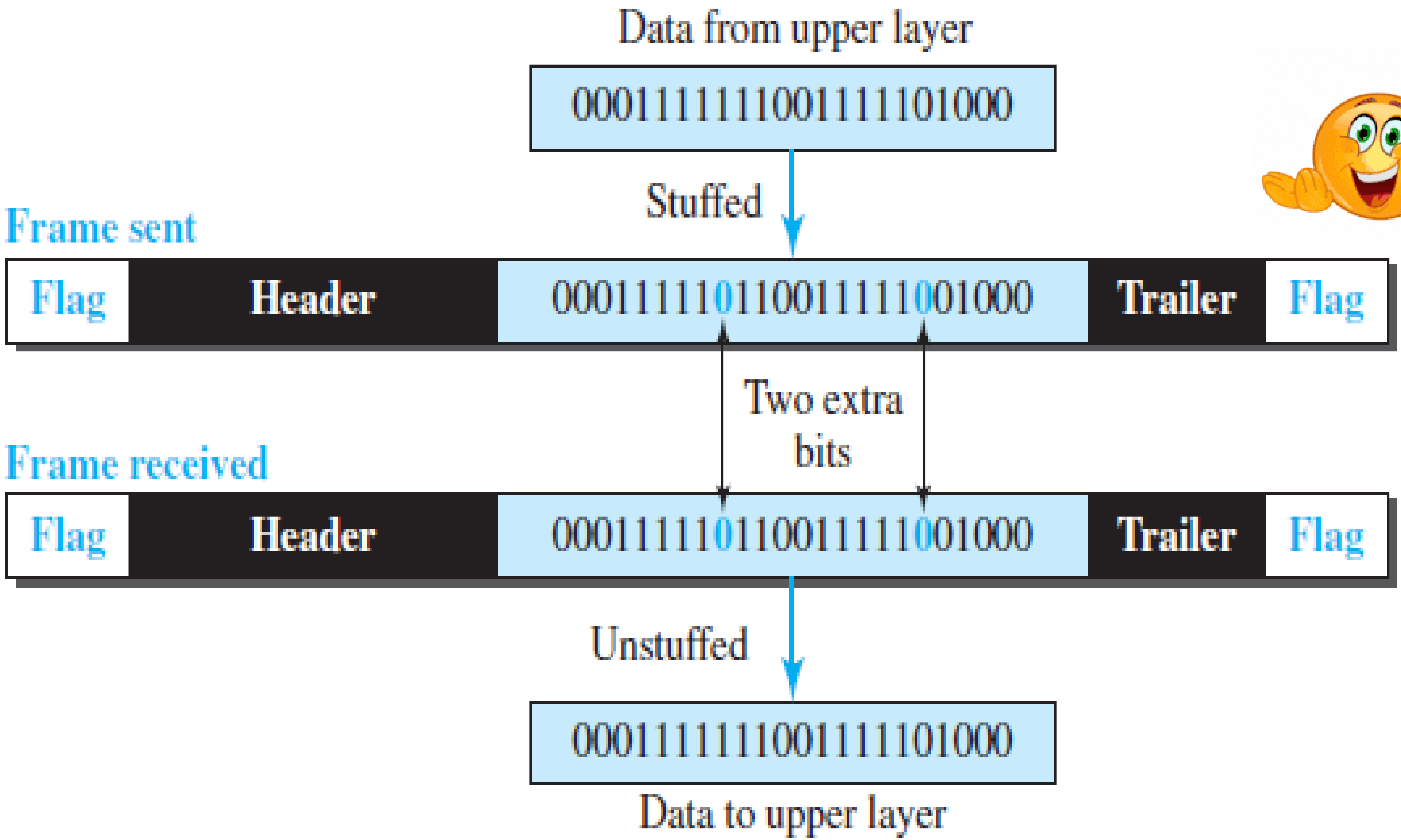
- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a 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 byte-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame.

- We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag. The strategy is called bit stuffing.

- In bit stuffing, if a 0 and five consecutive I bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.


Data from upper layer
0001111111001111101000
Stuffed
Frame sent
Flag
Header
000111110110011111001000
Trailer
Flag
Two extra bits
Frame received
Flag
Header
000111110110011111001000
Trailer
Flag
Unstuffed
0001111111001111101000
Data to upper layer

Q A bit-stuffing based framing protocol uses an 8-bit delimiter pattern of 01111110. If the output bit-string after stuffing is 01111100101, then the input bit-string is **(Gate-2014) (1 Marks)**

A) 0111110100

B) 0111110101

C) 0111111101

D) 0111111111

Q In a data link protocol, the frame delimiter flag is given by 0111. Assuming that bit stuffing is employed, the transmitter sends the data sequence 01110110 as **(Gate-2004) (2 Marks)**

(A) 01101011 **(B)** 011010110 **(C)** 011101100 **(D)** 0110101100