## Difference between TCP and UDP protocol:

1. TCP is a connection-oriented protocol. Example: HTTP, HTTPs, FTP, SMTP, Telnet
   UDP is a connectionless protocol. Example: DNS, DHCP, TFTP, SNMP, RIP, VOIP etc.

2. TCP is suited for applications that require high reliability, and transmission time is relatively less critical.
   UDP is suitable for applications that need fast, efficient transmission, such as games.

3. TCP rearranges data packets in the order specified.
   UDP has no inherent order as all packets are independent of each other. If ordering is required, it has to be managed by the application layer.

4. The speed for TCP is slower than UDP. There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.
   UDP is faster because there is no error-checking for packets. There is no guarantee that the messages or packets sent would reach at all.

5. TCP header size is 20 bytes, whereas UDP Header size is 8 bytes.

6. TCP is heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control.
   UDP is lightweight. There is no ordering of messages, no any tracking connections, etc.

7. TCP does error checking whereas UDP does error checking, but no recovery options.

---

## BYTE and BIT Stuffing Techniques:

While sending data over network, the data link layer divide into frames. Framing have several advantages than send raw very large data. It reduces the probability of error and reduces the amount of retransmission needed.

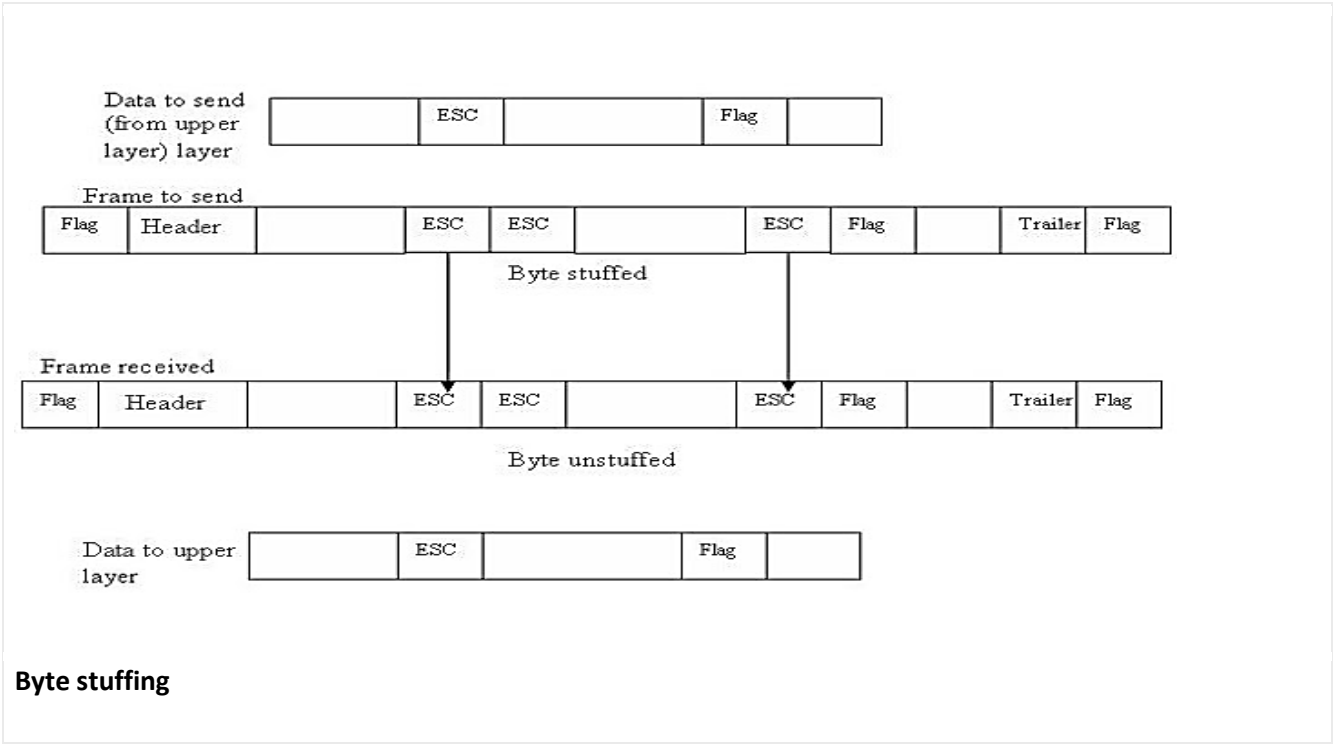
There exist two general methods for framing: **fixed size framing and variable size framing**. In fixed size framing, the data divided into fixed size frames and send over the transmission media. In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. ATM network use fixed size packets called cells.

In variable size framing, the data divided into variable size frames. Here the network system needs a mechanism to distinguish the end of a packet and beginning of another one. Two protocols are used for this purpose: **character oriented protocol** and **bit oriented protocol**.

**Character-Oriented Protocols**

In character-oriented protocol, we add special characters (called flag) to distinguish beginning and end of a frame. Usually flag has 8-bit length. The character-oriented protocols are popular only with text data. While using character–oriented protocol another problem is arises, pattern used for the flag may also part of the data to send. If this happens, the destination node, when it encounters this pattern in the middle of the data, assumes it has reached the end of the frame. To deal with this problem, a **byte stuffing** (**also known as character stuffing**) approach was included to character-oriented protocol. In byte stuffing a special byte is add to the data part, this is known as **escape character** (ESC). The escape characters have a predefined pattern. The receiver removes the escape character and keeps the data part. It cause to another problem, if the text contains

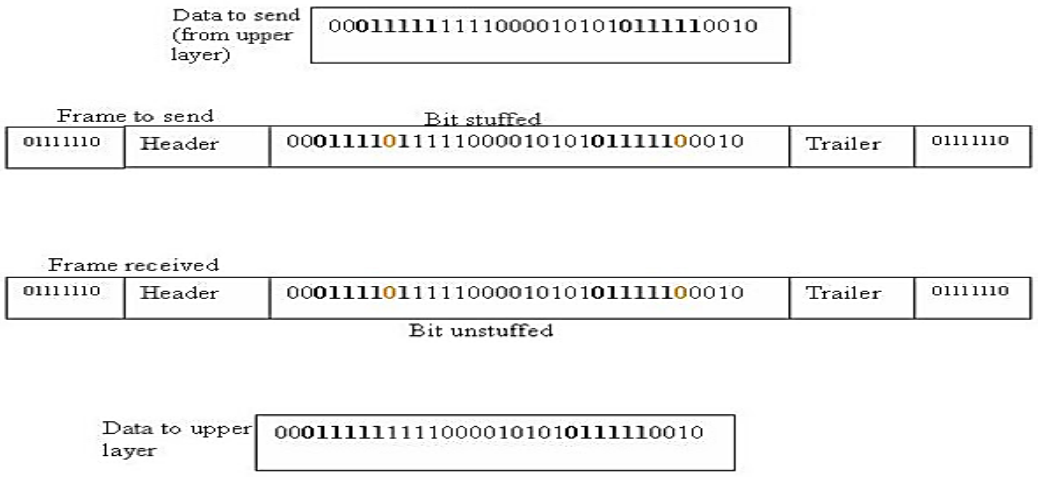
escape characters as part of data. To deal with this, an escape character is prefixed with another escape character. The following figure explains everything we discussed about character stuffing.




**Byte stuffing**

**Bit-Oriented Protocols**

In a bit-oriented protocol, the data to send is a series of bits. In order to distinguish frames, most protocols use a bit pattern of 8-bit length (01111110) as flag at the beginning and end of each frame. Here also cause the problem of appearance of flag in the data part to deal with this an extra bit added. This method is called **bit stuffing.** In bit stuffing, if a 0 and five successive 1 bits are encountered, an extra 0 is added. The receiver node removes the extra-added zero. This process is illustrate below,




**Bit stuffing**

**Simply, Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data. Byte stuffing is the method of adding 1 extra byte if there is a flag or escape character in the text.**