1 Objective(s)

- To implement the data link layer framing method Character stuffing.

2 Problem analysis

A technique where each frame starts with the ASCII character sequence DLE STX and ends with thesequence DLE ETX.(where DLE is Data Link Escape, STX is Start of TeXt and ETX is End of TeXt.) This method overcomes the drawbacks of the character count method. If the destination ever loses synchronization, it only has to look for DLE STX and DLE ETX characters. If however, binary data is being transmitted then there exists a possibility of the characters DLE STX and DLE ETX occurring in the data. Since this can interfere with the framing, a technique called character stuffing is used.

In character or byte stuffing; within the frame, replace every occurrence of DLE with the two-character sequence DLE DLE. The receiver reverses the processes, replacing every occurrence of DLE DLE with a single DLE before this data is given to the network layer.

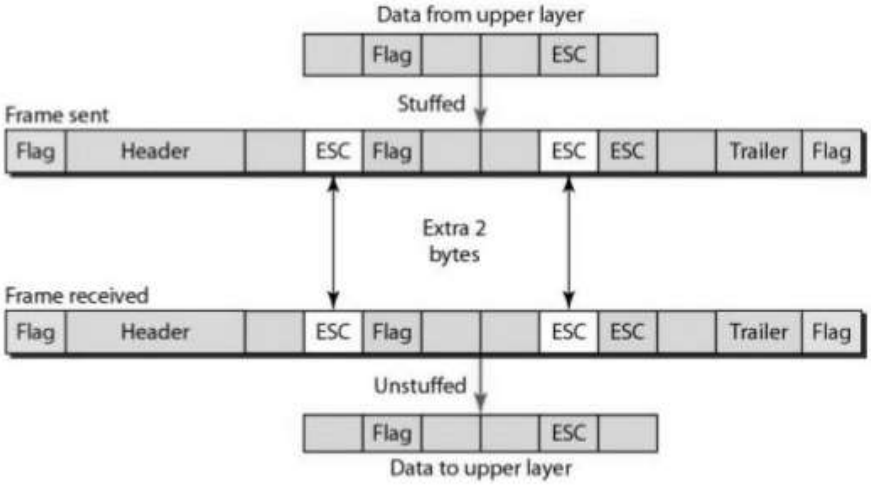

Figure 1: Character stuffing and de-stuffing in HDLC frame structure.

3 Algorithm

Algorithm 1: Character stuffing and de-stuffing

Input: Given a string of size N, flag character, escape character

- To stuff replace every occurrence of DLE with the two-character sequence DLE DLE.
- replace every occurrence of DLE DLE with a single DLE in de-stuffing.

4 Implementation in C

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i=0,j=0;
    char d[100],l[]="DLEETX",sd[100],ds[100];
    printf("Enter Data in UpperCase: ");
    scanf("%s",d);
    sd[0]='D', sd[1]='L', sd[2]='E',
    sd[3]='S',sd[4]='T',sd[5]='X';
    j=6;
    while(d[i]!='\0')
    {
        if(d[i]=='D'&& d[i+1]=='L'&& d[i+2]=='E')
        {
            sd[j]='D', sd[j+1]='L', sd[j+2]='E', sd[j+3]='D',
            sd[j+4]='L', sd[j+5]='E';
            j+=6;
            i+=3;
        }
        else
        sd[j++]=d[i++];
    }
    sd[j]='\0';
    strcpy(ds,sd);
    strcat(sd,l);
    printf("After Stuffing: ");
    printf("%s",sd);
    i=0;
    j=6;
    while(ds[j]!='\0')
    {
        if( ds[j]=='D'&&ds[j+1]=='L'&& ds[j+2]=='E'&&
        ds[j+3]=='D'&& ds[j+4]=='L'&& ds[j+5]=='E')
        {
            d[i]='D', d[i+1]='L',d[i+2]='E';
            j+=6;
            i+=3;
        }
        else
            d[i++]=ds[j++];
    }
    d[i]='\0';
    printf("\n\nAfter De-stuffing: ");
    printf("%s",d);
}
```

5 Input/Output (Compilation,Debugging & Testing)

Enter Data in UpperCase: UNITEDLEDSTATES

After Stuffing: DLESTX UNITEDLEDLEDSTATES DLEETX

After De-stuffing: DLESTX UNITEDLEDSTATES DLEETX

6 Discussion & Conclusion

Based on the focused objective(s) to understand about character stuffing and de-stuffing, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

7 Lab Task (Please implement yourself and show the output to the instructor)

- Implement the Byte (Character) stuffing and de-stuffing together where the system provides a choice to change the transmitted bit stream before de-stuffing. Consider the followings:

  A B C D E F F E F D F E G

  Where, F is the flag character and E is the escape character.

8 Lab Exercise (Submit as a report)

- Implement byte stuffing algorithm in C++ where the FLAG is "GALF" and the ESCAPE sequence is "EPACSE". The program should prompt the user to input data and perform byte stuffing according to the specified protocol, inserting the ESCAPE sequence before any occurrence of the FLAG or ESCAPE sequence. After the byte stuffing process, the program should display the resulting stuffed data.
  - ***Input- This is some data GALF and EPACSE another GALF example EPACSE data***
  - ***Output- GALFThis is some data EPACSEGALF and EPACSEEPACSE another EPACSEGALF example EPACSEEPACSE dataGALF***

9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.