EXPERIMENT 5

Aim: Write a program to implement various types of framing methods.

Theory:

Framing

Since the physical layer merely accepts and transmits a stream of bits without any regard to meaning or structure, it is upto the data link layer to create and recognize frame boundaries. This can be accomplished by attaching special bit patterns to the beginning and end of the frame. If these bit patterns can accidentally occur in data, special care must be taken to make sure these patterns are not incorrectly interpreted as frame delimiters.

The three framing methods that are widely used are

- Character count
- Starting and ending characters, with character stuffing
- Starting and ending flags, with bit stuffing

Character Count

This method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow, and hence where the end of the frame is. The disadvantage is that if the count is garbled by a transmission error, the destination will lose synchronization and will be unable to locate the start of the next frame. So, this method is rarely used.

Character stuffing

In the second method, each frame starts with the ASCII character sequence DLE STX and ends with the sequence 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. The sender's data link layer inserts an ASCII DLE character just before the DLE character in the data. The receiver's data link layer removes this DLE before this data is given to the network layer. However character stuffing is closely associated with 8-bit characters and this is a major hurdle in transmitting arbitrary sized characters.

Bit stuffing

The third method allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character. At the start and end of each frame is a flag byte consisting of the special bit pattern 01111110. Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a zero bit into the outgoing bit stream. This technique is called bit stuffing. When the receiver sees five consecutive 1s in the incoming data stream, followed by a zero bit, it automatically destuffs the 0 bit. The boundary between two frames can be determined by locating the flag pattern.

Program: 1. Character Count

1. **SENDER SIDE CODE**

```
#include<stdio.h>
#include<fcntl.h>
#include<string.h>
void main()
{
 int no,j=0,len,i,k,pid;
 char data[50],frame[100];
 system("clear");

 system(">pipe");
 pid=open("pipe",O_WRONLY);
```

```
    printf("Enter how many data want to enter: ");
    scanf("%d",&no);
    for(i=0;i<no;i++)
    {
     printf("\nEnter data : ");
     scanf("%s",data);
     len=strlen(data);
     frame[j]=len+1+48;
     for(k=0;k<len;k++)
     {
      j++;
      frame[j]=data[k];
     }
     j++;
    }
    frame[j]='\0';
    printf("\nFRAME TO SEND : %s",frame);

    write(pid,&frame,sizeof(frame));
   }
```

2. RECEIVER SIDE CODE

```
#include<stdio.h>
#include<fcntl.h>
#include<string.h>
void main()
{
 int no,j=0,len,i=0,k,pid;
 char data[50],frame[100];
 system("clear");

 pid=open("pipe",O_RDONLY);
 read(pid,&frame,sizeof(frame));

 while(frame[i]!='\0')
 {
  len=frame[i]-48;
  i++;
  for(k=0;k<len-1;k++)
  {
   data[k]=frame[i];
   i++;
  }
  data[k]='\0';
  printf("\ndata %s",data);
 }
}
```

Program: 2.Character Stuffing

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<process.h>
void main()
{
int i=0,j=0,n,pos;
char a[20],b[50],ch;
clrscr();
printf("enter string\n");
scanf("%s",&a);
n=strlen(a);
printf("enter position\n");
scanf("%d",&pos);
if(pos>n)
{
printf("invalid position, Enter again :");
scanf("%d",&pos);
}
printf("enter the character\n");
ch=getche();

b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
if(i==pos-1)
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]=ch;
b[j+4]='d';
b[j+5]='l';
b[j+6]='e';
j=j+7;
}
if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
j=j+3;
}
```

```
b[j]=a[i];
i++;
j++;
}
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]='e';
b[j+4]='t';
b[j+5]='x';
b[j+6]='\0';
printf("\nframe after stuffing:\n");
printf("%s",b);
getch();
}
```