[2CEIT503 COMPUTER NETWORKS]

Practical: 4

AIM- Write a program to implement various framing techniques.

a. Bit Stuffing

b. Byte Stuffing

Submitted By: Malay Patel
Enrollment number: 21172012015

Department of Computer Engineering/ Information Technology

Practical-4

Bit stuffing:

- Allows frame to contain arbitrary number of bits and arbitrary character size. The frames are separated by separating flag.
- Each frame begins and ends with a special bit pattern, 01111110 called a flag byte. When five consecutive l's are encountered in the data, it automatically stuffs a '0' bit into outgoing bit stream.
- In this method, frames contain an arbitrary number of bits and allow character codes with an arbitrary number of bits per character. In his case, each frame starts and ends with a special bit pattern, 01111110.
- In the data a 0 bit is automatically stuffed into the outgoing bit stream whenever the sender's data link layer finds five consecutive 1s.
- This bit stuffing is similar to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data.
- When the receiver sees five consecutive incoming i bits, followed by a o bit, it automatically destuffs (i.e., deletes) the 0 bit. Bit Stuffing is completely transparent to network layer as byte stuffing. The figure1 below gives an example of bit stuffing.
- This method of framing finds its application in networks in which the change of data into code on the physical medium contains some repeated or duplicate data. For example, some LANs encodes bit of data by using 2 physical bits.

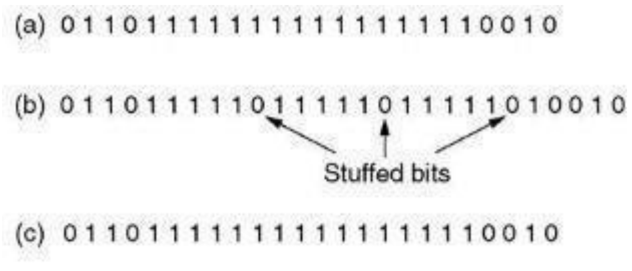

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0
(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0
Stuffed bits
(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Fig1: Bit stuffing

BYTE STUFFING:

- n this method, start and end of frame are recognized with the help of flag bytes. Each frames starts with and ends with a flag byte. Two consecutive flag bytes indicate the end of one frame and start of the next one. The flag bytes used in the figure 2 used is named as “ESC” flag byte.

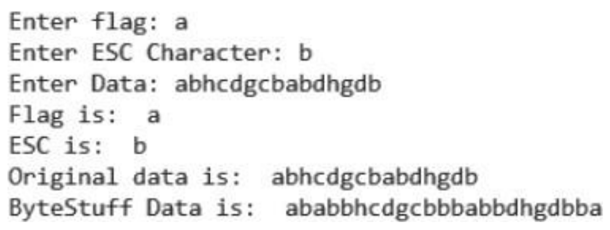
Practical-4

- A frame delimited by flag bytes. This framing method is only applicable in 8-bit character codes which are a major disadvantage of this method as not all charactercodes use 8-bit characters e.g. Unicode.
- Four example of byte sequences before and after stuffing:

Q.1 Byte Stuffing

```
flag=input("Enter flag: ")
esc_char=input("Enter ESC Character: ")
data=input("Enter Data: ")
byte_stuff=[]
byte_stuff.append(flag)
for i in data:
    if(i==flag or i==esc_char):
        byte_stuff.append(esc_char)
    byte_stuff.append(i)
byte_stuff.append(flag)
print("Flag is: ",flag)
print("ESC is: ",esc_char)
print("Original data is: ",data)
print("ByteStuff Data is: ","".join(byte_stuff))
```

Output:



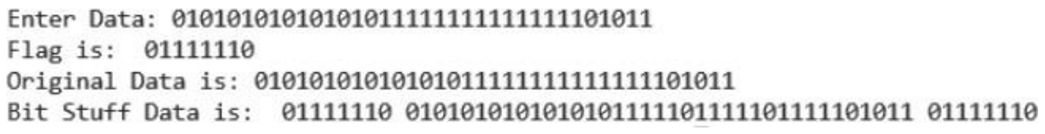
```
Enter flag: a
Enter ESC Character: b
Enter Data: abhcdgcbabdhgdb
Flag is:  a
ESC is:  b
Original data is:  abhcdgcbabdhgdb
ByteStuff Data is:  ababbhcdgcbbbabbdhgdbba
```

Practical-4

Q.2 Bit Stuffing

```
flag='01111110'
data_list = list(input("Enter Data: "))
c=0
index=0
print("Flag is: ",flag)
print("Original Data is: "+"".join(data_list))
while (index<len(data_list)):
    if(index<len(data_list)):
        if(data_list[index]=='1' and index==0):
            while (index<len(data_list)):
                if(data_list[index]=='1'):
                    index=index+1
                else:
                    break
    if(index<len(data_list)):
        if(data_list[index]=='0'):
            c=0
            index=index+1
        elif(data_list[index]=='1'):
            c=c+1
            index=index+1
        if(c==6):
            data_list.insert(index-1,'0')
            c=0
print("Bit Stuff Data is: ",flag,"".join(data_list),flag)
```

Output:



```
Enter Data: 0101010101010101111111111111111101011
Flag is:  01111110
Original Data is: 0101010101010101111111111111111101011
Bit Stuff Data is:  01111110 010101010101010101111101111101111101011 01111110
```