Sayan Chatterjee
2K19/CO/348

Experiment- 3

Aim:

Write a program which will take input data in byte sequence for byte stuffing and bit sequence for bit stuffing which will be containing some random data where you have to insert escape sequence for byte stuffing and 0 after five 1's in bit stuffing.

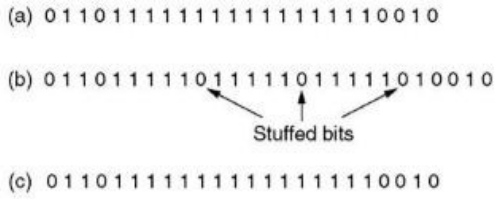
Language Used:

C++

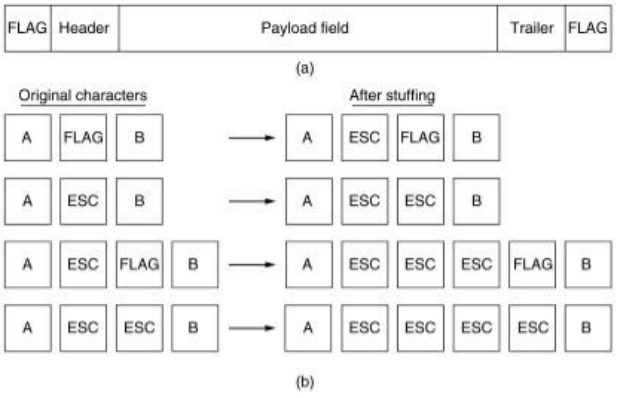
Theory:

In Data Link layer, the stream of bits from physical layer are divided into data frames. The data frames can be of fixed length or variable length. In variable – length framing, the size of each frame to be transmitted may be different. So, a pattern of bits is used as a delimiter to mark the end of one frame and the beginning of the next frame. To separate one frame from the next, an 8-bit (or 1-byte) flag is added at the beginning and the end of a frame. But the problem with that is, any pattern used for the flag could also be part of the information. There are two ways to overcome this problem:

1. **Bit stuffing:** Each frame begins and ends with a special bit pattern, 01111110 (7E in hexadecimal notation) called a flag byte. When five consecutive 1's are encountered in the data, it automatically stuffs a '0' bit into outgoing bit stream. When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e., deletes) the 0 bit.



2. **Byte stuffing:** If the pattern of the flag byte is present in the message byte, a special byte called the escape character (ESC) is stuffed before every byte in the message with the same pattern as the flag byte. If the ESC sequence is found in the message byte, then another ESC byte is stuffed before it.
The escape byte is 01111101 (7D in hexadecimal notation) in the Point-to-Point Protocol (PPP), and is stuffed before every byte that contains the same byte as the flag byte or the escape byte. The receiver on receiving the message removes the escape byte before passing it onto the network layer.



Code:

```
#include <bits/stdc++.h>
using namespace std;

#define sz(x) (int)x.size()

vector<string> byteSeq;

void bitStuffed(string& data)
{
   int cnt = 0;
   for (int i = 0; i < sz(data); i++){
      if (data[i] == '0'){        //reset the counter
         cnt = 0;
         continue;
      }
      ++cnt;
      if (cnt == 5){
         data.insert(i + 1, "0");
         cnt = 0;
      }
   }
}

void byteStuffed()
{
   int i = 0;
   while(i < sz(byteSeq)){
         string byte = byteSeq[i];
         if (byte == "7D" or byte == "7E"){
                  byteSeq.insert(byteSeq.begin() + i, "7D");
                  i += 2;
         }
         else i++;
   }
}

int main()
{
   int choice;
   cout << "1. Bit stuffing \n2. Byte stuffing\n\n";
   cout << "Enter your choice: "; cin >> choice;
   cin.ignore();

   string data;

   switch(choice){
      case 1: {
         cout << "\nEnter bit sequence: "; cin >> data;
```

```
            bitStuffed(data);
            cout << "Bit-stuffed data: " << data << endl;
               break;
            }

         case 2: {
            cout << "\nEnter space-separated byte sequence: ";
            getline(cin, data);

               string byte;
               for (char c: data){
                  if (c == ' ') {
                        byteSeq.push_back(byte);
                        byte.clear();
                     }
                     else byte += c;
               }
               byteSeq.push_back(byte);

               byteStuffed();
            cout << "Byte-stuffed data: ";
            for (string s: byteSeq) cout << s << ' ';
            cout << endl;
               break;
            }

         default: cout << "Invalid input!";
      }
      return 0;
   }
```

Output:

```
1. Bit stuffing
2. Byte stuffing

Enter your choice: 1

Enter bit sequence: 1100110101010101111101010110111111010101
Bit-stuffed data: 1100110101010101111100101011011111010101

--------------------------------
Process exited after 126.2 seconds with return value 0
Press any key to continue . . .
```

```
1. Bit stuffing
2. Byte stuffing

Enter your choice: 2

Enter space-separated byte sequence: 7E 7D 33 7E 42 7E
Byte-stuffed data: 7D 7E 7D 7D 33 7D 7E 42 7D 7E

--------------------------------
Process exited after 3.292 seconds with return value 0
Press any key to continue . . .
```