Practical 3-
Bit/Byte Stuffing
Implementation in Python
Computer Networks

18bce245

Tirth R. Patel

Activities Done:

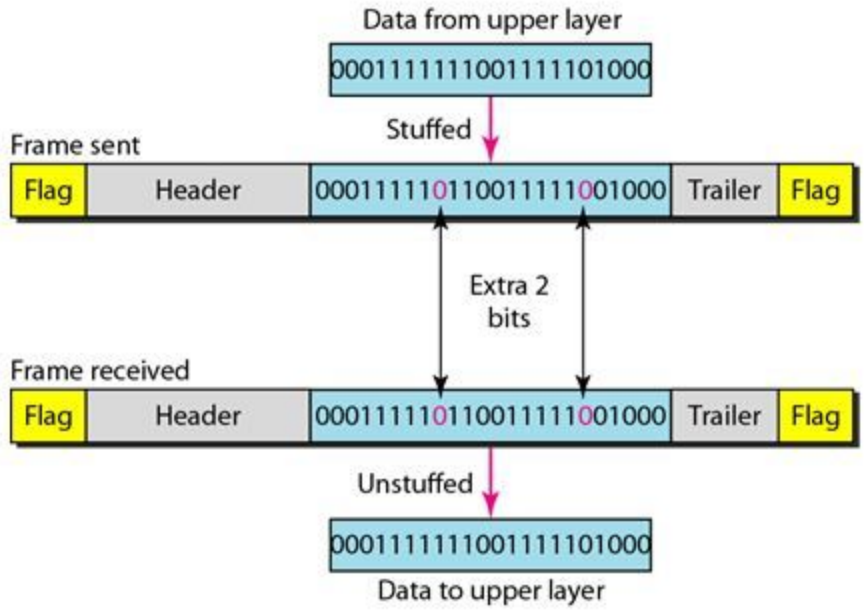
- Bit Stuffing
- Byte Stuffing

INTRODUCTION

The data link layer is responsible for something called Framing, which is the division of stream of bits from the network layer into manageable units (**called frames**). Frames could be of fixed size or variable size. The main purpose behind this idea is to have a flow control and synchronization between the sender and receiver.

Bit Stuffing

Bit stuffing is the process of inserting noninformation **bits** into data to break up **bit** patterns to affect the synchronous transmission of information. It is widely **used** in network and communication protocols, in which **bit stuffing** is a required part of the transmission process.

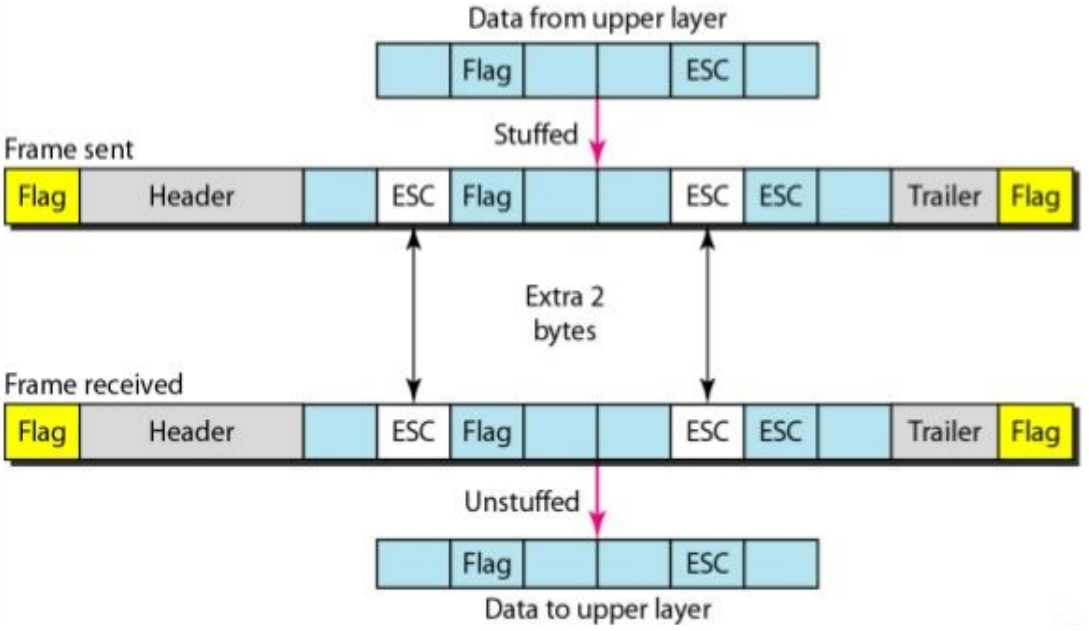


src-http://www.myreadingroom.co.in/notes-and-studymaterial/68-dcn/808-framing-and-framing-protocols.html

Byte Stuffing

A byte (usually escape character(ESC)), which has a predefined bit pattern is added to the data section of the frame when there is a character with the same pattern as the flag. Whenever the receiver encounters the ESC character, it removes from the data section and treats the next character as data, not a flag.

But the problem arises when the text contains one or more escape characters followed by a flag. To solve this problem, the escape characters that are part of the text are marked by another escape character i.e., if the escape character is part of the text, an extra one is added to show that the second one is part of the text.



src-http://www.myreadingroom.co.in/notes-and-studymaterial/68-dcn/808-framing-and-framing-protocols.html

Applications:

1. Synchronization
2. Limit the number of consecutive bits of the same value in the data.
3. Flow Control

My Implementation in Python

I have implemented bit/byte stuffing using a python programming language. Following is the structure of the Whole Code.

- **Sender Class**
  - def frames (this function used to make frames of random size.)
  - def byte_stuffing (this function is used to add a flag and escape character at appropriate places.)
  - def bit_conversion (this function is used to convert the byte to bit stream)
  - def bit_stuffing (this function is used to add '0' after 5 consecutive '1')

- **Receiver Class**
  - def bit_decoding (this function is used to decode the packet received by removing appropriate character)
  - def byte_decoding (this function is used to decode the packet by removing the flag and escape character)
  - def byte_conversion (this function is used to convert bit to byte stream)
  - def BinaryToDecimal (helper function for byte conversion)

Steps to be followed:

1. Extract all files from Zip File. It includes - python file and input.txt
2. If you want to transmit your own data then write data that need to be transmitted in input.txt and save it.
3. Run the given program and enter information according to choices given in the code. (use flag=X and esc=\ for better result.)

After completing the **Sender** program, it will save the data that is stuffed using bit/byte stuffing in **packet.txt** (will generate automatically). Then in the **Receiver** program, it will read the data send by the sender from packet.txt and will perform decoding on that data and will finally save the extracted data in **output.txt.** Then Finally, **the program will compare the output.txt and input.txt file and will print the acknowledgment message.**

OUTPUT (Byte Stuffing) :

```
-------------------------------------------
Enter 1 for Byte Stuffing!
Enter 2 for Bit Stuffing!
-------------------------------------------

1

Enter Flag Character: X

Enter Escape Character: \

Enter Minimum and Maximum frame number(x,y): 5,10

Original File data
_____________________________________________________________________________________

hello world! Welcome\tirth to dataX\X\ link layer.

##############################################################################
Framing the input according to the random limit given by the user!
##############################################################################
hello worl
d! We
lcome\t
irth to
 dataX\X
\ link la
yer.
##############################################################################

*********************************************************************
Frames Along with Byte Stuffing!          Flag=X  Escape=\
*********************************************************************

Xhello worlX

Xd! WeX

Xlcome\\tX

Xirth toX

X data\X\\\XX
```

X\\ link laX

Xyer.X

****

Sender Sends the message...

Xhello worlXXd! WeXXlcome\\tXXirth toXX data\X\\\XXX\\ link laXXyer.X

Receiver decodes the message

---

hello world! Welcome\tirth to dataX\X\ link layer.

Data Matched with Original!
Transmission Successful

OUTPUT (Bit Stuffing) :

Enter 1 for Byte Stuffing!

Enter 2 for Bit Stuffing!

2

Original File data

---

hello world! Welcome\tirth to dataX\X\ link layer.

Binary Converted data

11010001100101110110011011001101111010000011101111101111111001011011001100100010000101000001010
11111001011101100110001111011111101101110010110111001110100110100111100101110100110100001000001110

100110111101000001100100110000011110100110000110110001011100101100010111000100000110110011010
011101110110101101000001101100110000011111001110010111100100101110

Bit stuffed data (data to be send)

110100011001011101100110110011011110100000111011111001111101100101101100110010001000010100000101
010111110001011101100110001111011111010110111001011011100111010011010011111001011101001101000010000
001110100110111101000001100100110000011110100110000110110001011100101100010111000100000110110011010
011010011101110110101101000001101100110000011111000111001011110010010111010

Sender Sends the message...

Decoded bits from packet sent by sender:

110100011001011101100110110011011110100000111011111011111110010110110011001000100001010000010101
111110010111011001100011110111111011011100101101110011101001101001111001011101001101000010000011110
100110111101000001100100110000011110100110000110110001011100101100010111000100000110110011010
011101110110101101000001101100110000011111001110010111100100101110

Data Decoded

---

hello world! Welcome\tirth to dataX\X\ link layer.

Data Matched with Original!

Transmission Successful

CONCLUSION

Through this practical, we came to know one of the functions of the Data link layer and implemented it using a programming language. We came to know 2 methods for flow control and synchronization using Framing- bit stuffing and byte stuffing. I have developed a python code that is menu-driven along with the proper implementation of byte stuffing (by framing it in random size and adding a flag and escape character) and bit stuffing (by adding an extra ‘0’ if there is 5 consecutive ‘1’). I have also decoded the stuffed data and matched it with the original for safe transmission.