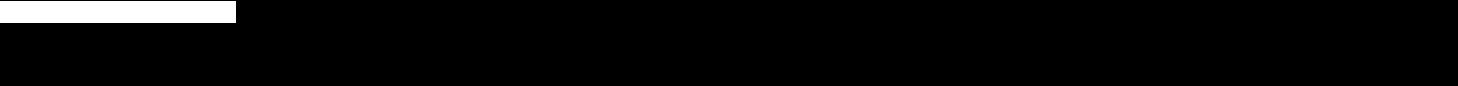
**FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEER**

**Department of Electronics and Computer Science**



**1. OUTPUT AND TRANSFER CHARACTERISTICS**

**Exp 3: Bit Stuffing & De- Stuffing Programming**

**Course, Subject & Experiment Details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Timeline (2)** | **Understanding (4)** | **Self Efforts (4)** | **Total (10)** |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student’s Name** | **Shalaka Vengurlekar** | **Roll No.** | **9171** | |
| **Academic Year** | **2022 – 23** | **Estimated Time** | **2 Hours** | |
| **Course & Semester** | **T.E. (ECS) Sem. VI** | **Subject Name** | **AI & Computer Networks Laboratory** | |
| **Unit No.** | **3** | **Chapter Title** | **Data Link Layer** | |
| **Experiment Type** | **Software Performance** | **Subject Code** | **ECL 602** | |
| **Rubrics for assessment of Experiment:**   |  |  |  |  | | --- | --- | --- | --- | | Indicator | Poor | Average | Good | | Timeliness :Maintains Experiment deadline (3) | Experiment not done (0) | One or More than One week late (1-2) | Maintains deadline (3) | | Completeness and neatness  Complete all parts of Experiment (3) | N/A | < 80% complete (1-2) | 100% complete (3) | | Originality  Extent of plagiarism (2) | Copied it from someone else (0) | At least try to implement but could not succeed (1) | Implemented (2) | | Knowledge  In depth knowledge of the Experiment (2) | Unable to answer any questions (0) | Unable to answer few questions (1) | Able to answer all questions (2) | | | | |
| **Assessment Marks:**   |  |  | | --- | --- | | Timeliness |  | | Completeness and neatness |  | | Ori Originality |  | | Kn Knowledge |  | | Tot Total |  | | | | |

**Signature of Teacher with date**

1. **Aim of the Experiment:**

To Implementing the data link layer framing methods Bit stuffing and Bit DE-stuffing.

1. **Software:**

C, Java or Python software

**4. Expected Outcome of Experiment**

Students will be able to demonstrate Bit Stuffing and De-Stuffing .A new technique allows data frames to contain arbitrary number of bits and allows character codes with arbitrary number of bits per character

**5. Theoretical Description**

**Bit Stuffing:** Bit stuffing is which an zero bit is stuffed after five consecutive ones in the input bit stream.

**Bit destuffing:** Bit destuffing is the process of removing the stuffed bit in the output stream.

**Explanation:** To provide service to network layer, the data link layer, must use the services provided to it by the physical layer. The bit stream is not guaranteed to be error free. The number of bits received may be less than, equal to, or more than data link layer to detect and, if necessary, correct errors.

The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the checksum for each frame. When a frame arrives at the destination, the checksum is re computed. If the newly computed checksum is different from one contained in the frame, the data link layer knows than an error has occurred and takes steps to deal it.

Each frame begins and ends with a special bit pattern, 01111110.When ever the sender’s data link layer encounter five consecutive 1’s in the data, it automatically stuffs a 0 bit in to outgoing bit stream. This bit stuffing is analogous to byte stuffing. When ever the receiver sees five consecutive incoming ones, followed by a 0 bit, it automatically dyestuffs the 0 bit.

**Algorithm for Bit−Stuffing & De-Stuffing**

1.Start

2. Initialize the array for transmitted stream with the special bit pattern 0111 1110 which indicates the beginning of the frame.

3. Get the bit stream to be transmitted in to the array.

4. Check for five consecutive ones and if they occur, stuff a bit 0

5. Display the data transmitted as it appears on the data line after appending 0111 1110 at the end

6. For de−stuffing, copy the transmitted data to another array after detecting the stuffed bits

7. Display the received bit stream

8. Stop

**INPUT/OUTPUT**:

Enter the input bit string :

Bit String After stuffing :

Bit String After De-Stuffing:

**Code:**

def stuff(sig):

    onecounter = 0

    index = 0

    one = []

    signal = list(sig)

    for i in signal:

        index += 1

        if i == '0':

            onecounter = 0

        else:

            onecounter += 1

        if onecounter == 5:

            one.append(index)

            onecounter = 0

    k = 0

    for i in one:

        signal.insert(i + k, '0')

        k += 1

    return signal

def destuff(sig):

    onecounter = 0

    index = 0

    one = []

    sig = list(sig)

    for i in sig:

        index += 1

        if i == '0':

            onecounter = 0

        else:

            onecounter += 1

        if onecounter == 5:

            one.append(index)

            onecounter = 0

    k = 0

    for i in one:

        sig.pop(i + k)

        k -= 1

    return sig

signal = input("Enter the signal: ")

print("Original Signal : ", signal)

stuffed = stuff(signal)

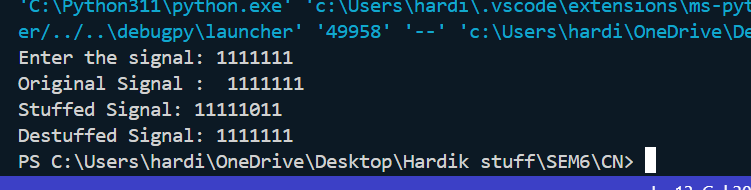
print("Stuffed Signal: ", end="")

print("".join([a for a in stuffed]))

destuffed = destuff(stuffed)

print("Destuffed Signal: ", end="")

print("".join([a for a in destuffed]))

****

**6. Post Lab Questions**

I) Explain the use of Bit Stuffing .