CN LAB 4

SAHIL BONDRE: U18CO021

Question

Implement the following datalink layer framing methods:

1. Bit stuffing
2. Character stuffing

Solution

1. Bit Stuffing

Bit stuffing is the insertion of non information bits into data. In this implementation, we'll add a 0 everytime we encounter a series of 5 ones.

bit_stuffing.py:

```
from typing import List

def bit_stuffing(bits: List[int]) -> List[int]:
    stuffed = []
    count = 0
    for i in range(len(bits)):
        if bits[i] == 1:
            count = count + 1
            stuffed.append(bits[i])
        else:
            count = 0
            stuffed.append(bits[i])
        if count == 5:
            stuffed.insert(i + 1, 0)

    return stuffed
```

```
input_bit = input("Enter input bits: ")


bits = [int(char) for char in input_bit]
print(bits)

stuffed = bit_stuffing(bits)
print(stuffed)
```

```
➜  lab-5 git:(master) ✗ python bit_stuffing.py
Enter input bits: 1011101111111101101
[1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1]
[1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1]
➜  lab-5 git:(master) ✗
```

2. Character Stuffing

Character stuffing is a process that transforms a sequence of data bytes that may contain 'illegal' or 'reserved' values (such as packet delimiter) into a potentially longer sequence that contains no occurrences of those values.

We will use a “flag” and an “escape” character to transmit the data by character stuffing in our implementation.

charecter_stuffing.py:

```
def charecter_stuffing(flag: str, escape: str, data: str) -> str:
    x = data.replace(escape, escape*2)
    y = x.replace(flag, escape+flag)
    return flag + y + flag


flag = input("Enter Flag Character: ")
escape = input("Enter Escape Character: ")
data = input("Enter Data: ")

final = charecter_stuffing(flag, escape, data)
```

```
print(final)
```


➜ lab-5 git:(master) ✗ python charecter_stuffing.py
Enter Flag Character: &
Enter Escape Character: \
Enter Data: cheese & cake
&cheese \& cake&
➜ lab-5 git:(master) ✗