Batch: B1 **Roll No.: 1711072**

Experiment / assignment / tutorial No. 3

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Experiment No.: 3

TITLE: Implementation of CRC for Computer Networks

AIM: To implement Layer 2 Error Control schemes: Cyclic Redundancy Check.

Expected Outcome of Experiment:
CO: Describe Data Link Layer, MAC layer technologies & protocols and implement the functionalities like error control, flow control.

Books/ Journals/ Websites referred:

1. A. S. Tanenbaum, "Computer Networks", Pearson Education, Fourth Edition
2. B. A. Forouzan, "Data Communications and Networking", TMH, Fourth Edition
3.

Pre Lab/ Prior Concepts:
Data Link Layer, Error Correction/Detection, Types of Errors

New Concepts to be learned: CRC.

CRC:

1. Consider 10011 as a data stream.
2. Consider 101 as a generator polynomial.
3. Add 00 at the end of the data stream. Data stream 1001100.
4. Divide the data stream by 101.
5. Append the remainder at the end of the data strea
6. Send that data stream to the receiver.
7. Repeat the same function at the receiver end side.
8. If remainder 0 then data correct otherwise data incorrect.

IMPLEMENTATION: (printout of codes)**:**

```
data=input("Enter the data to be sent: ")
gen=input('Enter the genf key: ')
red_code=str('0'*(len(gen)-1))
ap_data=data+str('0'*(len(gen)-1))
print("Appended Data is: ", ap_data)
def CRC(code,genf,redundant):
  code=code+redundant
  code,genf=list(code),list(genf)
  for i in range(len(code)-len(redundant)):
    if code[i]=='1':
      for j in range(len(genf)):
        code[i+j] = str(int(code[i+j])^int(genf[j]))
    print(''.join(code))
  return ''.join(code[-len(redundant):])
rem=CRC(data,gen,red_code)
res=data+rem
print('Remainder generated: ', rem,'\nData transmitted is: ',res)
rec=input("Enter the data received: ")
rec_data=CRC(rec[:-(len(gen)-1)], gen, rec[-len(gen)+1:])
print("Remainder on receiver side is: ",rec_data)
```

Output Screen:

```
Enter the data to be sent: 1011011
Enter the genf key: 1101
Appended Data is:  1011011000
0110011000
0000111000
0000111000
0000111000
0000001100
0000001100
0000000001
Remainder generated:  001
Data transmitted is:  1011011001
Enter the data received: 1011011001
0110011001
0000111001
0000111001
0000111001
0000001101
0000001101
0000000000
Remainder on receiver side is:  000
```

CONCLUSION:

The Python code for CRC Generator worked for both sender and receiver side successfully for all test cases.

Lab Questions

1. In CRC there is no error if the remainder at the receiver is __.
   A.equal to the remainder at the sender
   B.zero
   C.nonzero
   D.the quotient at the sender

Ans. B. zero

2. Let G(x) be the generator polynomial used for CRC checking. What is the condition that should be satisfied by G(x) to detect odd number of bits in error?

   A. G(x) contains more than two terms
   B. G(x) does not divide 1+x^k, for any k not exceeding the frame length
   C. 1+x is a factor of G(x)
   D. G(x) has an odd number of terms.

Ans. C. 1+x is a factor of G(x).