Batch: C2	Roll No.: 110
Experiment / assignment / tutorial No.______	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Experiment No.:4

TITLE: Implementation of CRC & Checksum for Computer Networks

AIM: To implement Layer 2 Error Control schemes: CRC & Checksum.

Expected Outcome of Experiment:
CO: 3

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

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

New Concepts to be learned: Checksum.

CRC(Cyclic Redundancy Check):

Cyclic Redundancy Check (CRC) is another error detection technique to detect errors in data that has been transmitted on a communications link. A sending device applies a 16 or 32 bit polynomial to a block of data that is to be transmitted and appends the resulting cyclic redundancy check (CRC) to the block. The receiving end applies the same polynomial to the data and compares its result with the result appended by the sender. If they agree, the data has been received successfully. If not, the sender can be notified to resend the block of data.

At Sender Side:

- Sender has a generator G(x) polynomial.
- Sender appends (n-1) zero bits to the data.
  Where, n= no of bits in generator
- Dividend appends the data with generator G(x) using modulo 2 division (arithmetic).
- Remainder of (n-1) bits will be CRC.

Codeword: It is combined form of Data bits and CRC bits i.e. Codeword = Data bits + CRC bits.

Example
Assume that –
(a) data is 10110.
(b) code generator is 1101.
(Code generator can also be mentioned in polynomial : x^3+x^2+1)

Calculate CRC Bits: While calculating the CRC bits, we pad (n-1) 0's to the message bits, where 'n' = no of bits in the code generator.

Cyclic Redundancy check will be generated as shown below –

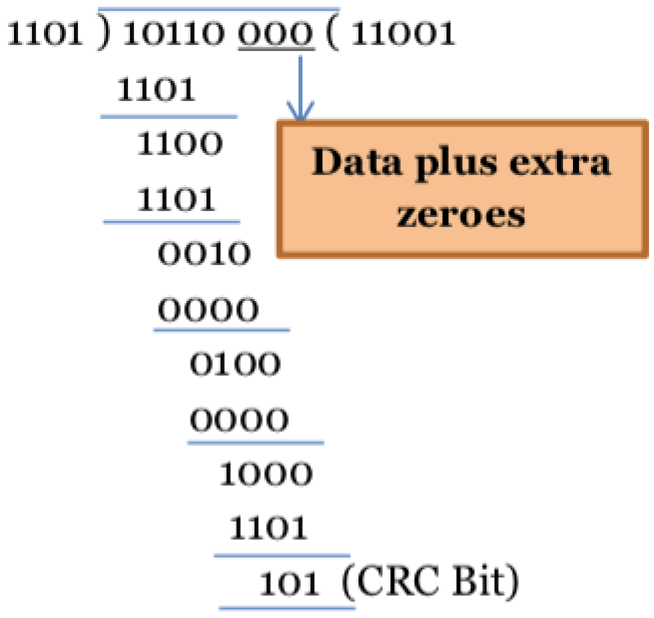


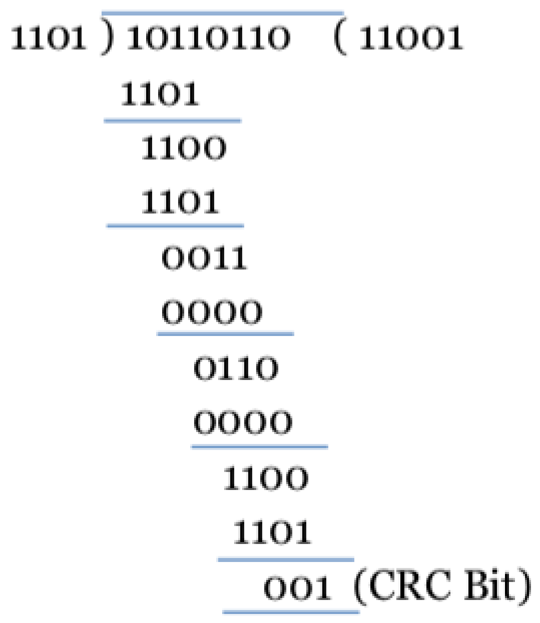
Figure 1: CRC calculation by sender

At Receiver Side
- Receiver has same generator G(x).
- Receiver divides received data (data + CRC) with generator.

- If remainder is zero, data is correctly received.
- Else, there is error.

Assume the received message is 10110110.

Calculate CRC Bits: It does not add any padding bits, rather calculates from the entire received code word.



```
1101 ) 10110110   ( 11001
       1101
       ----
        1100
        1101
        ----
         0011
         0000
         ----
          0110
          0000
          ----
           1100
           1101
           ----
            001 (CRC Bit)
```

Figure 2: CRC calculation by receiver

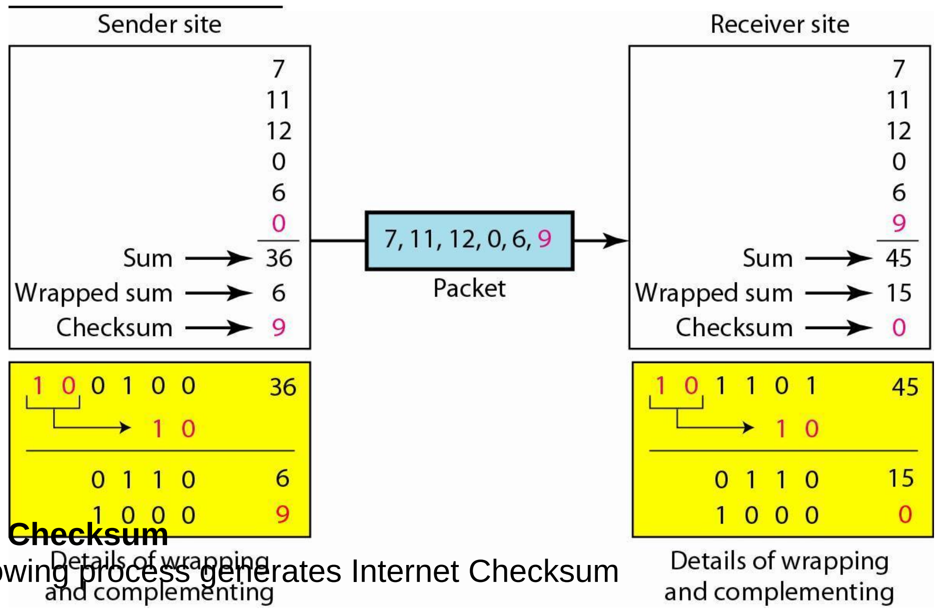
The CRC bits are calculated to be different. Thus, there is an error detected.

Internet Checksum :

A checksum is a simple type of redundancy check that is used to detect errors in data. Errors frequently occur in data when it is written to a disk, transmitted across a network or otherwise manipulated. The errors are typically very small, for example, a single incorrect bit, but even such small errors can greatly affect the quality of data, and even make it useless.

In its simplest form, a checksum is created by calculating the binary values in a packet or other block of data using some algorithm and storing the results with the data. When the data is retrieved from memory or received at the other end of a network, a new checksum is calculated and compared with the existing checksum. A non-match indicates an error; a match does not necessarily mean the absence of errors, but only that the simple algorithm was not able to detect any.

Simple Checksum:



Internet Checksum
The following process generates Internet Checksum

Assume the packet header is: 01 00 F2 03 F4 F5 F6 F7 00 00
(00 00 is the checksum to be calculated)

The first step is to form 16-bit words.
0100 F203 F4F5 F6F7

The second step is to calculate the sum using 32-bits.
0100 + F203 + F4F5 + F6F7 = 0002 DEEF

The third step is to add the carries (0002) to the 16-bit sum.
DEEF + 002 = DEF1

The fourth step is to take the complement. (1s becomes 0s and 0s become 1s)
~DEF1 = 210E

So the checksum is 21 0E.

The packet header is sent as: 01 00 F2 03 F4 F5 F6 F7 21 0E

* At the receiver, the steps are repeated.

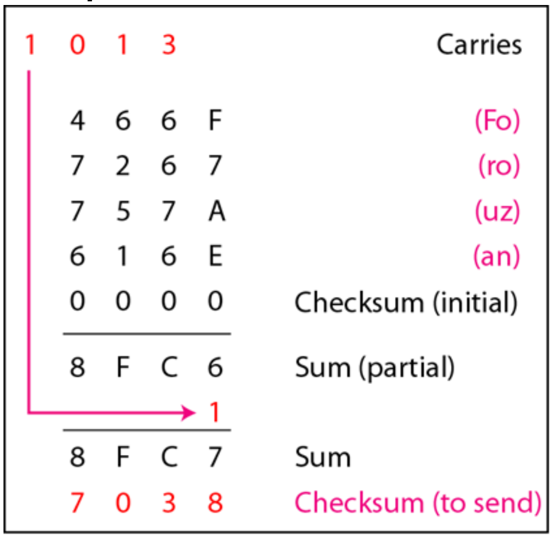
The first step is to form 16-bit words.
0100 F203 F4F5 F6F7 210E

The second step is to calculate the sum using 32-bits.
0100 + F203 + F4F5 + F6F7 + 210E = 0002 FFFD

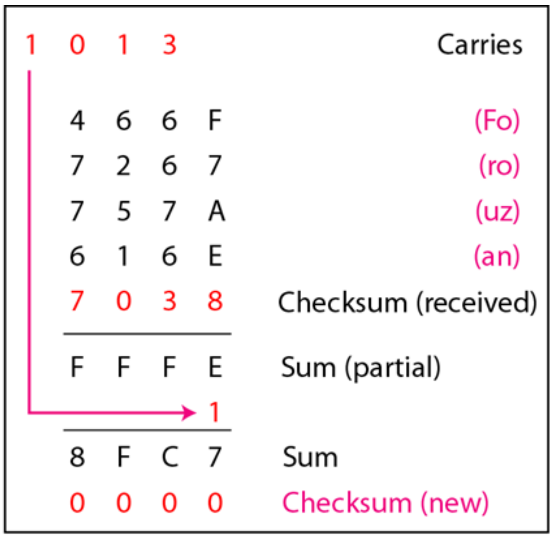
The third step is to add the carries (0002) to the 16-bit sum.
FFFD + 0002 = FFFF which means that no error was detected.

(In 1s complement, zero is 0000 or FFFF.)

Example:



a. Checksum at the sender site



a. Checksum at the receiver site

IMPLEMENTATION: (printout of codes)

CRC

```
generator='1001'
code='11100101'

def toDecimal(a):
    a=str(a)
    answer=0
    power=0
    for digit in a[::-1]:
        if(digit=='1'):
            answer+=2**power
        power+=1
    return answer

def toBinary(a):
    a=int(a)
    answer=[]
    if(a==0):
        return 0
    while(a!=0):
        answer.append(str(a%2))
        a=a//2
    return("".join(answer[::-1]))

sent=str(code+str(toBinary(toDecimal(code)%toDecimal(generator))))

print("Sent data is ",sent)
print()
recieved = sent

isCorrect = True if toBinary(toDecimal(recieved)%toDecimal(generator)) ==0 else False
print("Data recieved correctly?", isCorrect)
```

CHECKSUM

```
def toDecimal(a):
    a=str(a)
    answer=0
    power=0
    for digit in a[::-1]:
        if(digit=='1'):
            answer+=2**power
        power+=1
    return answer

def toBinary(a):
    a=int(a)
    answer=[]
    if(a==0):
        return 0
    while(a!=0):
        answer.append(str(a%2))
        a=a//2
    return("".join(answer[::-1]))


data=['110','101','001','000']

def checkSumCalculate(data):
    checkSum=toBinary(sum([toDecimal(int(i)) for i in data])) #110+101+001+000 = 1100

    if(len(checkSum)>len(data[1])):
        #wrap carry
        checkSum=toBinary(toDecimal(checkSum)+1)
        checkSum = str(checkSum) # necesary to convert to string
        checkSum=checkSum[1::] # remove first digit
        #101
    #complement checkSum
    checkSum = str(checkSum) # necesary to convert to string
    checkSum="".join(['1' if i=='0' else '0' for i in checkSum]) #010
    return checkSum

sent = data+[checkSumCalculate(data)]
print("data sent is",sent)

recieved = sent

validate = True if checkSumCalculate(recieved)=='000' else False
print(validate)
```

OUTPUT:

```
Sent data is  11100101100

Data recieved correctly? True
```

```
data sent is ['110', '101', '001', '000', '010']
True
```

CONCLUSION:

Thus we have understood how error is detected in computer networks. We have implemented CRC (cyclic redundancy control) and checksum method. A 16-bit sum-of-words checksum will detect all single bit errors and all error bursts of length 16 bits or fewer. It will also detect 99.998% of longer error bursts. Mathematically speaking, a 4-bit CRC applied to a data block of arbitrary length will detect any single error burst not longer than 4 bits, and a fraction 1 − 2−n of longer error bursts. We used programming language to simulate the error control methods.These two methods can be used to detect errors in computer networks. If error is found then retransmission occurs.

Post Lab Questions

1. Discuss about the rules for choosing a CRC generator.
2. State the advantages and disadvantages of Internet Checksum.

Date :___________ **Signature of Faculty In-charge**