COMPUTER NETWORKS
LAB MANUAL

B. TECH

III YEAR – I SEM (R23)

(2025-26)

DEPARTMENT OF CSE-AIML

Aditya College of Engineering & Technology

Aditya Nagar, ADB Road, Surampalem – 533437

SYLLABUS

1. Study of Network devices in detail and connect the computers in Local Area Network.
2. Write a Program to implement the data link layer framing methods such as
   i) Character stuffing ii) bit stuffing.
3. Write a Program to implement data link layer framing method checksum.
4. Write a program for Hamming Code generation for error detection and correction.
5. Write a Program to implement on a data set of characters the three CRC polynomials CRC 12, CRC 16 and CRC CCIP.
6. Write a Program to implement Sliding window protocol for Goback N.
7. Write a Program to implement Sliding window protocol for Selective repeat.
8. Write a Program to implement Stop and Wait Protocol.
9. Write a program for congestion control using leaky bucket algorithm
10. Write a program to implement Dijkstra’s algorithm to compute the shortest path through a graph.
11. Write a Program to implement Distance vector routing algorithm by obtaining routing table at each node (Take an example subnet graph with weights indicating delay between nodes).
12. Write a Program to implement Broadcast tree by taking subnet of hosts.
13. Wireshark
    i. Packet Capture Using Wire shark
    ii. Starting Wire shark
    iii. Viewing Captured Traffic
    iv. Analysis and Statistics & Filters.
14. How to run Nmap scan
15. Operating System Detection using Nmap
16. Do the following using NS2 Simulator
    i. NS2 Simulator-Introduction
    ii. Simulate to Find the Number of Packets Dropped
    iii. Simulate to Find the Number of Packets Dropped by TCP/UDP
    iv. Simulate to Find the Number of Packets Dropped due to Congestion
    v. Simulate to Compare Data Rate& Throughput.each node, and for re-creating the original graph with the updated PageRank values.

COMPUTER NETWORKS LAB

III B.Tech I-Semester

COURSE OUTCOMES

S.No	Course Code - CO	Course Outcomes	Blooms Taxonomy
1	CO1	***Know*** how to connect computers in LAN and ***implement*** different framing methods.	Application
2	CO2	***Implement*** error detection correction techniques.	Application
3	CO3	***Know*** how reliable data communication is achieved through data link layer.	Application
4	CO4	***Suggest*** appropriate routing algorithm for the network.	Application
5	CO5	***Provide*** internet connection to the system and its installation.	Application
6	CO6	***Work*** on various network management tools	Application

COMPUTER NETWORKS LAB

III B.Tech I-Semester

CO &POs MAPPING

Course Code	Course Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO 10	PO 11	PO 12
CO1	***Know*** how to connect computers in LAN and ***implement*** different framing methods.	2	2	1	1	1				1			1
CO2	***Implement*** error detection correction techniques.	2	2	3	1	1				2			2
CO3	***Know*** how reliable data communication is achieved through data link layer.	1	2	2	2	1				1			2
CO4	***Suggest*** appropriate routing algorithm for the network.	2	2	2	2	1				2			2
CO5	***Provide*** internet connection to the system and its installation.	1	2	2	2	1				1			2
CO6	***Work*** on various network management tools	1	2	2	2	2				1			2

COMPUTER NETWORKS LAB (R2032121)

III B.Tech I-Semester

CO & PSO MAPPING

Course Code	Course Outcomes	PSO1	PSO2	PSO3
CO1	***Know*** how to connect computers in LAN and ***implement*** different framing methods.	2	2	1
CO2	***Implement*** error detection correction techniques.	2	2	3
CO3	***Know*** how reliable data communication is achieved through data link layer.	1	2	2
CO4	***Suggest*** appropriate routing algorithm for the network.	2	2	3
CO5	***Provide*** internet connection to the system and its installation.	1	2	3
CO6	***Work*** on various network management tools	1	2	3

GENERAL INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.

2. Student should enter into the laboratory with:
   - Laboratory observation notes.
   - Laboratory Record updated up to the last session experiments.
   - Proper Dress code and Identity card.

3. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.

4. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.

5. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.

6. Computer labs are established with sophisticated and high-end branded systems, which should be utilized properly.

7. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.

8. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

9. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

<u>TOOLS USED DURING THE LAB</u>

PYTHON	python™
WIRESHARK	Wireshark
NMAP	NMAP
NS2 SIMULATOR	ns-2 NETWORK SIMULATOR

INDEX

EXPERIMENT 1

Study of Network devices in detail and connect the computers in Local Area Network.

AIM: Study of various network devices in detail All but the most basic of networks require devices to provide connectivity and functionality.Understanding how these networking devices operate and identifying the functions they performare essential skills for any network administrator and requirements for aNetwork+ candidate.Theall networkdevicesareexplained below:

Hubs:

The hub or network hub connects computers and devices and sends messages and data from any one device to all the others.If the desktop computer wants to send data to the laptop and it sends amessage to the laptop through the hub,the message willgetsentby the hub to all the computers and devices on the network.They need to do work to figure out that the message is not for them. The message also uses up bandwidth (room) on the network wires or wireless radiowavesandlimits howmuch communication can go on.Hubs arenot usedoften thesedays.

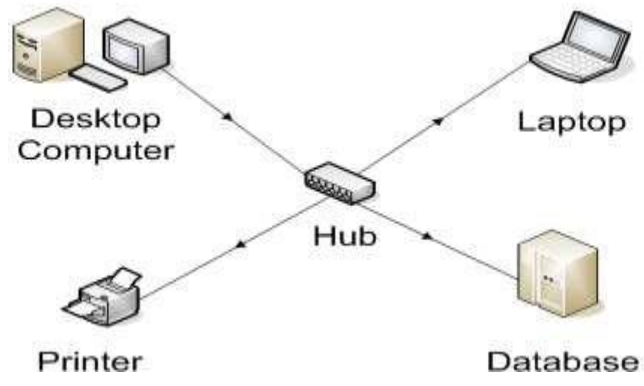


Figure 1

Switch:

The switch connects the computer network components but it is smart about it. It knows the address of each item and so when the desktop computer wants to talk to the laptop, it only sends the message to the laptop and nothing else. In order to have a small home network that just connects the local equipment all that is really needed is a switch and network cable or the switch can transmit wireless information that is received by wireless receivers that each of the network devices have.

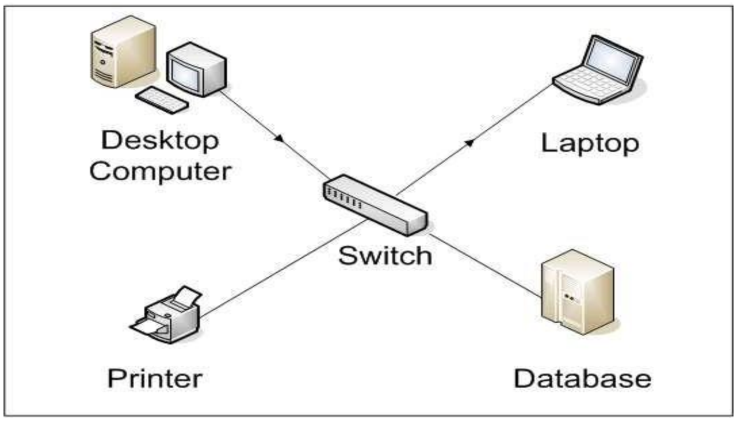


Figure 2

Bridges:

Bridges are used to divide larger networks into smaller sections. They do this by sitting between two physical network segments and managing the flow of data between the two. By looking at the MAC address of the devices connected to each segment, bridges can elect to forward the data (if they believe that the destination address is on another interface), or block it from crossing (if they can verify that it is on the interface from which it came).

A bridge functions by blocking or forwarding data, based on the destination MAC address written into each frame of data. If the bridge believes the destination address is on a network other than that from which the data was received, it can forward the data to the other networks to which it is connected. If the address is not on the other side of the bridge, the data is blocked from passing. Bridges "learn" the MAC addresses of devices on connected networks by "listening" to network traffic and recording the network from which the traffic originates. Figure 3 shows a representation of a bridge.

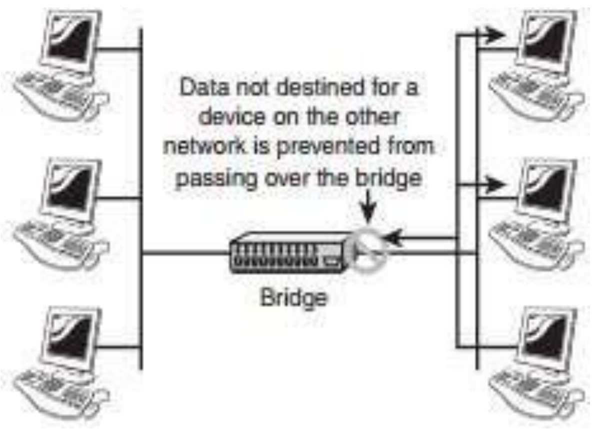

Data not destined for a device on the other network is prevented from passing over the bridge
Bridge

Figure 3

Routers:

In a common configuration, routers are used to create larger networks by joining two network segments. A router derives its name from the fact that it can route data it receives from one network onto another. When a router receives a packet of data, it reads the header of the packet to determine the destination address. Once it has determined the address, it looks in its routing table to determine whether it knows how to reach the destination and, if it does, it forwards the packet to the next hop on the route. The next hop might be the final destination, or it might be another router. Figure 4 shows, in basic terms, how a router works.

The routing tables play a very important role in the routing process. They are the means by which the router makes its decisions. For this reason, a routing table needs to be two things. It must be up-to-date, and it must be complete. There are two ways that the router can get the information for the routing table—through static routing or dynamic routing.

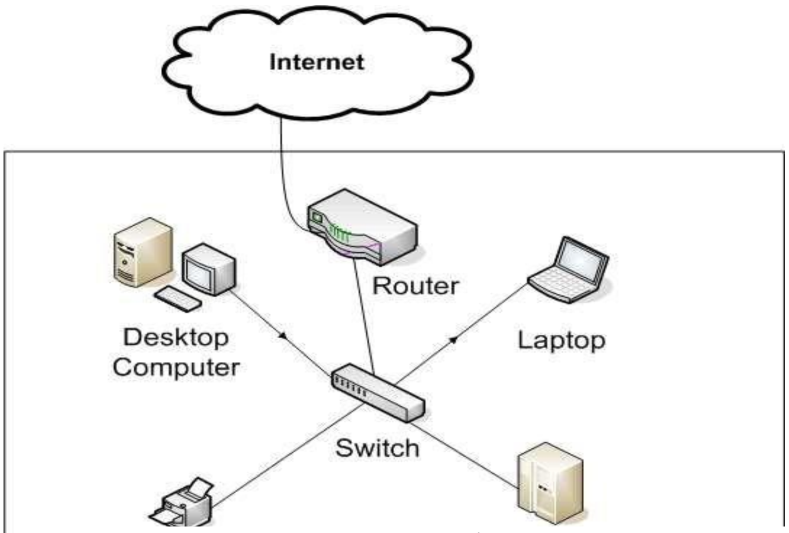


Figure 4

Modem:

Most everyone wants to connect to the internet. A broadband modem is used to take a high speed Internet connection provided by an ISP (Internet Service Provider) and convert the data into a form that your local network can use. The high speed connection can be DSL (Digital Subscriber Line) from a phone company or cable from a cable television provider.

In order to be reached on the Internet, your computer needs a unique address on the internet. Your ISP will provide this to you as part of your Internet connection package. This address will generally not be fixed which means that they may change your address from time to time. For the vast majority of users, this makes no difference. If you have only one computer and want to connect to the Internet, you strictly speaking don't need a router. You can plug the network cable from the modem directly into the network connection of your computer. However, you are much better off connecting the modem to a router. The ip address your ISP provides will be assigned to the router.

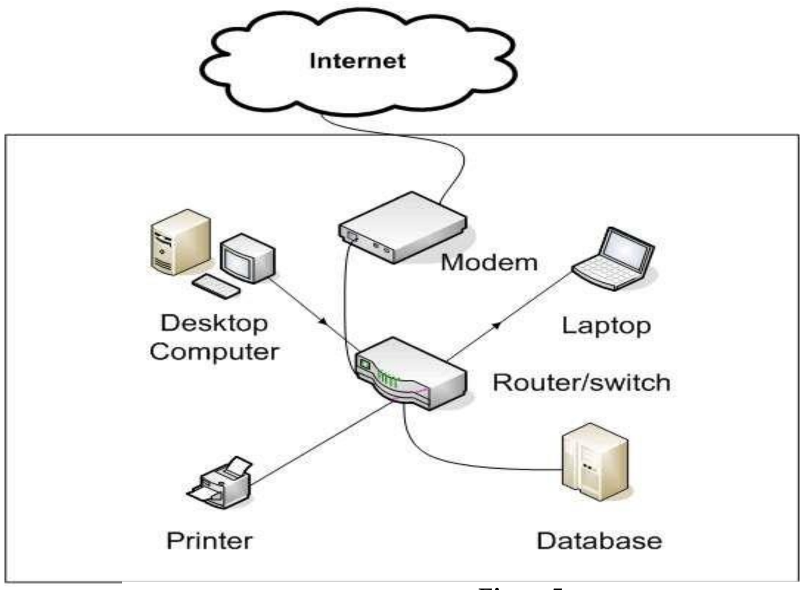


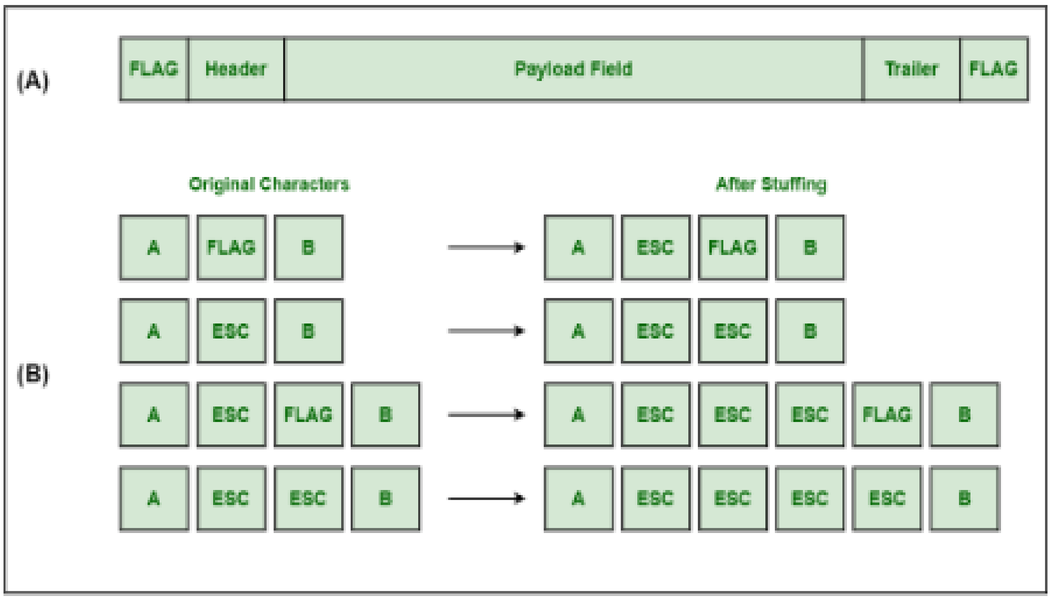
Figure 5

EXPERIMENT 2

2. i) Write a Program to implement the data link layer framing methods such as --> character stuffing.

Character Stuffing :
Character stuffing is also known as byte stuffing or character-oriented framing and is same as that of bit stuffing but byte stuffing actually operates on bytes whereas bit stuffing operates on bits. In byte stuffing, special byte that is basically known as ESC (Escape Character) that has predefined pattern is generally added to data section of the data stream or frame when there is message or character that has same pattern as that of flag byte.

But receiver removes this ESC and keeps data part that causes some problems or issues. In simple words, we can say that character stuffing is addition of 1 additional byte if there is presence of ESC or flag in text



A Character Stuffing
(A) A frame delimited by flag bytes
(B) Four examples of byte sequences before and after byte stuffing

PROGRAM:

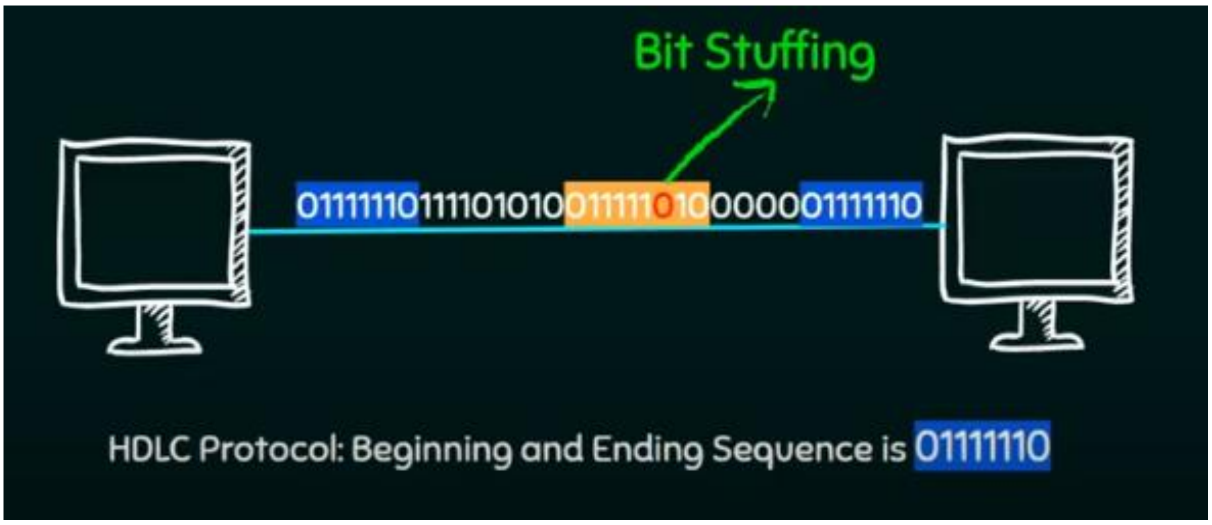
```
head = input("Enter character that represents the starting delimiter: ")
tail = input(" Enter character that represents the ending delimiter: ")
st = input("Enter the characters to be stuffed: ")
res=head
for i in st:
      if i==head or i ==tail:
             res = res + i + i
      else:
             res = res + i
res = res+tail
print("Frame after character stuffing: ", res)
```

OUTPUT:

```
Enter character that represents the starting delimiter: d

Enter character that represents the ending delimiter: g

Enter the characters to be stuffed: goodday

Frame after character stuffing: dggooddddayg
```

2. ii) Write a Program to implement the data link layer framing methods such as --> bit stuffing.

BitStuffing :
Bit stuffing is also known as bit-oriented framing or bit-oriented approach. In bit stuffing, extra bits are being added by network protocol designers to data streams. It is generally insertion or addition of extra bits into transmission unit or message to be transmitted as simple way to provide and give signaling information and data to receiver and to avoid or ignore appearance of unintended or unnecessary control sequences.



PROGRAM:

```
st = input ("Enter the frame: ") count = 0
res = ""
for i in st:
       if i == '1' and count < 5:
          res += '1'
          count += 1
       elif i == ' ':
          pass
       else:
          res += i count = 0
       if count == 5:
          res += '0'
          count= 0
print("Frame after bit stuffing: ", res)
```

OUTPUT:

```
Enter the frame: 01111110

Frame after bit stuffing: 011111010
```

EXPERIMENT 3

3. Write a Program to implement data link layer framing method checksum.

PROGRAM:

```
s1 = input("Enter the string of 0's and 1's as subunit1: ")

s2 = input("Enter the string of 0's and 1's as subunit2: ")

# Reverse both strings for easier addition (LSB first)

s1 = s1[::-1]

s2 = s2[::-1]

res = ""

c = '0'   # carry

# Binary addition of s1 and s2

for i, j in zip(s1, s2):

    if i == '0' and j == '0' and c == '0':

        res += '0'

        c = '0'


    elif i == '0' and j == '0' and c == '1':

        res += '1'

        c = '0'


    elif i == '0' and j == '1' and c == '0':

        res += '1'

        c = '0'


    elif i == '0' and j == '1' and c == '1':

        res += '0'

        c = '1'
    elif i == '1' and j == '0' and c == '0':
        res += '1'
        c = '0'

    elif i == '1' and j == '0' and c == '1':
        res += '0'
        c = '1'

    elif i == '1' and j == '1' and c == '0':
        res += '0'
        c = '1'

    elif i == '1' and j == '1' and c == '1':
        res += '1'
        c = '1'
# Handle final carry
if c == '1':
    ans = ""
    for i in res:
        if i == '1' and c == '1':
            ans += '0'
            c = '1'
        elif i == '0' and c == '0':
            ans += '0'
            c = '0'
        else:
            ans += '1'
            c = '0'
    res = ans
```

```
# Take 1's complement

final = ""

for i in res:

    if i == '1':

        final += '0'

    else:

        final += '1'


print("Checksum of two subunits: ", final[::-1].strip())
```

OUTPUT:

```
Enter the string of 0's and 1's as subunit1: 10101001

Enter the string of 0's and 1's as subunit2: 00111001

Checksum of two subunits: 00011101
```

EXPERIMENT 4

4. Write a program for Hamming Code generation for error detection and correction.

PROGRAM:

```
li = list(map(int,input("Enter 7 bits data of 0's and 1's separated
by spaces: ").split()))
rec = list(map(int,input("Enter the received 11 data bits of 0's
and 1's separated by spaces: ").split()))
# reverse the list
li = li[::-1]
# parity bits of 0 are added at the place of 2 pow's i.e. at
positions of 1,2,4,8 remaining places data bits are added
li = [0,0] + li[0:1] + [0] + li[1:4] + [0] + li[4:]
#now find the even parity bit position
li[0] = (li[2] + li[4] + li[6] + li[8] + li[10]) % 2
li[1] = (li[2] + li[5] + li[6] + li[9] + li[10]) % 2
li[3] = (li[4] + li[5] + li[6]) % 2
li[7] = (li[8] + li[9] + li[10]) % 2
# reverse the list
li = li[::-1]
#reverse the receiver side data and check the parity bits position
values
rec = rec[::-1]
r1 = (rec[0] + rec[2] + rec[4] + rec[6] + rec[8] + rec[10]) % 2
r2 = (rec[1] + rec[2] + rec[5] + rec[6] + rec[9] + rec[10]) % 2
r3 = (rec[3] + rec[4] + rec[5] + rec[6]) % 2
r4 = (rec[7] + rec[8] + rec[9] + rec[10]) % 2

bit = str(r4) + str(r3) + str(r2) + str(r1)
bit = int(bit,2)
if bit :
   print("received data is having error at position: ", bit)
else:
   print("received data doesn't have any error")
```

OUTPUT:

```
Enter 7 bits data of 0’s and 1’s separated by spaces: 1 0 1 0
1 0 1

Enter the received 11 data bits of 0's and 1's separated by
spaces 1 0 1 0 0 1 0 1 1 0 1

Received data is having error at position: 2

Enter 7 bits data of 0's and 1's separated by spaces: 1 0 1 0
1 0  1

Enter the received 11 data bits of 0's and 1's separated by 1
1 1  spaces:   1 0 1 0 0 1 0 1

received data doesn't have any error
```

EXPERIMENT 5

5. Write a Program to implement on a data set of characters the three CRC polynomials CRC 12, CRC 16 and CRC CCITT.

PROGRAM

```
def xor(x, y):
    ans = ""
    for i in range(len(y)):  # Compare all bits
        if x[i] == y[i]:
            ans += '0'
        else:
            ans += '1'
    return ans


def divide(dividend, divisor):
    a = len(divisor)
    temp = dividend[0:a]
    # Loop until all bits of the dividend have been processed
    while a < len(dividend):
        if temp[0] == '1':
            temp = xor(divisor, temp) + dividend[a]
        else:
            temp = xor('0' * a, temp) + dividend[a]
        a += 1
        if temp[0] == '1':
            temp = xor(divisor, temp)
        else:
            temp = xor('0' * a, temp)
    return temp
```

```
# Predefined generator polynomials

keys = ['1100000001111','11000000000000101',

  '10001000000100001']

print("Choose the CRC")

print("1. CRC - 12")

print("2. CRC - 16")

print("3. CRC - CCITT ")


n = int(input("Enter your choice (1/2/3): "))


send = input("Enter the string of binary data bits to be sent
from the sender: ")

rec = input("Enter the string of binary data received at the
receiver side: ")


# Select the appropriate key

key = keys[n - 1]


# Encoding on sender's side

length = len(key)

send1 = send + '0' * (length - 1)

rem = divide(send1, key)

# Decoding on receiver's side

ans = divide(rec, key)

# Check for transmission errors

if ans == '0' * (len(key) - 1):

    print("No error in transmission ")

else:

    print("Frame error detected ")
```

OUTPUT:

```
Choose the CRC


1. CRC - 12

2. CRC- 16

3. CRC- CCITT

1

Enter the string of code word of binary data bits of 0's and
1's to be sent from the sender: 1011


Enter the string of code word of binary data received at the
receiver side: 1011110

Sent Codeword: 1011001

no error


Choose the CRC


1. CRC - 12

2. CRC- 16

3. CRC- CCITT

2

Enter the string of code word of binary data bits of 0's and
1's to be sent from the sender: 101110111010101

Enter the string of code word of binary data received at the
receiver side: 1011101110101010100110011111011

no error


Choose the CRC


1. CRC- 12
```

```
2. CRC- 16

3. CRC- CCITT

1


Enter the string of code word of binary data bits of 0's and
1's to be sent from the sender: 1010101

Enter the string of code word of binary data received at the
receiver side: 10101010010000000010


no error
```

EXPERIMENT 6

6. Write a Program to implement Sliding window protocol for Go back N.

PROGRAM:

SENDER SIDE:

```
import socket
import random
import time

s = socket.socket()
s.bind(("localhost", 1450))
s.listen(5)
c, adr = s.accept()
print(str(adr))
n = int(input("Enter number of frames: "))
N = int(input("Enter window size: "))
seq = 1 # is used to keep track of the window starting
frame = 1 # frame to send starts with 1
# send first N window size frames
for i in range(N):
      print('Frames sent ->', frame)
      c.send(str(frame).encode())
      frame += 1
      time.sleep(2)
 timer = 5

# will start with acknowledgement frame of 1
while frame <= n:
 t = random.randint(1, 7)

 msg = c.recv(1).decode()

 msg = int(msg)


if (msg != seq):

# here we try to discard the already sent frames after
failed frame

      continue

if (timer > t):

# if the timer is greater than random number be consider
it as ack

      print("acknowledgement received")

      print('Frames sent ->', str(frame))

      # we will send next frame

      c.send(str(frame).encode())

      seq += 1

      frame += 1

      time.sleep(2)

else:

# if timer is less than the random number we consider as
not received ack

      print('acknowledgement not received')

      frame = seq

       # we will again send the frames from window
       starting i.e. seq

      for i in range(N):

           print('Frames sent ->', frame)

           c.send(str(frame).encode())

           frame += 1

           time.sleep(2)
```

RECEIVER SIDE:

```
import socket
import time
s=socket.socket()
s.connect(("localhost", 1450))
while 1:
      msg=s.recv(2).decode()
      print("Received --> ",int(msg))
      s.send(str(msg).encode())
      time.sleep(1)
```

OUTPUT:

SENDER SIDE: **RECEIVER SIDE:**

```
Enter number of frames: 8                 Received --> 1
Enter window size: 4                      Received --> 2
Frames sent -> 1                          Received --> 3
Frames sent -> 2                          Received --> 4
Frames sent -> 3                          Received --> 5
Frames sent -> 4                          Received --> 6
acknowledgement received                  Received --> 3
Frames sent -> 5                          Received --> 4
acknowledgement received                  Received --> 5
Frames sent -> 6                          Received --> 6
acknowledgement not received              Received --> 3
```

```
Frames sent -> 3                                    Received --> 4

Frames sent -> 4                                    Received --> 5

Frames sent -> 5                                    Received --> 6

Frames sent -> 6                                    Received --> 7

acknowledgement not received                        Received --> 8

Frames sent -> 3

Frames sent -> 4

Frames sent -> 5

Frames sent -> 6

acknowledgement received

Frames sent -> 7

acknowledgement received

Frames sent -> 8
```

EXPERIMENT 7

7. Write a Program to implement Sliding window protocol for Selective repeat.

PROGRAM:

SENDER SIDE:

```
import socket
import random
import time
s = socket.socket()
s.bind(("localhost",8038))
s.listen(5)
c, adr = s.accept()
print("from address”, str(adr), "connection has established")
n = int(input("Enter number of frames: "))
N = int(input("Enter window size: "))
seq = 1 # is used to keep track of the window starting
frame = 1 # frame to send starts with 1
# send first N window size frames
for i in range(N):
    print('Frames sent ->',frame)
    c.send(str(frame).encode()) frame += 1
    time.sleep(2)
timer = 5 # will start with acknowledgement frame of 1
while frame <= n :
     t = random.randint(1,7)
     msg = c.recv(1).decode()
     msg = int(msg)
     print("Frame ", msg)
     if(timer > t):
         print("acknowledgement received")
         print('Frames sent ->',str(frame))
         c.send(str(frame).encode())
         seq += 1
         frame += 1
         time.sleep(2)
     else:
         print('acknowledgement not received')
         print('Frames sent ->',msg)
         c.send(str(msg).encode())
         time.sleep(2)
```

RECEIVER SIDE:

```
import socket
import time
s=socket.socket()
s.connect(("localhost", 8038))
while 1:
   msg=s.recv(2).decode()
   print("Received --> ",int(msg))
   s.send(str(msg).encode())
   time.sleep(1)
```

OUTPUT:

SENDER SIDE:

```
Enter number of frames: 8

Enter window size: 4

Frames sent -> 1

Frames sent -> 2

Frames sent -> 3

Frames sent -> 4

Frame 1

acknowledgement not received

Frames sent -> 1

Frame 2

acknowledgement received

Frames sent -> 5
Frame    3
acknowledgement received
Frames sent -> 6
Frame    4
acknowledgement received
Frames sent -> 7
Frame 1
acknowledgement not received
Frames sent -> 1
Frame    5
acknowledgement received
Frames sent -> 8
Received --> 7
Received --> 1
Received --> 8
```

RECEIVER SIDE:

```
Received --> 1

Received --> 2

Received --> 3

Received --> 4

 Received --> 1

 Received --> 5

 Received --> 6

Received --> 7

 Received --> 1

 Received --> 8

Received --> 1
```

EXPERIMENT 8

8. Write a Program to implement Stop and Wait Protocol.

PROGRAM:

SENDER SIDE:

```
import socket

import time

import random

s=socket.socket()

s.bind(("localhost", 8020))

s.listen(5)

c, adr = s.accept()

print("connection to " + str(adr) + " established")

a=int(input("enter total number of frames"))

x = 0

print("sending -->", x)

c.send(str(x).encode())

while( a > 1 ):

     timer = 5

     t=random.randint(1,7)

     msg = c.recv(1).decode()

     if (timer > t):

        time.sleep(3)

        print("ack-->", msg)

        x=int(msg)

        print("sending -->", str(x))

        c.send(str(x).encode())
```

```
    else:
           time.sleep(3)
           print("timeout")
           print("sending again-->", x)
           c.send(str(x).encode())
           a=a+1
    a = a-1
```

RECEIVER SIDE:

```
import socket
s=socket.socket()
s.connect(("localhost", 8020))
while(1):
  msg=s.recv(1).decode()
  print("Received --> ", msg)
  x=int(msg)
  if(x==0):
     x=x+1
     s.send(str(x).encode())
  else:
     x=x-1
     s.send(str(x).encode())
```

OUTPUT:

SENDER SIDE:

```
enter total number of frames6
sending --> 0
timeout
sending again--> 0
timeout
sending again--> 0
ack--> 1
sending --> 1
ack--> 0
sending --> 0
timeout
sending again--> 0
timeout
sending again--> 0
timeout
sending again--> 0
ack--> 1
sending --> 1
timeout
sending again--> 1
timeout
sending again--> 1
ack--> 0
sending --> 0
```

RECEIVER SIDE:

```
Received --> 0
Received --> 0
Received --> 0
Received --> 1
Received --> 0
Received --> 0
Received --> 0
Received --> 0
Received --> 1
Received --> 1
Received --> 1
Received --> 0
Received --> 0
Received --> 1
```

```
timeout

sending again--> 0

ack--> 1

sending --> 1
```

EXPERIMENT 9

9. Write a program for congestion control using leaky bucket algorithm

PROGRAM:

```
print("Enter bucket size, outgoing rate, number of inputs and
incoming size")

bucketsize = int(input())

outgoing = int(input())

n = int(input())

incoming = int(input())

store=0

while n!= 0:

    print("Incoming size is ", incoming)

    if incoming <= (bucketsize-store):

        store += incoming

        print("Bucket buffer size is ",store," out of ",
        bucketsize)

    else:

        print("Packet loss : ",(incoming-(bucketsize-store)))

        store=bucketsize

        print("Bucket buffer size is ",store," out of ",
        bucketsize)

    store -= outgoing;

    print("After outgoing: " ,store," packets left out of ",
    bucketsize ,"in buffer")

    n=n-1
```

OUTPUT:

```
Enter  bucket  size,  outgoing  rate,  number  of  inputs  and
incoming size

300

50

2

200

Incoming size is 200

Bucket buffer size is    200 out of 300

After outgoing: 150 packets left out of 300 in buffer

Incoming size is 200

Packet loss: 50

Bucket buffer size is    300 out of 300

After outgoing: 250 packets left out of 300 in buffer
```

EXPERIMENT 10

10. Write a program to implement Dijkstra's algorithm to compute the shortest path through a graph.

PROGRAM:

```
INF = 1000


# Search minimum function

def search_min(length, se, n):

   global v

   mi = 100

   for i in range(n):

       if se[i] == 0:

           if length[i] < mi:

               mi = length[i]

               v = i

   return v


se = [0] * 10

length = []

path = []

graph = []


n = int(input("Enter No of Vertices: "))

print("Enter the adjacency matrix: ")
```

```
for i in range(n):
    graph.append(list(map(int, input().split())))

s = int(input("Enter Source node: "))

# INITIALIZATION PART
for i in range(n):
    if graph[s][i] == 0:
        length.append(INF)
        path.append(0)
    else:
        length.append(graph[s][i])
        path.append(s)

se[s] = 1
length[s] = 0

# ITERATION PART
c = 1
while c:
    c = 0
    j = search_min(length, se, n)
    se[j] = 1
    for i in range(n):
        if se[i] != 1:
            if graph[i][j] != 0:
                if length[j] + graph[i][j] < length[i]:
                    length[i] = length[j] + graph[i][j]
                    path[i] = j

    for i in range(0, n):
        if se[i] == 0:
            c += 1

# OUTPUT
print("From (source vertex) To ", s)
print("\tPath\t\tLength\t\tShortest path")

for i in range(n):
    if i != s:
        print("\t\t%d\t\t%d" % (i, length[i]), end='\t')
        j = i
        while j != s:
            print("\t%d->%d" % (j, path[j]), end='\t')
            j = path[j]
        print()
```

OUTPUT:

```
Enter No of Vertexes: 4
enter the adjacency matrix:
0 6 0 1
6 0 2 4
0 2 0 1
1 4 1 0
```

```
Enter Source node: 0

From(sourcevertex) To 0


 Path     Length    Shortest path

   1         4       1->2         2->3       3->0

   2         2       2->3         3->0

   3         1       3->0
```