Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No:1a **Date:** 12/08/2021

Client-server programming using TCP

Aim: Write a Socket program to implement client-server communication using TCP

Algorithm:-

a) Start

b) At the server side,

   a) Create TCP socket

   b) Bind the socket to the server address using bind()

   c) Using listen() the server is put in passive mode, where it waits for the client to approach the server to make a connection.

   d) Using accept() the connection is established between client and server and they are ready to transfer the data

3. At the server side,

   a) Create TCP socket

   b) Connect newly created client socket to the server, thus allowing them to transfer the data.

   c) Close socket descriptor.

4. Stop

Program

tcpserver.py

```
import socket
s=socket.socket()
port=12345
s.bind(('127.0.0.1',port))
print ("Binded to port #",port)
s.listen(5)
while True:
   print("Waiting for client...")
   k,a=s.accept()
   print ("Connected to",a)
   print("Message from client:", k.recv(15).decode())
   k.send('Welcome'.encode())
```

tcpclient.py

```
import socket
s=socket.socket()
port=12345
s.connect(('127.0.0.1',port))
s.send('Hallo Server!'.encode())
print ("Message from server:",s.recv(15).decode())
s.close()
```

Output

TCP Server

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~$ cd Desktop
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python tcpserver.py
Binded to port # 12345
Waiting for client...
Connected to ('127.0.0.1', 33498)
Message from client: Hallo Server!
Waiting for client...
```

TCP Client

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python tcpclient.py
Message from server: Welcome
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$
```

Result: Successfully implemented client-server programming using TCP

Remarks:(To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No:1b **Date:** 12/08/2021

Client-server programming using UDP

Aim: Write a Socket program to implement client-server communication using UDP

Algorithm:-

1. Start

2. At the server side,
   a) Create UDP socket
   b) Bind the socket to the server address using bind()
   a) The server waits until datagram packet arrives from the client.
   b) When the datagram arrives, the server process the packet and sends a reply to the client.
3. At the server side,
   a) Create UDP socket
   b) Connect client socket to the server.
   c) Sends a message to the server.
   d) Wait until response from the srever is received.
   e) Process the reply and repeat from step (c) if necessary.
   f) Close socket descriptor.
4. Stop

Program

udpserver.py

```
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
ip='127.0.0.1'
port=12345
s.bind((ip,port))
print ("Binded to port #",port)
while(True):
    print("Waiting for client...")
    d,a=s.recvfrom(1024)
    print ("Connected to",a)
    print ("Message from client:",d.decode())
    s.sendto('Welcome'.encode(),a)
```

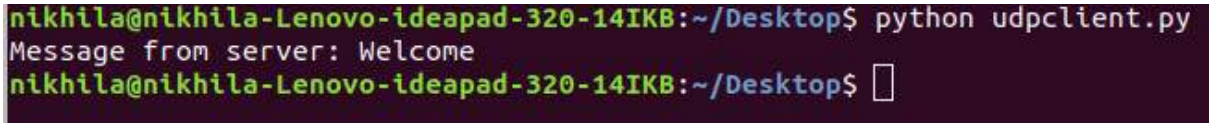
udpclient.py

```
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
 ip='127.0.0.1'
 port=12345
 s.connect((ip,port))
 s.sendto("Hallo Server!".encode(),(ip,port))
 print ("Message from server:", s.recvfrom(1024)[0].decode())
 s.close()
```

Output

UDP Server

UDP Client



Result: Successfully implemented client-server programming using UDP

Remarks:(To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No:2a **Date:** 26/08/2021

Single Chat

Aim: Implement a Client-Server program where the client sends a number and Server replies back whether it is an odd or even number.

Algorithm:-

a) Start

b) At the server side,

   a) Create TCP socket

   b) Bind the socket to the server address using bind()

   c) Using listen() the server is put in passive mode, where it waits for the client to approach the server to make a connection.

   d) Using accept() the connection is established between client and server and they are ready to transfer the data

   e) If the number is divisible by 2, send the message'Even Number' to the client otherwise send the message 'Odd Number'

3. At the client side,

   a) Create TCP socket

   b) Connect newly created client socket to the server, thus allowing the client to transfer the number to the server.

   c) Print the message send by the server.

   d) Close socket descriptor.

4. Stop

Program

singleserver.py

```
import socket
s=socket.socket()
port=12345
s.bind(('127.0.0.1',port))
print ("Binded to port #",port)
s.listen(5)
while True:
   print("Waiting for client...")
   k,a=s.accept()
   print ("Connected to",a)
   msg=k.recv(1024).decode()
```

```
    print("Message from client:", msg)
    a=msg.split()
    for i in a:
        if (i.isdigit() or (i[0]=="-" and i[1:].isdigit())):
            no=int(i)
    if (no%2==0):
       msg1=str(no)+' is an even number'
       k.send(msg1.encode())
       k.close()
    else:
      msg1=str(no)+' is an odd number'
      k.send(msg1.encode())
      k.close()
```

singleclient.py

```
import socket
s=socket.socket()
port=12345
s.connect(('127.0.0.1',port))
n=input("Enter a number:")
msg="Check whether "+n+" is even or odd"
s.send(msg.encode())
print ("Message from server:",s.recv(1024).decode())
s.close()
```

Output

Server

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~$ cd Desktop
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python singleserver.py
Binded to port # 12345
Waiting for client...
Connected to ('127.0.0.1', 45064)
Message from client: Check whether 12 is even or odd
Waiting for client...
Connected to ('127.0.0.1', 45066)
Message from client: Check whether 27 is even or odd
Waiting for client...
Connected to ('127.0.0.1', 45068)
Message from client: Check whether -5 is even or odd
Waiting for client...
Connected to ('127.0.0.1', 45076)
Message from client: Check whether 0 is even or odd
Waiting for client...
```

Client

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python singleclient.py
Enter a number:12
Message from server: 12 is an even number
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python singleclient.py
Enter a number:27
Message from server: 27 is an odd number
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python singleclient.py
Enter a number:-5
Message from server: -5 is an odd number
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python singleclient.py
Enter a number:0
Message from server: 0 is an even number
```

Result: Successfully implemented single chat

Remarks:(To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No:2b **Date:** 26/08/2021

MultiChat

Aim: Implement a multi chat program

Algorithm:-

a) Start

b) At the server side,

   a) Create TCP socket

   b) Bind the socket to the server address using bind()

   c) Using listen() the server is put in passive mode, where it waits for the client to approach the server to make a connection.

   d) Using accept() the connection is established between client and server and they are ready to transfer the data

   e) Create thread for each client and append the client to the client list.

   f) When client sends a message, the message is broadcasted to all other clients except to the client who is sending the messgae.

3. At the client side,

   a) Create TCP socket

   b) Connect newly created client socket to the server, thus allowing the client to transfer the messgae.

   c) To send the message to all other clients in the chatroom, send the message to the server

   d) Print the message send by the server, if any.

   e) Close socket descriptor when ‘l’ is entered

4. Stop

Program

singleserver.py

```
import socket
import threading

port= 5050
add = ('127.0.0.1', port)
leave_msg = "l"
active = True

server = socket.socket()
```

```
server.bind(add)

client_list = []

def serverside():
    server.listen(5)
    print(f"Server is listening...")
    while active:
        conn, addr = server.accept()
        client_list.append(conn)
        name=conn.recv(1024).decode()
        thread = threading.Thread(target=handle_client, args=(conn, name))
        thread.start()
        print(f"Number of active connections: {threading.active_count() - 1}")



def sendToAllClients(msg,conn):
    for client in client_list:
        if conn!=client:
          client.send(msg.encode())



def handle_client(conn, name):
    try:
        msg=f"{name} has joined the chat!"
        print(msg)
        sendToAllClients(msg,conn)
        while True:
            msg = conn.recv(1024).decode()

            if msg == leave_msg:
                break
            msg = (f"{name}:{msg}")
            print(msg)
            sendToAllClients(msg,conn)
        msg=f"{name} has left the chat!"
        print(msg)
```

```
            sendToAllClients(msg,conn)
            client_list.remove(conn)
            conn.close()

        except:
            return

    serverside()
```

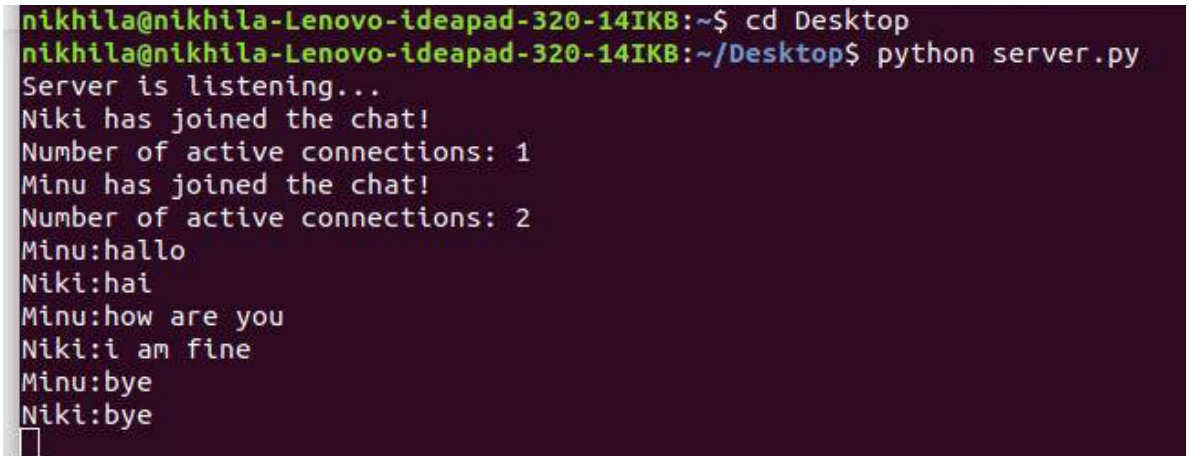
singleclient.py

```
import socket
import select
import sys
port = 5050
add = ('127.0.0.1', port)
leave_msg = "l"
active = True

client = socket.socket()
client.connect(add)
client.send(input("Enter name:").encode())
print("Sucessfully joined the chat!\nTo leave the chat enter 'l'\n")

 def send(msg):
        msg = msg.encode()
       client.send(msg)



 def receive():
        msg = client.recv(1024).decode()
       print(msg)



 while active:
         rlist, wlist, errlist = select.select([client, sys.stdin], [], [])
        for s in rlist:
```

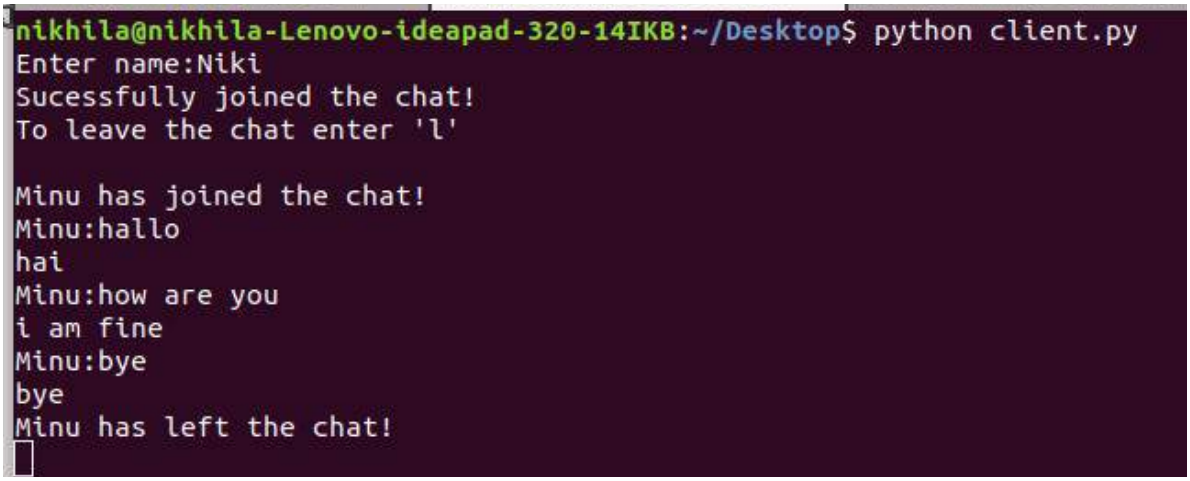
```
        if s == client:
            receive()
        else:
            msg = input()
            if msg == leave_msg:
                active = False
            send(msg)

send(msg)
```

Output

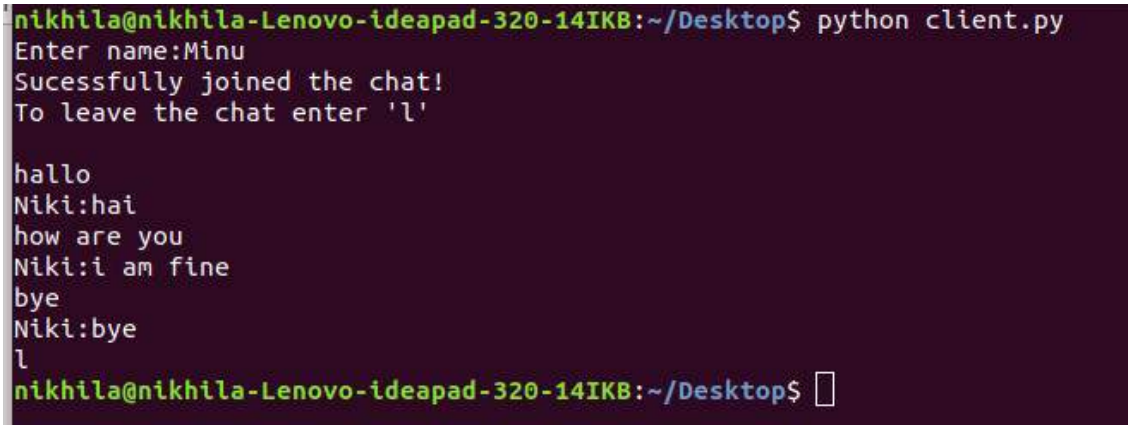
Server



Client 1



Client 2



Result: Successfully implemented multichat

Remarks:(To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No:3 **Date:** 09/09/2021

Error detection using CRC

Aim: Write a client server program to implement CRC for error detection. Implementation should include case:
1) with error
2) without error

Algorithm:-

a) Start

b) At the server side,

   a) Create TCP socket

   b) Bind the socket to the server address using bind()

   c) Using listen() the server is put in passive mode, where it waits for the client to approach the server to make a connection.

   d) Using accept() the connection is established between client and server and they are ready to transfer the data

   e) The server receives the encoded data string from the client.

   f) The server with the help of the key decodes the data and finds out the remainder by doing mod2 division.

   g) If the remainder is zero then server sends ‘No error’ message otherwise the server sends ‘Error found’ message to the client.

   h) Introduce an error by changing the last bit of the encoded data recieved by the client and repeat step (f) & (g)

3. At the client side,

   a) Create TCP socket

   b) Connect newly created client socket to the server, thus allowing the client to transfer the messgae.

   c) Input the data to be send to the server and convert it to its equivalent binary string.

   d) Append length of key-1 zeros to the binary string.

   e) The remainder of mod2 division of the appended data and the key is concatenated with the binary string. This is the encoded data.

   f) Send the encoded data to the server

   g) Print the messages send by the server.

4. Stop

Program

crcserver.py

```
import socket
def xor(a,b):
    ans=''
    for i in range(1,len(b)):
        if a[i]==b[i]:
            ans+='0'
        else:
            ans+='1'
    return ans

def div(dividend,divisor):
    pick=len(divisor)
    check=dividend[0:pick]
    while pick<len(dividend):
        if check[0]=='1':
            check=xor(divisor,check)+dividend[pick]
        else:
            check=xor('0'*pick,check)+dividend[pick]
        pick+=1
    if check[0]=='1':
       check=xor(key,check)
    else:
       check=xor('0'*pick,check)
    return check

def decodeData(data,key):
    new=data+'0'*(len(key)-1)
    rem=div(new,key)
    return rem

s=socket.socket()
port=12345
s.bind(('127.0.0.1',port))
print (''Binded to port #'',port)
s.listen(5)
```

```
while True:
    print("Waiting for client...")
    k,a=s.accept()
    print ("Connected to",a)
    data=k.recv(1024).decode()
    print("When no disturbance occured during transmission\nMessage from client:", data)
    key="1001"
    rem=decodeData(data,key)
    print("Remainder after decoding: ",rem)
    temp="0"*(len(key)-1)
    if rem==temp:
      msg="Data received is "+data+"; No error"
      k.send(msg.encode())
    else:
      msg="Data received is "+data+"; Error in data"
      k.send(msg.encode())

    if data[len(data)-1]=='1':
      data=data[:-1]
      data+="0"
    else:
      data=data[:-1]
      data+="1"
    print("When disturbance occured during transmission\nMessage from client:", data)
    key="1001"
    rem=decodeData(data,key)
    print("Remainder after decoding: ",rem)
    temp="0"*(len(key)-1)
    if rem==temp:
      msg="Data received is "+data+"; No error"
      k.send(msg.encode())
    else:
      msg="Data received is "+data+"; Error in data"
      k.send(msg.encode())
    k.close()
```

crcclient.py

```
import socket
def xor(a,b):
    ans=''
    for i in range(1,len(b)):
    if a[i]==b[i]:
       ans+='0'
    else:
      ans+='1'
    return ans

def div(dividend,divisor):
     pick=len(divisor)
     check=dividend[0:pick]
     while pick<len(dividend):
          if check[0]=='1':
             check=xor(divisor,check)+dividend[pick]
          else:
             check=xor('0'*pick,check)+dividend[pick]
          pick+=1
     if check[0]=='1':
        check=xor(key,check)
    else:
        check=xor('0'*pick,check)
    return check

def encodeData(data,key):
    new=data+'0'*(len(key)-1)
    rem=div(new,key)
    code=data+rem
    return code

s=socket.socket()
port=12345
s.connect(('127.0.0.1',port))
```

```
string=input("Enter the data you want to send: ")
data =(''.join(format(ord(x), 'b') for x in string))
key="1001"
msg=encodeData(data,key)
print("Data send:",msg)
s.send(msg.encode())
print ("Without disturbance\nMessage from server:",s.recv(1024).decode())
print ("With disturbance\nMessage from server:",s.recv(1024).decode())
s.close()
```

Output

Server (With and without error)

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python crcserver.py
Binded to port # 12345
Waiting for client...
Connected to ('127.0.0.1', 49452)
When no disturbance occured during transmission
Message from client: 10010001101001101
Remainder after decoding:  000
When disturbance occured during transmission
Message from client: 10010001101001100
Remainder after decoding:  001
Waiting for client...
```

Client (With and without error)

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python crcclient.py
Enter the data you want to send: Hi
Data send: 10010001101001101
Without disturbance
Message from server: Data received is 10010001101001101; No error
With disturbance
Message from server: Data received is 10010001101001100; Error in data
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$
```

Result: Successfully implemented error detection using CRC

Remarks:(To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No:4 **Date:** 16/09/2021

Hamming Code for error correction

Aim: Write a client server program to implement Hamming code using python.
Include 2 test cases
1) Data transmitted without error from client to server
2) Data transmitted with 1 bit error from client to server and then correct the error

Algorithm:-

a) Start

b) At the server side,

   a) Create TCP socket

   b) Bind the socket to the server address using bind()

   c) Using listen() the server is put in passive mode, where it waits for the client to approach the server to make a connection.

   d) Using accept() the connection is established between client and server and they are ready to transfer the data

   e) The server receives the encoded data string from the client.

   f) Calculate the redundant bit using the length of the encoded data.

   g) Recalculate each redundant bits of the encoded data and find the decimal equivalent of the parity bits binary value. This is the position of error.

   h) If position is zero the server sends ‘No error’ message otherwise the server sends ‘Error found’ message along with corrected data to the client.

   i) Introduce an error by changing a random bit of the encoded data recieved by the client and repeat step (f) to (h)

3. At the client side,

   a) Create TCP socket

   b) Connect newly created client socket to the server, thus allowing the client to transfer the messgae.

   c) Input the data to be send to the server and convert it to its equivalent binary string.

   d) Calculate the number of redundant bits using the formula 2^r>=m+r+1, where r is no. of redundant bits and m is the no. of data bits.

   e) Find the positions of redundant bits i.e. if the position is power of 2 insert ‘0’ as redundant bit otherwise append the data.

   f) Calculate the values of each redundant /parity bit

   g) Send the encoded data to the server

   h) Print the messages send by the server.

4. Stop

Program

hamserver.py

```
import socket
import random

def calRedundantBits(m):
    for i in range(m):
        if(2**i >= m):
            return i

def detectError(arr, nr):
    n = len(arr)
    res = 0
    for i in range(nr):
        val = 0
        for j in range(1, n + 1):
            if(j & (2**i) == (2**i)):
                val = val ^ int(arr[-1 * j])
        res = res + val*(10**i)
    if (res==0):
        return (int(str(res),2))
    else:
        return (n-int(str(res), 2)+1)

s=socket.socket()
port=12345
s.bind(('127.0.0.1',port))
print ("Binded to port #",port)
s.listen(5)
while True:
    print("Waiting for client...")
    k,a=s.accept()
    print ("Connected to",a)
    data=k.recv(1024).decode()
    print("When no disturbance occured during transmission\nMessage from client:", data)
    m=len(data)
    nr=calRedundantBits(m)
```

```
    correction = detectError(data, nr)
    if (correction==0):
      msg="Data received is "+data+"; No error"
      k.send(msg.encode())
    else:
      data1="
      if (data[correction-1]=='0'):
        data1=data[:correction-1]+"1"+data[correction:]
      else:
        data1=data[:correction-1]+"0"+data[correction:]
      msg="Data received is "+data+"; Error found at position "+str(correction)+"; Corrected data
is "+data1
      k.send(msg.encode())


    rand=random.randint(1,m)
    ndata="
    if (data[rand-1]=='0'):
      ndata=data[:rand-1]+"1"+data[rand:]
    else:
      ndata=data[:rand-1]+"0"+data[rand:]
    print("When disturbance occured during transmission\nMessage from client:", ndata)
    correction = detectError(ndata, nr)
    if (correction==0):
      msg="Data received is "+ndata+"; No error"
      k.send(msg.encode())
    else:
      data1="
      if (ndata[correction-1]=='0'):
        data1=ndata[:correction-1]+"1"+ndata[correction:]
      else:
        data1=ndata[:correction-1]+"0"+ndata[correction:]
      msg="Data received is "+ndata+"; Error found at position (from left): "+str(correction)+";
Corrected data is "+data1
      k.send(msg.encode())
```

hamclient.py

```
import socket
def calcRedundantBits(m):
    for i in range(m):
    if (2**i >= m + i + 1):
        return i


def posRedundantBits(data, r):
    j = 0
    k = 1
    m = len(data)
    res = ''
    for i in range(1, m + r+1):
        if(i == 2**j):
            res = res + '0'
            j += 1
        else:
            res = res + data[-1 * k]
            k += 1
    return res[::-1]


def calcParityBits(arr, r):
     n = len(arr)
     for i in range(r):
          val = 0
          for j in range(1, n + 1):
               if(j & (2**i) == (2**i)):
                    val = val ^ int(arr[-1 * j])
          arr = arr[:n-(2**i)] + str(val) + arr[n-(2**i)+1:]
     return arr


s=socket.socket()
port=12345
s.connect(('127.0.0.1',port))
```

```
string=input("Enter the data you want to send: ")
data =(''.join(format(ord(x), 'b') for x in string))
print("Binary format of the data: ",data)
n = len(data)
r = calcRedundantBits(n)
arr = posRedundantBits(data, r)
arr = calcParityBits(arr, r)
print("Data transferred: " + arr)
s.send(arr.encode())
print ("Without disturbance\nMessage from server:",s.recv(1024).decode())
print ("With disturbance\nMessage from server:",s.recv(1024).decode())
s.close()
```

Output

Server (With and without error)

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python hamserver.py
Binded to port # 12345
Waiting for client...
Connected to ('127.0.0.1', 40380)
When no disturbance occured during transmission
Message from client: 11001111001
When disturbance occured during transmission
Message from client: 11001101001
Waiting for client...
```

Client (With and without error)

```
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$ python hamclient.py
Enter the data you want to send: n
Binary format of the data:  1101110
Data transferred: 11001111001
Without disturbance
Message from server: Data received is 11001111001; No error
With disturbance
Message from server: Data received is 11001101001; Error found at position (from
 left): 7; Corrected data is 11001111001
nikhila@nikhila-Lenovo-ideapad-320-14IKB:~/Desktop$
```

Result: Successfully implemented error detection using Hamming Code

Remarks:(To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No: 6 **Date:** 25/10/2021

Playfair Cipher

Aim: Implement polyalphabetic cipher- Playfair Cipher

Algorithm: -

1. Start
2. At the server side,
   a) Create TCP socket
   b) Bind the socket to the server address using bind()
   c) Using listen() the server is put in passive mode, where it waits for the client to approach the server to make a connection.
   d) Using accept() the connection is established between client and server and they are ready to transfer the data
   e) The server receives the plaintext and the key from the client.
   f) For encrypting, do from step i
      i. Convert the case of the plaintext and the key as lower.
      ii. Set the plaintext by replacing all the J's of the plaintext to I's and add a filter letter x either when there exist same letters in pair or when the length of plaintext is odd.
      iii. Set cipher box as a 5*5 key square matrix by initially filling it with non-repeating letters of the keyword row wise and fill the rest in alphabetic order.
      iv. Create a dictionary of index value of each letter in cipher box.
      v. Take plaintext pairwise till the length of the plaintext and do the following:
         - If both letters belong to the same row in the matrix append cipher text with letters to its right
         - If both the letters belong to same column in the matrix append the cipher text with letters to their bottom
         - If neither of the above, form a rectangle with the two letters and append the cipher text with letters on the horizontal opposite corner of the rectangle.
   g) Send the cipher text to the client
3. At the client side,
   a) Create TCP socket
   b) Connect newly created client socket to the server, thus allowing the client to transfer the message.
   c) Input the plaintext to be encrypted and the keyword and send the same to the server.
   d) Print the encrypted text send by the server.
4. Stop

Program

server.py

```
import socket
def setCipherBox(keyword):
    alphabet = [1 for i in range(26)]
    i, j = 0, 0
    cipherbox = [[0 for i in range(5)] for j in range(5)]
    for k in keyword:
        if j == 5:
            i += 1
            j = 0
        if k == 'j':
            k = 'i'
        idx = ord(k) - ord('a')
        if alphabet[idx] == 1:
            cipherbox[i][j] = k
            alphabet[idx] = 0
            j += 1
    for k in range(ord('a'), ord('z')+1):
        if(chr(k) == 'j'):
            continue
        idx = k - ord('a')
        if(alphabet[idx] == 1):
            if j == 5:
                i += 1
                j = 0
            cipherbox[i][j] = chr(k)
            j += 1
    return cipherbox

def setCipherDict(cipherbox):
    cipherDict = {}
    for i in range(len(cipherbox)):
        for j in range(len(cipherbox[i])):
            ch = cipherbox[i][j]
            cipherDict[ch] = [i, j]
    return cipherDict
```

```
def setPlaintext(plaintext):
    i, j = 0, 1
    plaintext = plaintext.lower().replace('j', 'i')
    while j < len(plaintext):
        if plaintext[i] == plaintext[j]:
            if plaintext[i] != 'z':
                plaintext = plaintext[:j] + 'z' + plaintext[j:]
            else:
                plaintext = plaintext[:j] + 'y' + plaintext[j:]
        i += 2
        j += 2
    if i < len(plaintext):
        if plaintext[-1] != 'z':
            plaintext += 'z'
        else:
            plaintext += 'y'
    return plaintext

def encrypt(plaintext, keyword):
    if(not plaintext or not keyword):
        return ("plaintext or keyword invalid")
    plaintext = setPlaintext(plaintext)
    keyword = keyword.lower()
    ciphertext = ""
    cipherbox = setCipherBox(keyword)
    cipherDict = setCipherDict(cipherbox)
    i, j = 0, 1
    while i < len(plaintext):
        ch1 = {
            "x": cipherDict[plaintext[i]][0],
            "y": cipherDict[plaintext[i]][1]
        }
        ch2 = {
            "x": cipherDict[plaintext[j]][0],
            "y": cipherDict[plaintext[j]][1]
        }
```

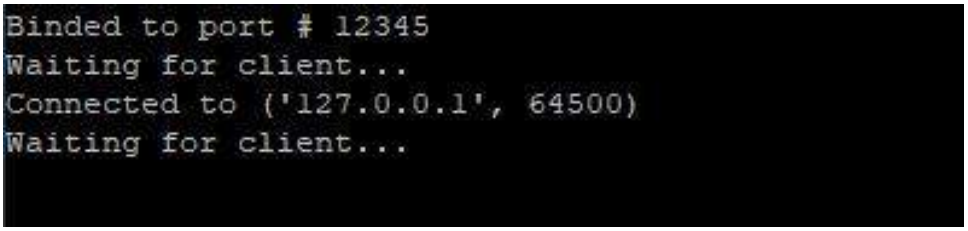
```
        if ch1["x"] == ch2["x"]:
            ciphertext += cipherbox[ch1["x"]][(ch1["y"] + 1) % 5]
            ciphertext += cipherbox[ch2["x"]][(ch2["y"] + 1) % 5]
        elif ch1["y"] == ch2["y"]:
            ciphertext += cipherbox[(ch1["x"] + 1) % 5][ch1["y"]]
            ciphertext += cipherbox[(ch2["x"] + 1) % 5][ch2["y"]]
        else:
            ciphertext += cipherbox[ch1["x"]][ch2["y"]]
            ciphertext += cipherbox[ch2["x"]][ch1["y"]]
        i += 2
        j += 2
    return ciphertext

s=socket.socket()
port=12345
s.bind(('127.0.0.1',port))
print ("Binded to port #",port)

s.listen(5)

while True:
        print("Waiting for client...")
        k,a=s.accept()
        print ("Connected to",a)
        msg=k.recv(1024).decode()
        a=msg.split()
        ciphertext = encrypt(a[0], a[1])
        k.send(ciphertext.encode())
        k.close()
```

client.py

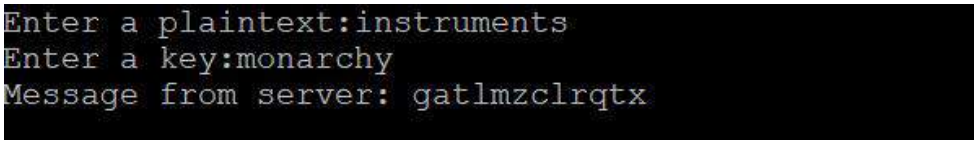
```
import socket
s=socket.socket()
port=12345
s.connect(('127.0.0.1',port))
plaintext=input("Enter a plaintext:")
key=input("Enter a key:")
msg=plaintext+" "+key
s.send(msg.encode())
print ("Message from server:",s.recv(1024).decode())
s.close()
```

Output

Server



Client



Result: Successfully implemented Playfair cipher

Remarks: (To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No: 7 **Date:** 01/11/2021

Transposition Cipher

Aim: Implement transposition cipher algorithm- Rail Fence Cipher

Algorithm: -

1. Start
2. Input the plain text to be encrypted and the key value.
3. Create a matrix having the number of rows as the key value and number of columns as the length of the plain text.
4. Initialize the variables ‘row’, ‘col’ and ‘i’ as 0.
5. Also initialize the variables for direction ‘down’ as False.
6. If the value of i is less than the length of the plain text, do steps 7 to 11
7. If it is the first or the last row i.e row=0 or row=key-1, reverse the direction by negating the variable ‘down’
8. Fill the particular cell of the matrix with the corresponding alphabets.
9. Increment the col.
10. Increment the row if down is True, else decrement it.
11. Increment the value of ‘i’.
12. Initialize an empty list ‘cipher’ and ‘i’ as 0
13. If the value of i is less than the key value do the steps 14 to
14. Initialize the variable ‘j’ as 0.
15. If the value of j is less than the length of the plain text do from steps 16 to
16. If the value of the matrix in the corresponding row and column is not ‘\n’, then append the value to the cipher list.
17. Increment j and i
18. The list ‘cipher’ is the encrypted message, display it.
19. Stop

Program

```
def RailFence(text, key):
    rail = [['\n' for i in range(len(text))]
            for j in range(key)]
    down = False
    row, col = 0, 0

    for i in range(len(text)):
        if (row == 0) or (row == key - 1):
            down = not down
        rail[row][col] = text[i]
        col += 1
        if down:
            row += 1
        else:
            row -= 1

    encrypt = []
    for i in range(key):
        for j in range(len(text)):
            if rail[i][j] != '\n':
                encrypt.append(rail[i][j])
    return("" . join(encrypt))

print("Rail Fence Cipher (Encryption)")
text = input("Enter the plain text: ")
key = int(input("Enter the key value: "))
print("Encrypted text: ", RailFence(text, key))
```

Output

```
PS C:\Users\USER\Desktop\S7\np lab> python 2.py
Rail Fence Cipher (Encryption)
Enter the plain text: Hallo World
Enter the key value: 3
Encrypted text:  Horal ollWd
PS C:\Users\USER\Desktop\S7\np lab> python 2.py
Rail Fence Cipher (Encryption)
Enter the plain text: Attack At Once
Enter the key value: 2
Encrypted text:  Atc tOctakA ne
```

Result: Successfully implemented Rail Fence cipher

Remarks: (To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:**61

Exp. No:8 **Date:** 05/11/2021

LZW Compression and Decompression

Aim: Write a program to implement lzw compression and decompression.

ALGORITHM

Compression:

1. Start
2. Accept the string to be compressed
3. Initialize table with single character strings
4. Let P = first input character
5. WHILE not end of input stream
    a. C = next input character
    b. IF P + C is in the string table
        i. P = P + C
    c. ELSE
        i. output the code for P
    d. add P + C to the string table
    e. P = C
6. output code for P
7. Stop

Decompression:

1. Start
2. Initialize table with single character strings
3. OLD = first input code
4. output translation of OLD
5. WHILE not end of input stream
    a. NEW = next input code
    b. IF NEW is not in the string table
        i. S = translation of OLD
        ii. S = S + C
    c. ELSE
        i. S = translation of NEW
    d. output S
    e. C = first character of S
    f. OLD + C to the string table
    g. OLD = NEW
6. Stop

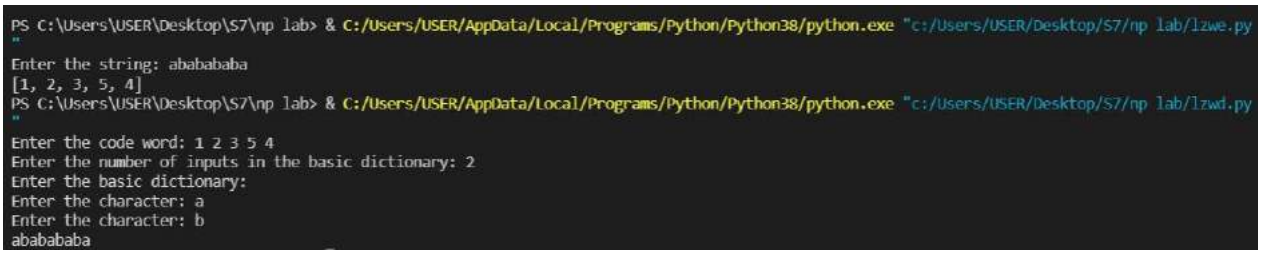
PROGRAM
Compression

```
import sys
from sys import argv
from struct import *

uncompressed=raw_input("Enter the string: ")

dict_size = 1
char_seen = []
dictionary={}
for char in uncompressed:
    if char not in char_seen:
        char_seen.append(char)
string=''.join(char_seen)
for i in string:
    dictionary[i] = dict_size
    dict_size+=1

w = ""
result = []
for c in uncompressed:
    wc = w + c
    if wc in dictionary:
       w = wc
    else:
       result.append(dictionary[w])
       dictionary[wc] = dict_size
       dict_size += 1
       w = c

    # Output the code for w.
if w:
   result.append(dictionary[w])
print(result)
```

Decompression

```
code=raw_input("Enter the code word: ")
compressed=code.split(" ")
map_object=map(int,compressed)
compressed=list(map_object)
dictionary={}
dict_size=1
n=input("Enter the number of inputs in the basic dictionary: ")
print("Enter the basic dictionary: ")
for i in range(n):
    c=raw_input("Enter the character: ")
    dictionary[dict_size]=c
    dict_size+=1

result = ""
w = dictionary[compressed.pop(0)]
result+=w
for k in compressed:
    if k in dictionary:
       entry = dictionary[k]
    elif k == dict_size:
       entry = w + w[0]
    else:
       raise ValueError('Bad compressed k: %s' % k)

    result+=entry
    dictionary[dict_size] = w + entry[0]
    dict_size += 1
    w = entry

print(result)
```

OUTPUT



Result: Successfully implemented compression and decompression using lzw

Remarks:(To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No: 9 **Date:** 12/11/2021

Topology creation using NS2

Aim: Create a 4-node fully connected network topology using NS2

Algorithm: -

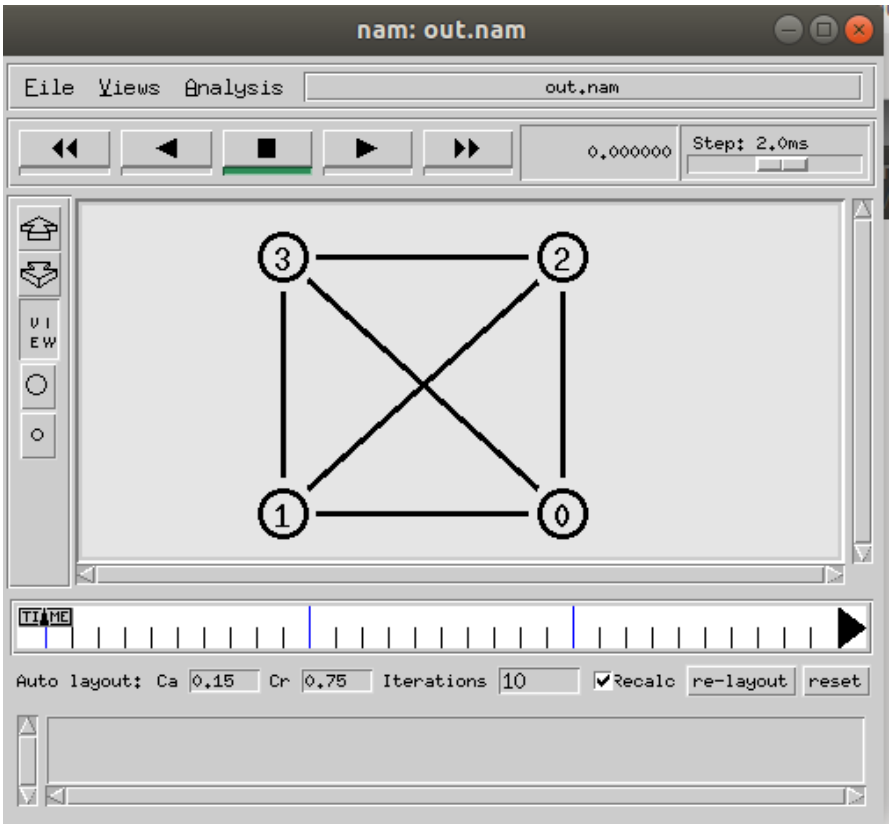
1. Start
2. Create a new simulator object.
3. Open the nam trace file.
4. Open the trace file.
5. Define a 'finish' procedure.
6. Close the trace files.
7. Execute nam on the trace file.
8. Create four nodes.
9. Create 6 duplex link between four nodes.
10. Call finish procedure.
11. Run the simulation.
12. Stop

Program

```
#create a new simulator object
set ns [new Simulator]

#open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#open the trace file
set tf [open out.tr w]
$ns trace-all $tf

#define a 'finish' procedure
proc finish {} {
global ns nf tf
$ns flush-trace

#close the trace files
close $nf
close $tf


#execute nam on the trace file
exec nam out.nam &
exit 0
}
```

```
#create two nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n3 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

#call finish procedure
$ns at 5.0 "finish"
#run the simulation
$ns run
```

Output



Result: Successfully created a 4-node fully connected network topology using NS2

Remarks: (To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No: 10 **Date:** 12/11/2021

TCP Stimulation using NS2

Aim: Stimulate TCP using NS2

Algorithm: -

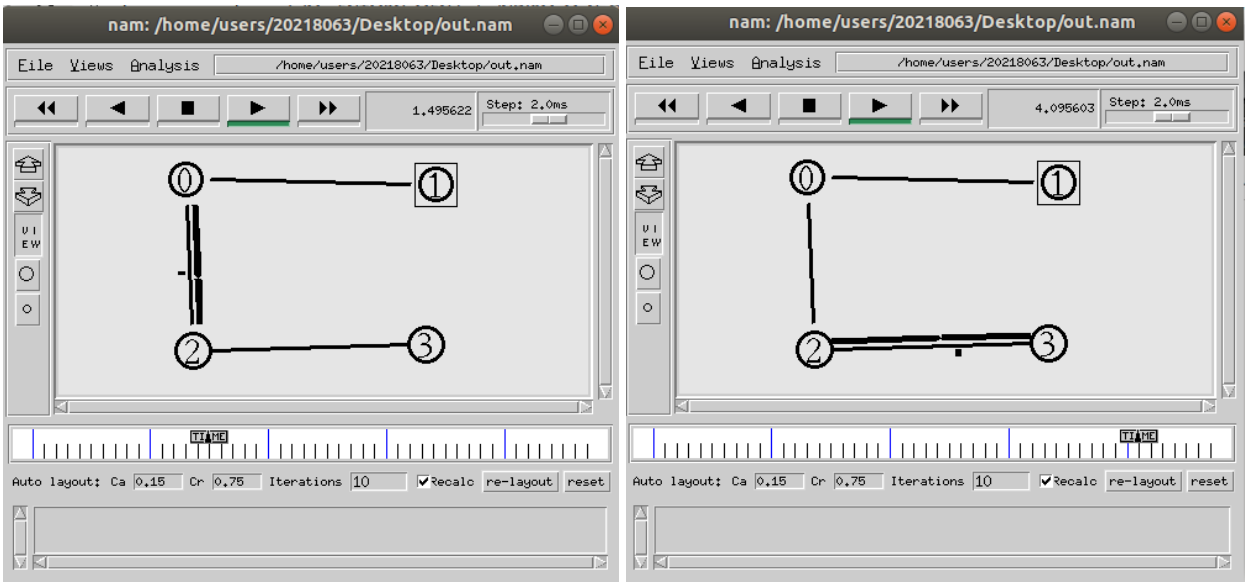
1. Start
2. Create a new simulator object.
3. Open the nam trace file.
4. Open the trace file.
5. Define a 'finish' procedure.
6. Close the trace files.
7. Execute nam on the trace file.
8. Create four nodes n0, n1, n2, n3.
9. Create 3 duplex link between nodes n0 & n1; n0 & n2 and n2 & n3.
10. Create a udp agent and attach it nodes n0 and n2.
11. Create a Null agent (a traffic sink) and attach the node 2 and 3.
12. Connect the traffic source to the sink.
13. Create a CBR traffic source and attach it to udp0 and udp2.
14. Schedule two event for CBR traffic.
15. Call the finish procedure after 5 secs of simulated time.
16. Run the simulation.
17. Call finish procedure.
18. Run the simulation.
19. Stop

Program

```
#create a new simulator object
set ns [new Simulator]

#open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#open the trace file
set tf [open out.tr w]
$ns trace-all $tf

#define a 'finish' procedure
proc finish {} {
global ns nf tf
$ns flush-trace
```

```
#close the trace files
close $nf
close $tf

#execute nam on the trace file
exec nam out.nam &
exit 0
}

#create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

#create a tcp agent and attach it to node 0 and node 2
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2

#create a traffic sink and attach it to node 2 and node 3
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink2
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3

#Connect the traffic source to the sink2 and sink3
$ns connect $tcp0 $sink2
$ns connect $tcp2 $sink3

#Create a FTP traffic source and attach it to tcp0 and tcp2
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp2

#Schedule two events for FTP traffic
$ns at 0.5 "$ftp0 start"
$ns at 3.5 "$ftp0 stop"
$ns at 3.5 "$ftp1 start"
$ns at 4.5 "$ftp1 stop"

#call the finish procedure after 5 secs of simulated time
$ns at 5.0 "finish"

#run the simulation
$ns run
```

Output



Result: Successfully simulated TCP using NS2

Remarks: (To be filled by faculty)

Name: Nikhila Elizabeth Shaji **Roll No:** 61

Exp. No: 11 **Date:** 12/11/2021

UDP Stimulation using NS2

Aim: Stimulate UDP using NS2

Algorithm: -

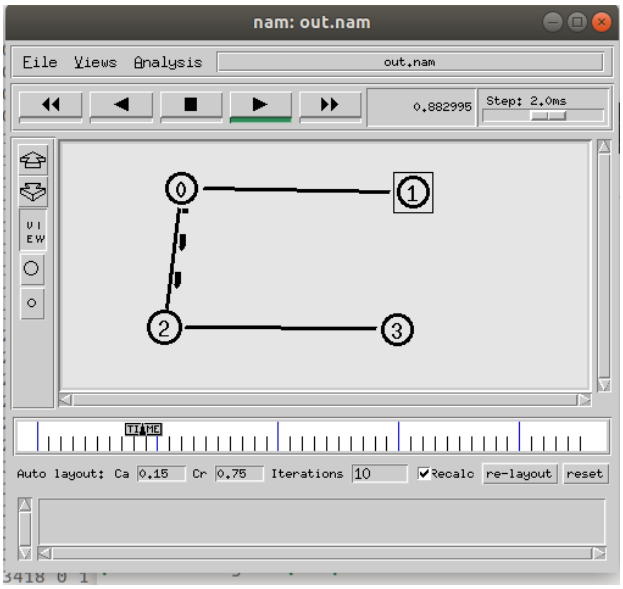
1. Start
2. Create a new simulator object.
3. Open the nam trace file.
4. Open the trace file.
5. Define a 'finish' procedure.
6. Close the trace files.
7. Execute nam on the trace file.
8. Create four nodes n0, n1, n2, n3.
9. Create 3 duplex link between nodes n0 & n1; n0 & n2 and n2 & n3.
10. Create a udp agent and attach it nodes n0 and n2.
11. Create a Null agent (a traffic sink) and attach the node 2 and 3.
12. Connect the traffic source to the sink.
13. Create a CBR traffic source and attach it to udp0 and udp2.
14. Schedule two event for CBR traffic.
15. Call the finish procedure after 5 secs of simulated time.
16. Run the simulation.
17. Call finish procedure.
18. Run the simulation.
19. Stop

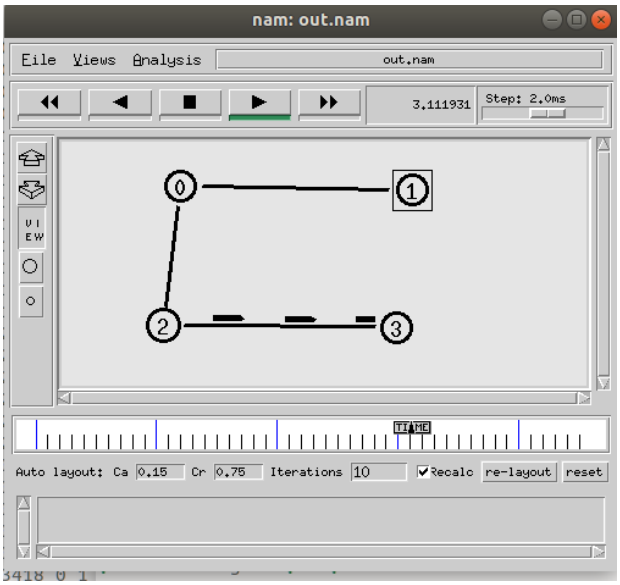
Program

```
#create a new simulator object
set ns [new Simulator]

#open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#open the trace file
set tf [open out.tr w]
$ns trace-all $tf

#define a 'finish' procedure
proc finish {} {
global ns nf tf
$ns flush-trace
```

```
#close the trace files
close $nf
close $tf

#execute nam on the trace file
exec nam out.nam &
exit 0
}

#create two nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

#create a udp agent and attach it to node 0 and 2
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2

#create a Null agent(a traffic sink) and attach it to node 2 and 3
set null2 [new Agent/Null]
$ns attach-agent $n2 $null2
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3

#Connect the traffic source to the sink
$ns connect $udp0 $null2
$ns connect $udp2 $null3

#Create a CBR traffic source and attach it to udp0 and udp2
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2

#Schedule events for CBR traffic
$ns at 0.5 "$cbr0 start"
$ns at 3.0 "$cbr0 stop"
$ns at 3.0 "$cbr2 start"
$ns at 4.5 "$cbr2 stop"

#call the finish procedure after 5 secs of simulated time
$ns at 5.0 "finish"

#run the simulation
$ns run
```

Output





Result: Successfully simulated UDP using NS2

Remarks: (To be filled by faculty)