Ex No : 1 **IMPLEMENTATION OF ERROR DETECTION / ERROR**

Date : 14/09/2021 **CORRECTION**

AIM:

To implement the error correction technique using hamming code.

OBJECTIVE:

1. To learn about the concept Hamming Code
2. To understand error correction technique

SOFTWARE REQUIRED:

Turbo C++/Text Editor and GCC complier

ALGORITHM:

ERROR DETECTION :

STEP1 : include Standard input and output file and string files.

STEP 2: define strlen() to find the length of the length of the string and store it in variable 'N'.

STEP 2: Initialize char variables s, t, and g of required size.

STEP 4 : Initialize integer variables a, e and c.

STEP 5: Create void module "xor" as global function so that it can be called in main function.

STEP 51: In for loop, when "c=1, then C < N, and C++" the condition

STEP 5.2: In 's' char array "c" as increment variable check "s[c] == g[c]? 0:1" to get XoR operation truth table

STEP 6 : Create void module "crc" as global function.

STEP 6.1 In for loop, when e=o, e < N, e++ is the condition.

STEP 6.2: Initialize s[e] = t[e].

STEP 6.3 In do operation, check the condition if (s[0]== '1') then call xor operation, now in for conditional loop (c=0, c < N, c++) and initialize 's[c] = t[e++]', where in while checks "e <= a + N -1".

STEP 7: In ‘int main ()’ main function.

STEP 7.1: print “Enter the message” as store the message in variable “g”.

STEP 7.2: Now print “Enter the Data" and store the data in variable 't'.

STEP 7 3: Now , print “Generating Polynomial” of the value ‘t’.

STEP 7 4 : Get the string length of 't' and store it in variable ‘a’.

STEP 7.5: Using for loop “(e = a; e < a+N-1; e++)” and 'zeros to “t[e]”

STEP 7.6: Now print the Modified data.

STEP 7.7: Now, call the function "crc" and print the "crc" word.

STEP 7.8: In for loop for loop “(e = a; e < a+N-1; e++)” initialize “t[e] = s[e-a]” and now print the final codeword ‘t’.

STEP 7.9: To test error detection get ‘0’ or ‘1’ and save it in the address of ‘e’.

STEP 7.10: In if condition ‘e == 0’ :

STEP 7.10.1: In do operation, print "Enter the position where error is to be inserted: " and store it in the address of ‘e’.

STEP 7.11: In while loop ‘(e == 0 || e <= a+N-1)’ and initialize ‘t[e-1] = ((t[e-1] == '0')? ‘1’ : ‘0’)

STEP 7.12 : Print “Erroneous Data" in the variable in ‘t’.

STEP 8: Stop .

PROGRAM :

ERROR DETECTION :

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)
char s[28],t[28],g[28];
int a,e,c;
void xor()
{
   for(c=1;c<N;c++)
   s[c]=((s[c]==g[c]?'0':'1'));
}
void crc()
{
   for(e=0;e<N;e++)
   s[e]=t[e];
   do{
      if(s[0]=='1')
      xor();
      for(c=0;c<N-1;c++)
      s[c]=t[e++];
   }
   while(e<=a+N-1);
}
int main()
{
   printf("\n Enter The Message:");
   scanf("%s",g);
   printf("\nEnter Data:");
   scanf("%s",t);
   printf("\n--------------------");
   printf("\nGenerating Polynomial:%s”,t);
```

```
    a=strlen(t);
    for(e=a;e<a+N-1;e++)
    t[e]='0';
    printf("\n--------------------");
    printf("\nModified Data is:%s",t);
    printf("\n--------------------");
    crc();
    printf("\nChecksum is :%s",s);
    for(e=a;e<a+N-1;e++)
    t[e]=s[e-a];
    printf("\n---------------------");
    printf("\nFinal codeword is :%s",t);
    printf("\n--------------------");
    printf("\nTest erro detection 0(yes) 1(no)?:”);
    scanf("%d",&e);
    if(e==0)
    {
       do{
          printf("\nEnter the position where error is to be inserted:");
          scanf("%d",&e);
       }
       while(e==0||e>a+N-1);
       t[e-1]=((t[e-1]=='0')?'1':'0');
       printf("\n--------------------");
       printf("\n--------------------");
       printf("\nErroneous Data:%s\n",t);
    }
}
```

OUTPUT :

ERROR DETECTION :

ErrorDetection.c

```
macbook@macbooks-MacBook-Pro CN Lab Programs % gcc -o ErrorDetection.out ErrorDetection.c
macbook@macbooks-MacBook-Pro CN Lab Programs % ./ErrorDetection.out

 Enter The Message:Hello

Enter Data:12

--------------------
Generating Polynomial:12
--------------------
Modified Data is:120000
--------------------
Checksum is :0
----------------------
Final codeword is :120
--------------------
Test erro detection 0(yes) 1(no)?:0

Enter the position where error is to be inserted:2

--------------------
--------------------
Erroneous Data:100
macbook@macbooks-MacBook-Pro CN Lab Programs %
```

ALGORITHM :

ERROR CORRECTION :

STEP1: Create the variable that are needed for the program and their own respective data type.

STEP2:Encode the data to be encoded

STEP3:Decode the data to be decoded

STEP4:Based upon the status the printf statement will be printed.

STEP5:Check all the conditions.

STEP6:End the program.

PROGRAM :

ERROR CORRECTION :

```
#include<stdio.h>
int main()
{
    int data[7],rec[7],i,c1,c2,c3,c;
    printf("\nThis works for message of 4 bits in size\nEnter message bit one by one:");
    scanf("%d%d%d%d",&data[0],&data[1],&data[2],&data[4]);
    data[6]=data[0]^data[2]^data[4];
    data[5]=data[0]^data[1]^data[4];
    data[3]=data[0]^data[1]^data[2];
    printf("\nThe encoded bits are given below:\n");
    for(i=0;i<7;i++){
        printf("%d",data[i]);
    }
    printf("\nEnter the received data bits one by one:");
    for(i=0;i<7;i++){
        scanf("%d",&rec[i]);
    }
    c1=rec[6]^rec[4]^rec[2]^rec[0];
    c2=rec[5]^rec[4]^rec[1]^rec[0];
    c3=rec[3]^rec[2]^rec[1]^rec[0];
    c=c3*4+c2*2+c1;
    if(c==0){
        printf("\nCongratulations there is no erro:");
    }
    else{
        printf("\nError on the position:%d\nThe correct message is \n",c);
        if(rec[7-c]==0)
            rec[7-c]=1;
        else
            rec[7-c]=0;
        for(i=0;i<7;i++){
            printf("%d",rec[i]);
        }
    }
    return 0;
}
```

OUTPUT :

ERROR CORRECTION :

ErrorCorrection.c

```
macbook@macbooks-MacBook-Pro CN Lab Programs % gcc -o ErrorCorrection.out ErrorCorrection.c
macbook@macbooks-MacBook-Pro CN Lab Programs % ./ErrorCorrection.out

This works for message of 4 bits in size
Enter message bit one by one:1
0
1
1

The encoded bits are given below:
1010101
Enter the received data bits one by one:1
0
1
0
1
0
1

Congratulations there is no erro:%
macbook@macbooks-MacBook-Pro CN Lab Programs % ./ErrorCorrection.out

This works for message of 4 bits in size
Enter message bit one by one:1
0
0
1

The encoded bits are given below:
1001100
Enter the received data bits one by one:1
0
0
1
0
0
0

Error on the position:3
The correct message is
1001100%
macbook@macbooks-MacBook-Pro CN Lab Programs %
```

RESULT:

Thus, the implementation of error correction and detection using hamming code was successfullyexecuted.

Ex No : 2 **IMPLEMENTATION OF STOP AND WAIT PROTOCOL**

Date : 18/09/2021

AIM:

To implement a stop and wait protocol python

OBJECTIVE:

To learn how to implement a stop and wait protocol.

SOFTWARE REQUIRED:

Idle python 3.7.4/ Text Editor and python 3.7.4 interpreter software

ALGORITHM:

STOP AND WAIT SERVER :

STEP 1: Import Socket

STEP 2: Assign Host ID and Port Number so that client and server connect by IP.

STEP 3: Create an socket at ‘s’ with required parameter .

STEP 4 : Give command to combine the host port Number using ‘s.bind (HOST, PORT)’

and give command ‘s.listen()’ to wait for connection.

STEP 5: With Prebuilt function "conn", after getting connection, print by “connected by! the variable “addr”

STEP 6 : In While condition get the data from client upto required limit of bytes if the client input is “q” or “Q” Close the loop then; Else print the message

"1. SEND ACK

2. TIME OUT"

and get the option from the client as", to

Continue or else consider “time out” as “send_data”.

STEP 7 : End.

STOP AND WAIT CLIENT :

STEP 1 : Import Socket

STEP 2: Assign “HOST ID” and “Port Number” same as that of the server to connect with server.

STEP 3: Create socket as ‘s’ with required parameter to connect with server.

STEP 4 : Give command to connect to serve using HOST and PORT number by “s.connect (HOST, PORT))”.

STEP 5: In while condition, using print function, get the message which is to be sent to server and store it in "send_data", now send the message by encoding using the command “s.sendall (send_data.encode())”. Now, if “send-data == “q” or “Q”, close the loop and exit.

STEP 6: In while condition initialize “recv_data = s.recv(1024)”;

STEP 6.1: In it condition, “recv_data = b‘q’ or recv_data == b"Q"):" Close the Socket and exit loop.

STEP 6.2: elif (recr_data = b”timeout"): then print "ack is not received" and “s.sendall (send_data. encode())”.

STEP 6.3: oh else, print “received acknowledgment” and break the loop

STEP 7: End

PROGRAM :

STOP AND WAIT SERVER :

```
import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT))
        s.listen()
        conn, addr = s.accept()
        with conn:
                print('Connected by', addr)
                while True:
                        recv_data=conn.recv(1024)
                        if(recv_data==b"q" or recv_data=="Q"):
                                s.close()
                                exit(0)
                        else:
                                print("\n1.SEND ACK\n2.TIME OUT\n")
                                opt=input("Enter Option:\n")
                                if(opt=="1"):
                                        send_data="ack"
                                        print("\n RECEIVED DATA=",recv_data )
                                else:
                                  send_data="timeout";
                                conn.sendall(send_data.encode())
```

OUTPUT :

STOP AND WAIT SERVER :

StopAndWaitServer.py | StopAndWaitClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % python3 StopAndWaitServer.py
Connected by ('127.0.0.1', 49679)

1.SEND ACK
2.TIME OUT

Enter Option:
1

 RECEIVED DATA= b'Bonjour'

1.SEND ACK
2.TIME OUT

Enter Option:
2

1.SEND ACK
2.TIME OUT

Enter Option:
1

 RECEIVED DATA= b'Auf Widersen'

1.SEND ACK
2.TIME OUT

Enter Option:
1

 RECEIVED DATA= b'Konichiwa'
macbook@macbooks-MacBook-Pro CN Lab Programs %
```

PROGRAM :

STOP AND WAIT CLIENT :

```
import socket

HOST = "  # The server's hostname or IP address
PORT = 65432        # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        while(1):
                print("\nEnter Message to server(q or Q to quit):")
                send_data= input()
                s.sendall(send_data.encode())
                if(send_data=="q" or send_data== "Q"):
                  s.close()
                  exit(0)
                while(1):
                  recv_data=s.recv(1024)
                  if(recv_data==b"q" or recv_data==b"Q"):
                          s.close()
                          exit(0)
                  elif(recv_data==b"timeout"):
                          print("\n ack is not received\n resend data=",send_data)
                          s.sendall(send_data.encode())
                  else:
                          print("\n RECEIVED ACK :",recv_data)
                          break
```

OUTPUT :

STOP AND WAIT CLIENT :

StopAndWaitServer.py | StopAndWaitClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % python3 StopAndWaitClient.py

Enter Message to server(q or Q to quit):
Bonjour

 RECEIVED ACK : b'ack'

Enter Message to server(q or Q to quit):
Auf Widersen

 ack is not received
 resend data= Auf Widersen

 RECEIVED ACK : b'ack'

Enter Message to server(q or Q to quit):
Konichiwa

 RECEIVED ACK : b'ack'

Enter Message to server(q or Q to quit):
q
macbook@macbooks-MacBook-Pro CN Lab Programs %
```

RESULT:

Thus, the Stop and Wait Protocol has been implemented successfully.

Ex No : 3. **IMPLEMENTATION OF SLIDING WINDOW PROTOCOL**
Date : 09/09/2021

AIM:

To implement sliding window technique using python

OBJECTIVE:

To learn how to implement sliding window technique.

SOFTWARE REQUIRED:

Idle python 3.7.4/ Text Editor and python 3.7.4 interpreter software

ALGORITHM:

SLIDING WINDOW SERVER :

STEP 1: Import Socket

STEP 2: Assign Host ID and Port Number so that client and server connect by IP.

STEP 3: Create an socket at ‘s’ with required parameter .

STEP 4 : Give command to combine the host port Number using ‘s.bind (HOST, PORT)’

and give command ‘s.listen()’ to wait for connection.

STEP 5: With Prebuilt function "conn", after getting connection, print by “connected by! the variable “addr”

STEP 6 : In While condition get the data from client up to required limit of bytes if the client input is “q” or “Q” Close the loop then; Else print the message

"1. SEND ACK

2. TIME OUT"

and get the option from the client as", to

Continue or else consider “time out” as “send_data”.

STEP 7 : End.

ALGORITHM :

SLIDING WINDOW CLIENT :

STEP 1 : Import Socket

STEP 2: Assign “HOST ID” and “Port Number” same as that of the server to connect with server.

STEP 3: Create socket as ‘s’ with required parameter to connect with server.

STEP 4 : Give command to connect to serve using HOST and PORT number by “s.connect (HOST, PORT))”.

STEP 5: In while condition, using print function, get the message which is to be sent to server and store it in "send_data", now send the message by encoding using the command “s.sendall (send_data.encode())”. Now, if “send-data == “q” or “Q”, close the loop and exit.

STEP 6: In while condition initialize “recv_data = s.recv(1024)”;

STEP 6.1: In it condition, “recv_data = b‘q’ or recv_data == b"Q"):" Close the Socket and exit loop.

STEP 6.2: elif (recr_data = b”timeout"): then print "ack is not received" and “s.sendall (send_data. encode())”.

STEP 6.3: oh else, print “received acknowledgment” and break the loop

STEP 7: End

PROGRAM :

SLIDING WINDOW SERVER :

```
import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT))
        s.listen()
        conn, addr = s.accept()
        with conn:
                print('Connected by', addr)
                while True:
                        recv_data=conn.recv(1024)
                        if(recv_data==b"q" or recv_data=="Q"):
                                s.close()
                                exit(0)
                        else:
                                print("\n1.SEND ACK\n2.TIME OUT\n")
                                opt=input("Enter Option : ")
                                if(opt == "1"):
                                        send_data = "ack"
                                        print("RECEIVED DATA = ",recv_data )
                                else:
                                  send_data = "timeout"
                                conn.sendall(send_data.encode())
```

OUTPUT :

SLIDING WINDOW SERVER :

SlidingWindowServer.py | SlidingWindowClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % cd SlidingWindow
macbook@macbooks-MacBook-Pro SlidingWindow % ls
SlidingWindowClient.py  SlidingWindowServer.py
macbook@macbooks-MacBook-Pro SlidingWindow % python3 SlidingWindowServer.py
Connected by ('127.0.0.1', 50500)

1.SEND ACK
2.TIME OUT

Enter Option : 1
RECEIVED DATA =  b'hello'

1.SEND ACK
2.TIME OUT

Enter Option : 2

1.SEND ACK
2.TIME OUT

Enter Option : 2

1.SEND ACK
2.TIME OUT

Enter Option : 1
RECEIVED DATA =  b'goodbye server'
macbook@macbooks-MacBook-Pro SlidingWindow %
```

PROGRAM :

SLIDING WINDOW CLIENT :

```
import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 65432        # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        while(1):
                print("Enter Message to server (q or Q to quit) : ")
                send_data= input()
                s.sendall(send_data.encode())
                if(send_data == "q" or send_data == "Q"):
                  s.close()
                  exit(0)
                while(1):
                  recv_data=s.recv(1024)
                  if(recv_data == b"q" or recv_data == b"Q"):
                          s.close()
                          exit(0)
                  elif(recv_data == b"timeout"):
                          print("ack is not received\n resend data = ",send_data)
                          s.sendall(send_data.encode())
                  else:
                          print("RECEIVED ACK : ",recv_data)
                          break
```

OUTPUT :

SLIDING WINDOW CLIENT :

SlidingWindowServer.py | SlidingWindowClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % cd SlidingWindow
macbook@macbooks-MacBook-Pro SlidingWindow % ls
SlidingWindowClient.py  SlidingWindowServer.py
macbook@macbooks-MacBook-Pro SlidingWindow % python3 SlidingWindowClient.py
Enter Message to server (q or Q to quit) :
hello
RECEIVED ACK :  b'ack'
Enter Message to server (q or Q to quit) :
goodbye server
ack is not received
 resend data =  goodbye server
ack is not received
 resend data =  goodbye server
RECEIVED ACK :  b'ack'
Enter Message to server (q or Q to quit) :
q
macbook@macbooks-MacBook-Pro SlidingWindow %
```

RESULT:

Thus, the implementation of Sliding Window Technique was successfully executed.

Ex No : 4. **IMPLEMENTATION AND STUDY OF GO BACK N PROTOCOL**

Date : 09/09/2021

AIM:

To implement Go Back N Protocol using python .

OBJECTIVE:

To learn how to implement Go Back N Protocol

SOFTWARE REQUIRED:

Idle python 3.7.4/ Text Editor and python 3.7.4 interpreter software

ALGORITHM:

GO BACK N SERVER :

STEP 1 : Import socket and assign HOST= €127.0.0.1' and POST = 65432.

STEP 2: with socket.socket(socket. AF_INET, socket.SOCK_STREAM) as s assign s.blind (HOST,

PORT)), s.listen (), conn, addr = s.accept ().

STEP 3: with conn: print (‘connected by’, addr)

STEP 3.1: while True, recv_data = conn.recv(1024)

STEP 3.1.1: if (recv_data == b‘q’) then s.close() and exit(0)

STEP 3.1.2: else print (“\n do not send ack”) and opt = input(“Enter opt = \n”)

STEP 3.1.3 If (opt == “1") then send_data = "Ack" and print(“received data”). assign i= input ("Enter to continue”)

step 3.1.4: else send_back = “NACK” and conn.sendall(send_data.encode())

GO BACK N CLIENT :

STEP 1: Import socket and assign, HOST = '127.0.0.1' and PORT = 65432.

STEP 2: with socket.socket (socket. AF_INET, socket.SOCK_STREAM) as S:

STEP 3: Give s.connect ((HOST, PORT)) and send_data = list ().

STEP 4: while (1) assign pkt = int (input (/n Enter no of packets (o to exit): "))

STEP 4.1: If (pkt ==0); then assign s.sendall("q".encode()), s.close() and exit(0)

STEP 4.2: while (i < pkt) then if (resend == 1) and assign nack = 100 and print("Resending Packet", I+1, ":", send_data[i]), opt = input ("Enter 1 to continue"), else print("Enter the packet :", i+1) and assign s_data = input (), send_data.append (s_data), s. sendall (sand_data[i]. encode ()) and recv_data= s. recv (1024) .

STEP 4.3: if (recv_data == b" NACK") then print received negative acknowledgment.

STEP 4.4: if (nack >i) then nack = i

STEP 4.5: if (i+1) >= pkt then assign, resend =1, i = nack, continue, i=i+l and print ("retransmission complemented").

PROGRAM :

GO BACK N SERVER :

```
import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            recv_data=conn.recv(1024)
            if(recv_data==b"q"):
                s.close()
                exit(0)
            else:
                print("\n1.SEND ACK\n2.DO NOT SEND
ACK\n")
                opt=input("Enter Option:\n")
                if(opt=="1"):
                    send_data="ACK"
                    print("\n RECEIVEDDATA=",recv_data )
                    i=input("Enter 1 to continue:\n")
                else:
                  send_data="NACK";
                conn.sendall(send_data.encode())
```

OUTPUT :

GO BACK N SERVER :

GoBackNServer.py | GoBackNClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % cd GoBackN
macbook@macbooks-MacBook-Pro GoBackN % ls
GoBackNClient.py        GoBackNServer.py
macbook@macbooks-MacBook-Pro GoBackN % python3 GoBackNServer.py
Connected by ('127.0.0.1', 50595)

1.SEND ACK
2.DO NOT SEND ACK

Enter Option:
1

 RECEIVED DATA= b'Hello'
Enter 1 to continue:
1

1.SEND ACK
2.DO NOT SEND ACK

Enter Option:
2

1.SEND ACK
2.DO NOT SEND ACK

Enter Option:
1

 RECEIVED DATA= b'Goodbye!'
Enter 1 to continue:
1
macbook@macbooks-MacBook-Pro GoBackN %
```

PROGRAM :

GO BACK N CLIENT :

```
import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 65432        # The port used by the server
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
   s.connect((HOST, PORT))
   send_data=list()
   while(1):
      pkt=int(input("/nENTER NUMBER OF PACKETS(0 to exit):"))
      if(pkt==0):
         s.sendall("q".encode())
         s.close()
         exit(0)
      nack=100;
      resend=0;
      i=0
      while(i < pkt):
         if (resend==1):
            nack=100
            print("\n RESENDING PACKET ",i+1," :",send_data[i])
            opt=input(" ENTER 1 TO CONTINUE")
         else:
            print("\nENTER THE PACKET:",i+1)
            s_data=input()
            send_data.append(s_data)
         s.sendall(send_data[i].encode())
         recv_data=s.recv(1024)
         if(recv_data==b"NACK"):
            print("\nRECEIVED NEGATIVE ACKNOWLEDGEMENT")
            if(nack>i):
               nack=i
         if((i+1)>=pkt):
            resend=1
            i=nack
            continue
         i=i+1
      print("\nRETRANSMISSION COMPLETED”);
```

OUTPUT :

GO BACK N CLIENT :

GoBackNServer.py | GoBackNClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % cd GoBackN
macbook@macbooks-MacBook-Pro GoBackN % ls
GoBackNClient.py        GoBackNServer.py
macbook@macbooks-MacBook-Pro GoBackN % python3 GoBackNClient.py
/nENTER NUMBER OF PACKETS(0 to exit):2

ENTER THE PACKET: 1
Hello

ENTER THE PACKET: 2
Goodbye!

RECEIVED NEGATIVE ACKNOWLEDGEMENT

 RESENDING PACKET  2   : Goodbye!
 ENTER 1 TO CONTINUE1

RETRANSMISSION COMPLETED
/nENTER NUMBER OF PACKETS(0 to exit):0
macbook@macbooks-MacBook-Pro GoBackN %
```

RESULT:

Thus, the implementation of Go Back N Protocol was successfully executed.

Ex No : 5 **IMPLEMENTATION OF SELECTIVE REPEAT PROTOCOL**

Date : 18/09/2021

AIM:

To implement Selective Repeat Protocol server and client using python.

OBJECTIVE:

To learn how to Selective Repeat Protocol server and client using python.

SOFTWARE REQUIRED:

Idle python 3.7.4/ Text Editor and python 3.7.4 interpreter software

ALGORITHM:

SELECTIVE REPEAT SERVER :

STEP 1: Import socket and assign HOST= 127.0.0.1 and PORT = 65432.

STEP 2: with socket socket (socket. AF_INEI, socket sock_STREAM) as, bind HOST and PORT and give listen().

STEP 3: Assign conn, adda as s.accept ().

STEP 4: with conn, print "connected by" with address

STEP 4.1: while True, assign recv_data as conn.recv(1024).

STEP 4.2: If recv_data == b"q", s.close() and exit.

STEP 4.3: Else print 1. Send ack 2. Do not send ack

STEP 4.4: Get option as opt

SELECTIVE REPEAT CLIENT :

STEP 1: Import Socket and assign HOST and PORT as '127.0.0.1' and 65432 respectively.

STEP 2: Assign send_data = list() and nack= list().

STEP 3: while True, get the no of pocket as pkt;

STEP 3.1: if pkt is equal to 0 then s.send((‘q’.encode())).

PROGRAM :

SELECTIVE REPEAT SERVER :

```
import socket

HOST = '127.0.0.1' # Standard loopback interface address (localhost)
PORT = 65432 # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT))
        s.listen()
        conn, addr = s.accept()
        with conn:
                print('Connected by', addr)
                while True:
                        recv_data=conn.recv(1024)
                        if(recv_data==b"q"):
                                s.close()
                                exit(0)
                        else:
                                print("\n1.SEND ACK\n2.DO NOT SEND ACK\n")
                                opt=input("Enter Option:\n")
                                if(opt=="1"):
                                        send_data="ACK"
                                        print("\n RECEIVED DATA=",recv_data )
                                        i=input("Enter 1 to continue:\n")
                                else:
                                        send_data="NACK"
                                        conn.sendall(send_data.encode())
```

OUTPUT :

SELECTIVE REPEAT SERVER :

SelectiveRepeatServer.py	SelectiveRepeatClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % cd SelectiveRepeat
macbook@macbooks-MacBook-Pro SelectiveRepeat % ls
SelectiveRepeatClient.py        SelectiveRepeatServer.py
macbook@macbooks-MacBook-Pro SelectiveRepeat % python3 SelectiveRepeatServer.py
Connected by ('127.0.0.1', 51033)

1.SEND ACK
2.DO NOT SEND ACK

Enter Option:
1

 RECEIVED DATA= b'Hello and Goodbye!'
Enter 1 to continue:
1
```

PROGRAM :

SELECTIVE REPEAT CLIENT :

```
import socket


HOST = "127.0.0.1"
PORT = 65432
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        send_data = list()
        nack = list()
        while 1:
                pkt = int(input("\nENTER NUMBER OF PACKETS (0 to exit) : "))
                if pkt == 0:


                        s.sendall("q".encode())
                        s.close()
                        exit(0)
                for i in range(0, pkt):
                        nack.append(0)
                        resend = 0
                i = 0
                while i < pkt:
                        if resend == 1:
                                if nack[i] == 1:
                                        nack[i] = 0
                                        print("\nRESENDING PACKET ", i + 1, " : ", send_data[i])
                                        opt = input("ENTER 1 TO CONTINUE")
                                        s.sendall(send_data[i].encode())
                                        recv_data = s.recv(1024)
```

```
                        if recv_data == b"NACK":
                                print("\nRECEIVED NEGATIVE
ACKNOWLEDGEMENT")
                                nack[i] = 1
                        else:
                                print("\nReceived Acknowledgement")
                                opt = int(input("\nEnter 1 to continue"))
                        if (i + 1) >= pkt:
                                break
                else:
                        print("\nENTER THE PACKET : ", i + 1)
                        s_data = input()
                        send_data.append(s_data)
                        s.sendall(send_data[i].encode())
                        recv_data = s.recv(1024)
                        if recv_data == b"NACK":
                                print("\nRECEIVED NEGATIVE ACKNOWLEDGEMENT")
                                nack[i] = 1
                        else:
                                print("Received Acknowledgement")
                        if (i + 1) >= pkt:
                                resend = 1
                                i = 1
                                continue
                i = i + 1
        print("\nRETRANSMISSION COMPLETED")
```

OUTPUT :

SELECTIVE REPEAT CLIENT :

SelectiveRepeatServer.py | SelectiveRepeatClient.py

```
macbook@macbooks-MacBook-Pro CN Lab Programs % cd SelectiveRepeat
macbook@macbooks-MacBook-Pro SelectiveRepeat % ls
SelectiveRepeatClient.py        SelectiveRepeatServer.py
macbook@macbooks-MacBook-Pro SelectiveRepeat % python3 SelectiveRepeatClient.py

ENTER NUMBER OF PACKETS (0 to exit) : 1

ENTER THE PACKET :  1
Hello and Goodbye!
```

RESULT:

Thus, the implementation of Selective Repeat Protocol was successfully executed.

EXP NO : 6 **STUDY OF HIGH LEVEL DATA LINK PROTOCOL**
DATE : 09/10/2021

Aim:

To study about (HDLC) High Level Data Link Protocol

Introduction:

HDLC - Short for High-level Data Link Control, a transmission protocol used at the data link layer (layer 2) of the OSI seven layer model for data communications. The HDLC protocol embeds information in a data frame that allows devices to control data flow and correct errors. HDLC is an ISO standard developed from the Synchronous Data Link Control (SDLC) standard proposed by IBM in the 1970's. HDLC NRM (also known as SDLC) .HDLC is a bit oriented protocol that supports both half-duplex and full-duplex communication over point to point & multipoint link.

For any HDLC communications session, one station is designated primary and the other secondary. Asession can use one of the following connection modes, which determine how the primary and secondary stations interact

Normal unbalanced: The secondary station responds only to the primary station.

Asynchronous: The secondary station can initiate a message.

Asynchronous balanced: Both stations send and receive over its part of a duplexline.This mode is used for **X.25 packet-switching networks**.
The **Link Access Procedure-Balanced (LAP-B) and Link Access Procedure D-channel (LAP-D)protocols are subsets of HDLC**.

LAPB is a bit-oriented synchronous protocol that provides complete data transparency in a full-duplex point-to-point operation. It supports a peer-to-peer link in that neither end of the link plays the role ofthe permanent master station. HDLC NRM, on the other hand, has a permanent primary station with one or more secondary stations.

The concept of a frame window is used to send multiple frames before receiving confirmation thatthe first frame has been correctly been received. This means that data can continue to flow in situations where there may be long "turn-around" time lagswithout stopping to wait for an acknowledgement. This kind of situation occurs, for instance in satellite communication.

HDLC defines three types of frames:

Information frames (I-frame)
Supervisory frame (S-frame)
Unnumbered frame (U-frame)

Information frames

I-frames carry user's data and control information about user's data.I-frame carries user data in the information field.The first bit ofcontrol field is always zero, i.e. the presence ofzero at this place indicates that it is I-frame.

Bit number 2, 3 & 4 in control field is called N(S) that specifies the sequence number of the frame.Thus it specifies the number of the frame that is currently being sent. Since it is a 3.bit field, only eight sequence numbers are possible 0, 1,2,3,4,5,6, 7 (000 to 111).
Bit number 5 in control field is P/F i.e. Poll/Final and is used for these two purposes. It has, meaningonly when it is set i.e. when P/F=1.

It can represent the following two cases.
It means poll when frame is sent by a primary station to secondary (when address field contains theaddress of receiver).
It means final when frame is sent by secondary to a primary (when the address field contains the address of the sender).
Bit number 6, 7, and 8 in control field specifies N(R) i.e. the sequence number of the frame expectedin return in two-way communication.

Supervisory frame

S-frame carries control information, primarily data link layer flow and error controls. The first two bits in the control field of S-frame are always 10.Receive Ready-used to acknowledge frames when no I-framesare availab1e to piggyback the acknowledgement.

REJ Reject-used by the receiver to send a NAK when error has occurred.RNR Receive Not Ready-used for flow control.SREJ Selective Reject-indicates to the transmitter that it should retransmit the frameindicated in the N(R) subfield.There is no N(S) field in control field of S-frame as S-frames do not transmit data.

Last three bits in control field indicates N(R) i.e. they correspond to the ACK or NAK value.

Unnumbered frame

U-frames are reserved for system management and information carried by them is used for managingthe link. U-frames are used to exchange session management and control information between the twoconnected devices. Information field in U-frame does not carry user information rather, it carries system managementinformation.

The frame format of U-frame is shown in diagram.U-frame is identified by the presence of 11 in the first and second bit position in control field.

These frames do not contain N(S) or N(R) in control field.

U. frame contains two code fields, one two hit and other three bit.

V. These five bits can create upto 32 different U-frames.

Protocol Structure - HDLC: High Level Data Link Control

Flag - The value of the flag is always (0x7E).

Address field - Defines the address of the secondary station which is sending the frame or the destination of the frame sent by the primary station. It contains Service Access Point (6bits), a Command/Response bit to indicate whether the frame relates to information frames (I-frames) being sent from the node or received by the node, and an address extension bit which is usually set to true to indicate that the address is of length one byte. When set to false it indicates an additional byte follows.

Extended address - HDLC provides another type of extension to the basic format. The address field may be extended to more than one byte by agreement between the involved parties.

Control field - Serves to identify the type of the frame. In addition, it includes sequence numbers, control features and error tracking according to the frame type.

FCS - The Frame Check Sequence (FCS) enables a high level of physical error control by allowing the integrity of the transmitted frame data to be checked.

Related Protocols: LAPB, ISDN, X.25, Frame Relay, SDLC

RESULT:

Thus the High level data link control protocol was studied.

Ex No:7 **IMPLEMENTATION OF PING COMMAND**

Date: 27/08/2021

AIM:

To implement Ping Command using command prompt.

OBJECTIVE:

. To learn how to implement Ping Command.

SOFTWARE REQUIRED:

Command prompt/ Unix based Terminal

PROCEDURE:

To do this experiment – follow these steps:

In this experiment – students have to understand basic networking commands e.g ping, tracert etc.All commands related to network configuration which includes how to switch to privilege mode and normal mode and how to configure router interface and how to save this configuration to flash memoryor permanent memory.

PING COMMAND

DESCRIPTION:

Ping sends an ICMP ECHO_REQUEST packet to the specified host. If the host responds, you get an ICMP packet back. Sound strange? Well, you can “ping” an IP address to see if amachine is alive. If there is no response, you know something is wrong.

INPUT: ping <IP address>

ALGORITHM :

STEP 1: Inet address is used for reading the IP address

STEP 2: IP address are given as input to the function

STEP 3: The IP address is checked for reachability

STEP 4: Prints “Host is Reachable” if IP address is reachable else “Host is not reachable” is printed

OUTPUT :

```
Last login: Wed Oct 13 19:04:21 on ttys000
macbook@macbooks-MacBook-Pro ~ % ping www.google.com
PING www.google.com (142.250.193.100): 56 data bytes
64 bytes from 142.250.193.100: icmp_seq=0 ttl=117 time=20.919 ms
64 bytes from 142.250.193.100: icmp_seq=1 ttl=117 time=20.797 ms
64 bytes from 142.250.193.100: icmp_seq=2 ttl=117 time=20.846 ms
64 bytes from 142.250.193.100: icmp_seq=3 ttl=117 time=27.907 ms
^C
--- www.google.com ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.797/22.617/27.907/3.054 ms
macbook@macbooks-MacBook-Pro ~ %
```

```
macbook@macbooks-MacBook-Pro ~ % ping 142.250.195.36
PING 142.250.195.36 (142.250.195.36): 56 data bytes
64 bytes from 142.250.195.36: icmp_seq=0 ttl=117 time=22.241 ms
64 bytes from 142.250.195.36: icmp_seq=1 ttl=117 time=22.929 ms
64 bytes from 142.250.195.36: icmp_seq=2 ttl=117 time=21.842 ms
64 bytes from 142.250.195.36: icmp_seq=3 ttl=117 time=14.715 ms
^C
--- 142.250.195.36 ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 14.715/20.432/22.929/3.323 ms
macbook@macbooks-MacBook-Pro ~ %
```

```
macbook@macbooks-MacBook-Pro ~ % ping 192.168.120.2
PING 192.168.120.2 (192.168.120.2): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
Request timeout for icmp_seq 2
^C
--- 192.168.120.2 ping statistics ---
4 packets transmitted, 0 packets received, 100.0% packet loss
macbook@macbooks-MacBook-Pro ~ %
```

RESULT:

Thus, the implementation of Ping Command was successfully implemented.

Ex No : 8. **IMPLEMENTATION OF TRACE ROUTING COMMAND**
Date : 27/08/2021

AIM:

To implement Trace Routing Command using command prompt.

OBJECTIVE:

To learn how to implement Trace Routing Command.

SOFTWARE REQUIRED:

Command prompt/ Unix based Terminal

PROCEDURE:

To do this experiment – follow these steps:In this experiment – students have to understand basic networking commands e.g ping, tracert etc.All commands related to network configuration whichincludes how to switch to privilege mode and normal mode and how to configure router interface and how to save this configuration to flash memory or permanent memory.

TRACEROUTE

DESCRIPTION:

Tracert is a command which can show you the path a packet of information takes from yourcomputer to one you specify. It will list all the routers it passes through until it reaches its destination, or fails to and is discarded. In addition to this, it will tell you how long each 'hop' from router to router takes.

INPUT: tracert<IP address>

ALGORITHM:

STEP 1: A buffered reader is initialized.

STEP 2: Tracert is used for showing the path a packet and no. of hops

STEP 3: A website is given as input for which the no. of hops and other details of the website

STEP 4: If the site is unreachable then “Could not connect” is printed

OUTPUT :

```
Last login: Fri Oct  8 09:36:21 on ttys000
macbook@macbooks-MacBook-Pro ~ % traceroute www.google.com
traceroute to www.google.com (142.250.67.228), 64 hops max, 52 byte packets
 1  192.168.1.1 (192.168.1.1)  14.305 ms  5.708 ms  4.269 ms
 2  192.168.29.1 (192.168.29.1)  6.347 ms  5.985 ms  6.729 ms
 3  10.39.112.1 (10.39.112.1)  8.471 ms  8.311 ms  8.422 ms
 4  172.31.0.144 (172.31.0.144)  9.580 ms  9.139 ms  8.671 ms
 5  192.168.68.142 (192.168.68.142)  8.778 ms
    192.168.68.146 (192.168.68.146)  10.206 ms
    192.168.68.142 (192.168.68.142)  10.028 ms
 6  172.26.77.165 (172.26.77.165)  7.541 ms  8.141 ms  8.475 ms
 7  172.26.77.130 (172.26.77.130)  8.871 ms  8.661 ms  10.281 ms
 8  192.168.68.136 (192.168.68.136)  11.048 ms  8.086 ms
    192.168.68.134 (192.168.68.134)  6.454 ms
 9  192.168.68.131 (192.168.68.131)  10.043 ms  10.105 ms
    192.168.68.137 (192.168.68.137)  8.631 ms
10  172.31.2.63 (172.31.2.63)  11.328 ms  8.464 ms
    172.31.2.65 (172.31.2.65)  10.667 ms
11  72.14.217.254 (72.14.217.254)  12.695 ms
    74.125.50.202 (74.125.50.202)  13.412 ms
    72.14.196.126 (72.14.196.126)  10.033 ms
12  74.125.242.130 (74.125.242.130)  13.283 ms
    74.125.242.147 (74.125.242.147)  10.655 ms
    108.170.253.119 (108.170.253.119)  12.538 ms
13  216.239.43.234 (216.239.43.234)  10.095 ms
    172.253.72.136 (172.253.72.136)  10.695 ms
    74.125.242.129 (74.125.242.129)  10.990 ms
14  108.170.253.122 (108.170.253.122)  13.941 ms
    108.170.248.209 (108.170.248.209)  42.391 ms
    74.125.242.146 (74.125.242.146)  11.176 ms
15  72.14.232.34 (72.14.232.34)  30.799 ms
    72.14.232.50 (72.14.232.50)  29.353 ms
    108.170.237.68 (108.170.237.68)  29.124 ms
16  bom07s24-in-f4.1e100.net (142.250.67.228)  31.555 ms
    142.250.228.47 (142.250.228.47)  30.551 ms
    bom07s24-in-f4.1e100.net (142.250.67.228)  30.617 ms
macbook@macbooks-MacBook-Pro ~ %
```

RESULT:

Thus, the trace routing of a website was successfully executed.

Ex No : 9. IMPLEMENTATION OF NSLOOKUP COMMAND

Date : 27/08/2021

AIM:

To demonstrate the usage of nslookup command.

OBJECTIVE:

To learn how to implement the nslookup command.

SOFTWARE REQUIRED:

Command prompt/ Unix based Terminal

PROCEDURE:

To do this experiment – follow these steps:

In this experiment – students have to understand basic networking commands e.g ping, tracert etc.All commands related to network configuration which includes how to switch to privilege mode and normal mode and how to configure router interface and how to save this configuration to flash memoryor permanent memory.

NSLOOKUP

DESCRIPTION:

Displays information that you can use to diagnose domain name system (DNS) infrastructure.Before using this tool, you should be familiar with how DNS works. The nslookup command line tool is available only if you have installed the TCP/IP protocol.

INPUT: nslookup <domain name>

ALGORITHM:

STEP 1: Import the net, input output packages.

STEP 2: Use the method exec with the nslookup command and store the result of the command in astream.

STEP 3: Process the stream and print the output.

STEP 4: End the process.

OUTPUT :

```
Last login: Wed Oct 13 19:47:43 on ttys001
macbook@macbooks-MacBook-Pro ~ % nslookup www.python.org
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
www.python.org  canonical name = dualstack.python.map.fastly.net.
Name:   dualstack.python.map.fastly.net
Address: 199.232.20.223

macbook@macbooks-MacBook-Pro ~ % nslookup www.linux.org
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   www.linux.org
Address: 172.67.154.61
Name:   www.linux.org
Address: 104.21.4.234

macbook@macbooks-MacBook-Pro ~ %
```

RESULT:

Thus, the implementation of nslookup command was successfully executed.

Ex No : 10 **IMPLEMENTATION OF IP ADDRESS CONFIGURATION**

Date : 27/08/2021

AIM:

To demonstrate the usage of IP address configuration.

OBJECTIVE:

To learn how to implement the IP address configuration.

SOFTWARE REQUIRED:

Cisco Packet Tracer

PROCEDURE:

STEP 1: Open Cisco Packet Tracer.

STEP 2: Place PC and switch devices.

STEP 3: Form a Connection between two devices.

STEP 4: click on PC to configure ip.

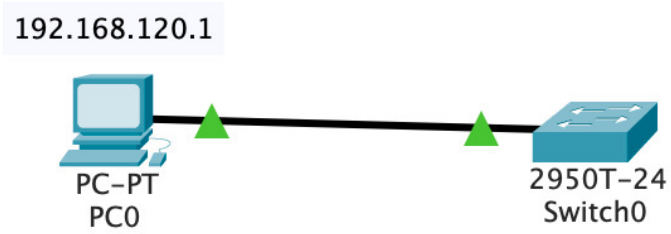
STEP 5: Assign an ip address to pc.

STEP 6: Place a note near PC for reference.

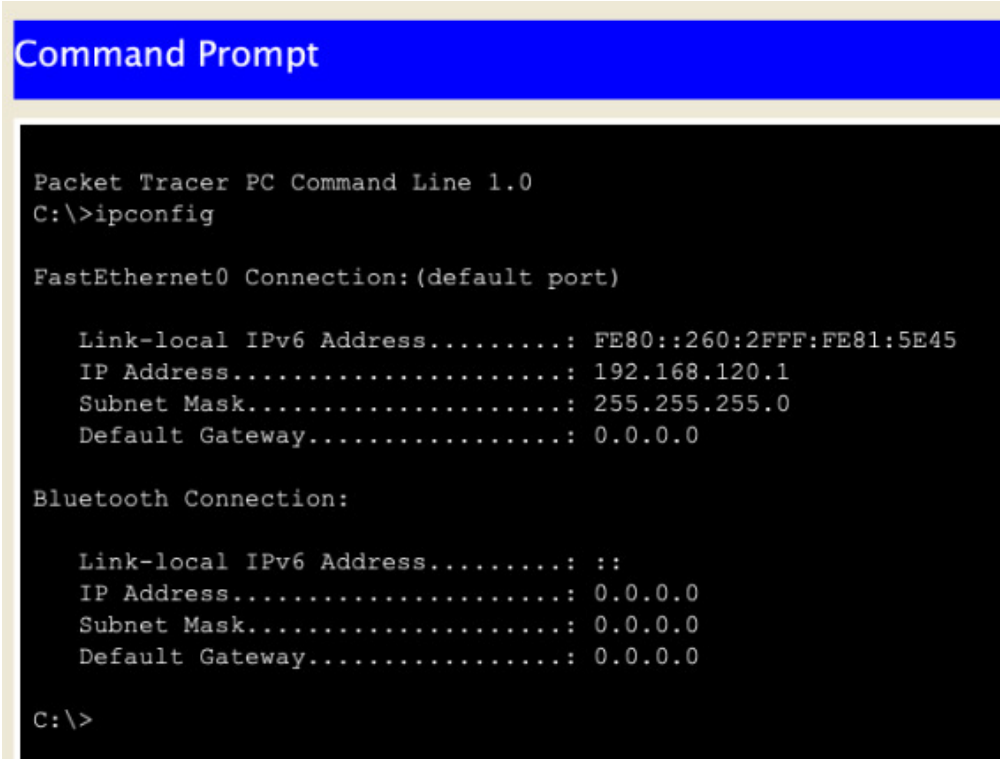
STEP 7: Go to command Prompt and type ipconfig to know it's given ip address.

STEP 8: End

IP ADRESS :



OUTPUT :



RESULT:

Thus, the implementation of IP address configuration was successfully executed.

Ex No : 11 **IMPLEMENTATION OF AND STUDY OF CARRIER SENSE MULTIPLE**

Date: 30/09/2021 **COLLISION DETECTION ACCESS/COLLISION DETECTION,**

COLLISION DETECTION

AIM:

To implement the CSMA/ CD,CA in VIR-TISM.

OBJECTIVE:

To learn how to implement CSMA/ CD,CA using virtism.

SOFTWARE REQUIRED:

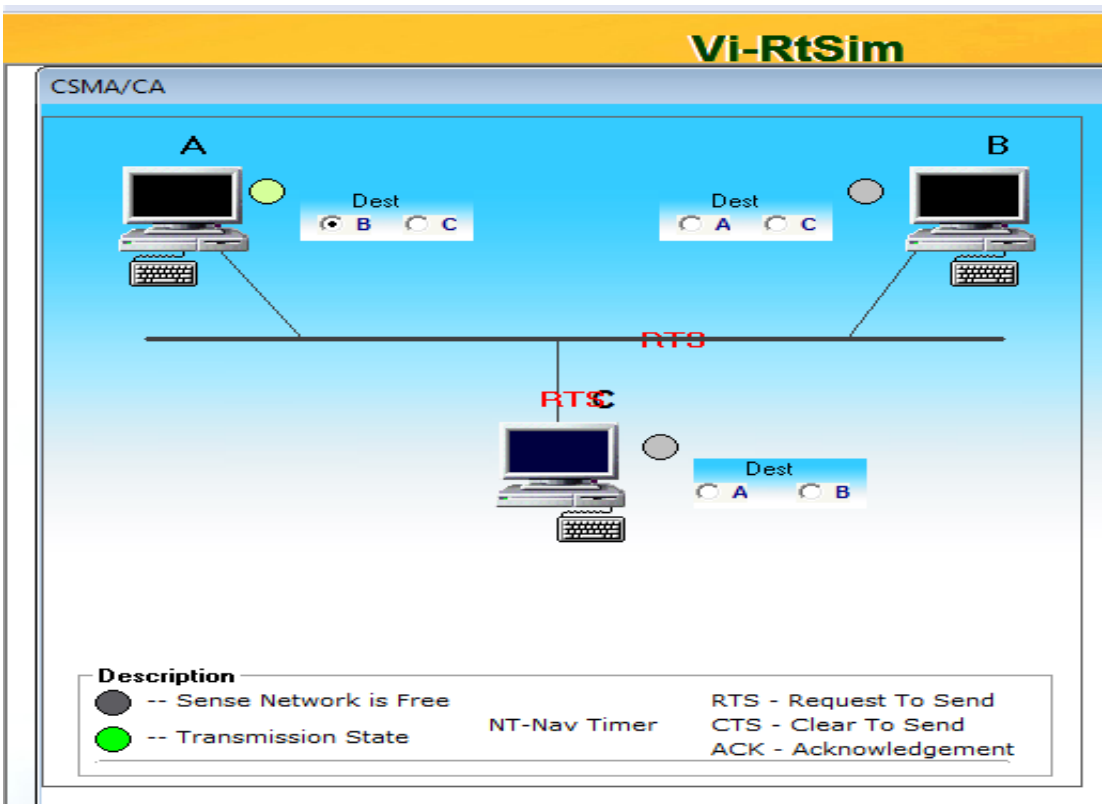
VIR-TISM

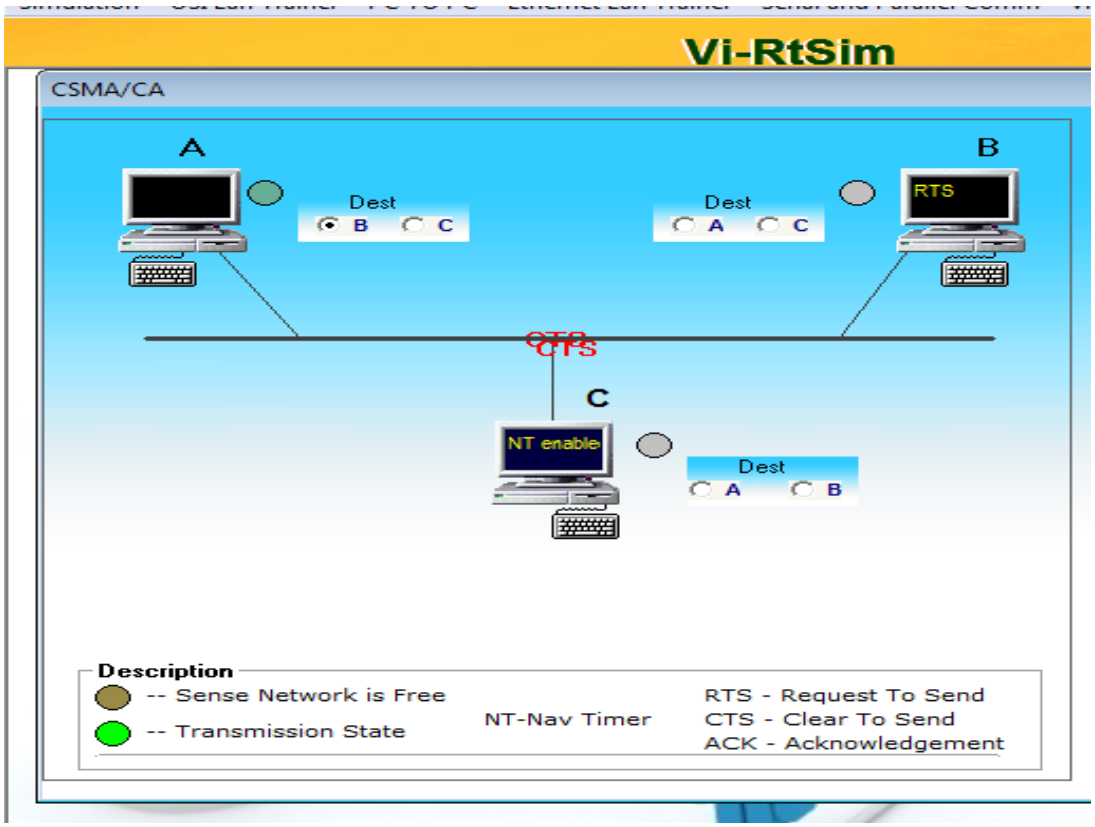
PROGRAM:

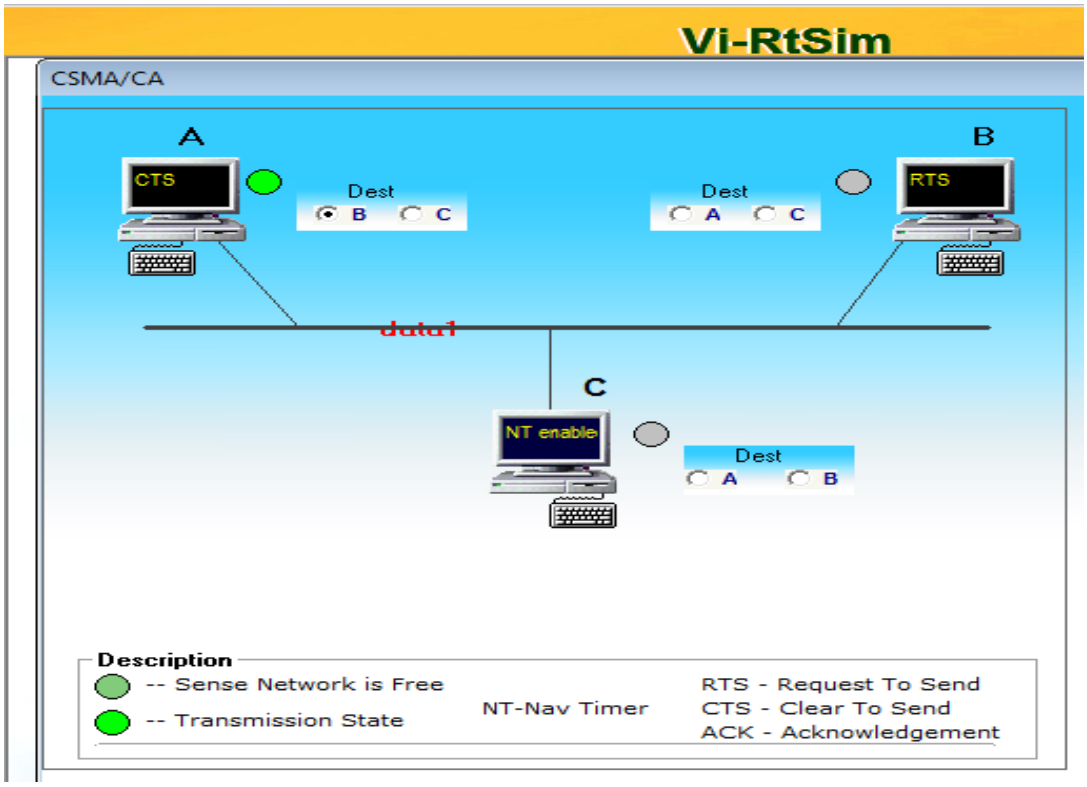
CSMA/CA:

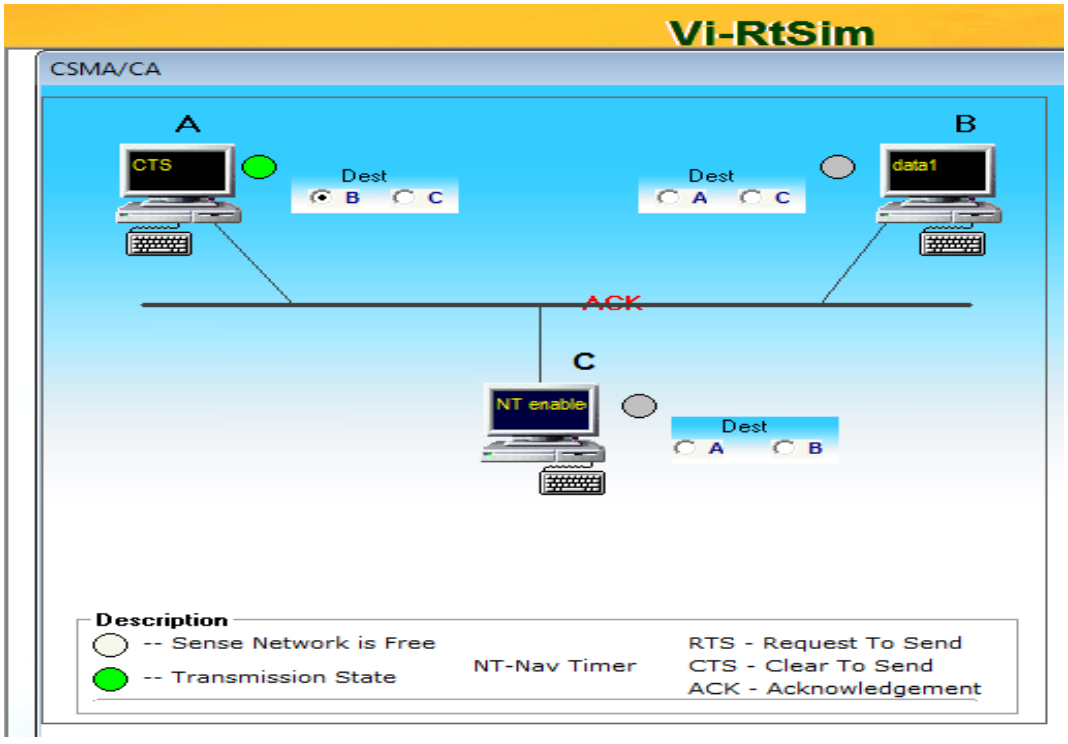
```
include <protocol.h>
void main()
{
        Frame X;
      X="data1";

      CSMACA_INIT();

      CSMACA_START();

        NODE_LISTEN();

        REQUESTTO_SEND(A,B);

      CLEARTO_SEND(B,A);

        DATATO_SEND(A,B,X);

        ACKNOWLEDGE(B,A);
}
```

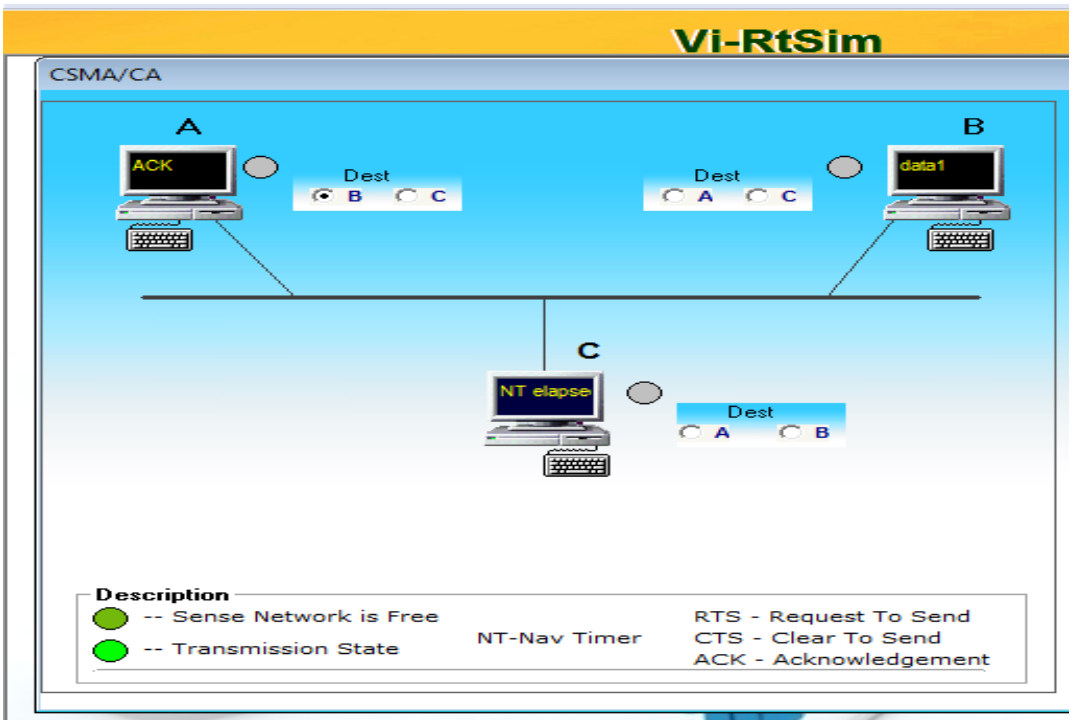
OUTPUT:


Vi-RtSim
CSMA/CA
A
B
C
Dest
B C
Dest
A C
Dest
A B
RTS
RTS
Description
-- Sense Network is Free
-- Transmission State
NT-Nav Timer
RTS - Request To Send
CTS - Clear To Send
ACK - Acknowledgement


Vi-RtSim
CSMA/CA
A
B
C
Dest
B C
Dest
A C
RTS
CTS
NT enable
Dest
A B
Description
-- Sense Network is Free
-- Transmission State
NT-Nav Timer
RTS - Request To Send
CTS - Clear To Send
ACK - Acknowledgement


Vi-RtSim
CSMA/CA
A
B
CTS
RTS
Dest
B C
Dest
A C
data1
C
NT enable
Dest
A B
Description
-- Sense Network is Free
-- Transmission State
NT-Nav Timer
RTS - Request To Send
CTS - Clear To Send
ACK - Acknowledgement


Vi-RtSim
CSMA/CA
A
B
CTS
data1
Dest
B C
Dest
A C
ACK
C
NT enable
Dest
A B
Description
-- Sense Network is Free
-- Transmission State
NT-Nav Timer
RTS - Request To Send
CTS - Clear To Send
ACK - Acknowledgement



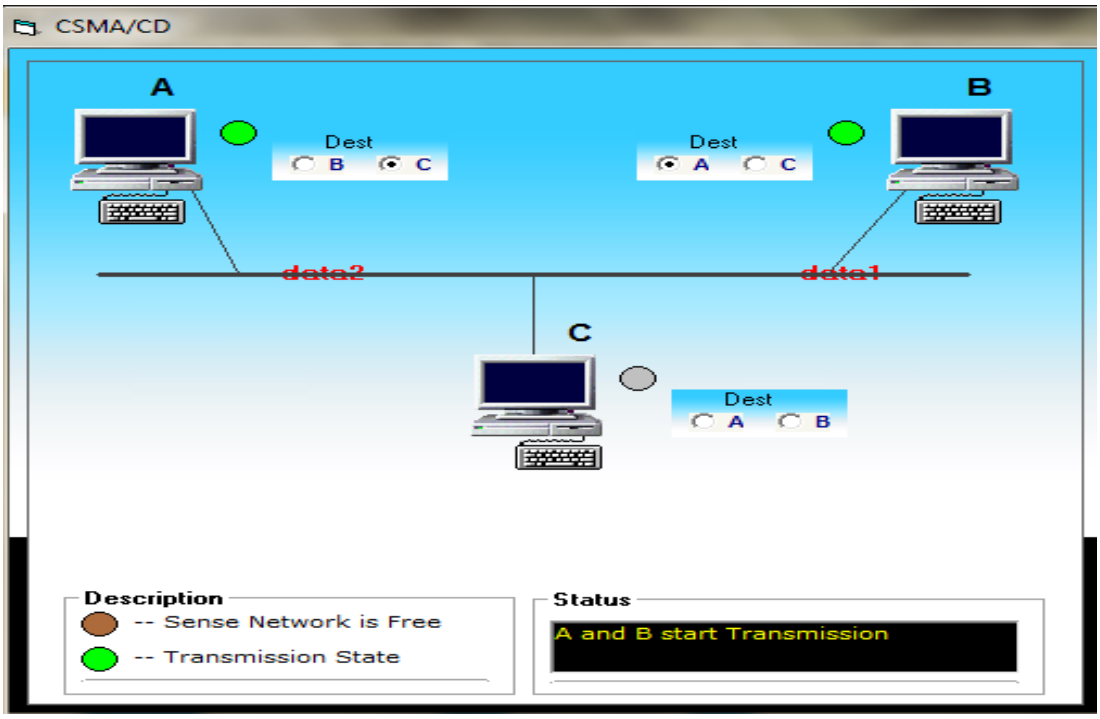
PROGRAM :

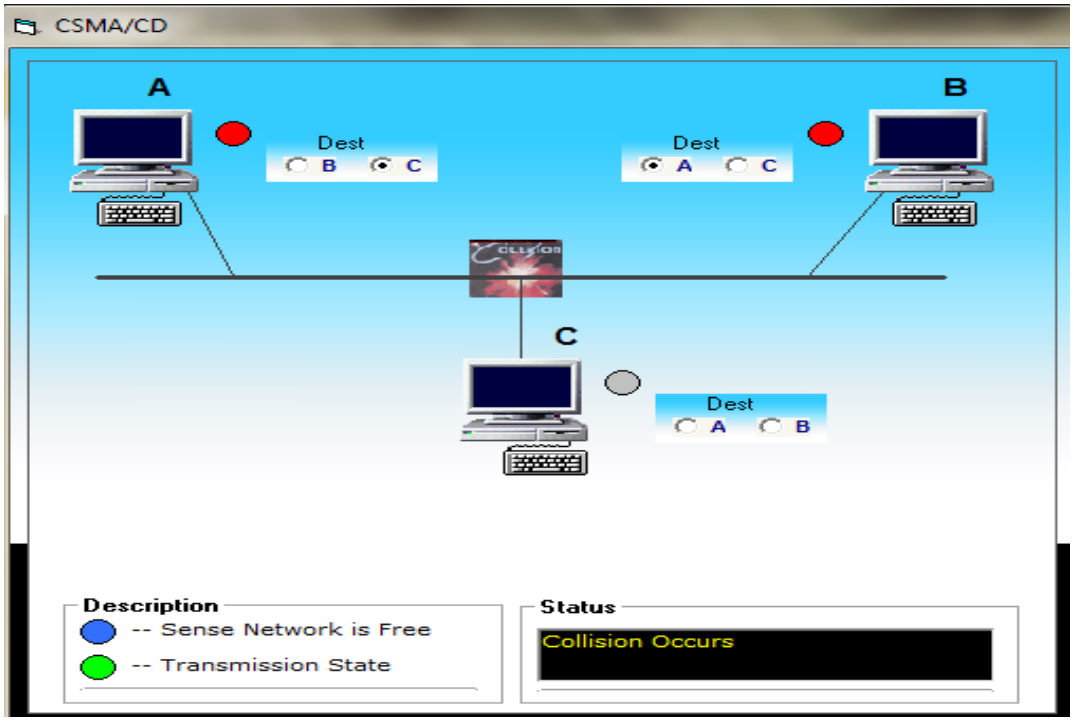
CSMA/CD :

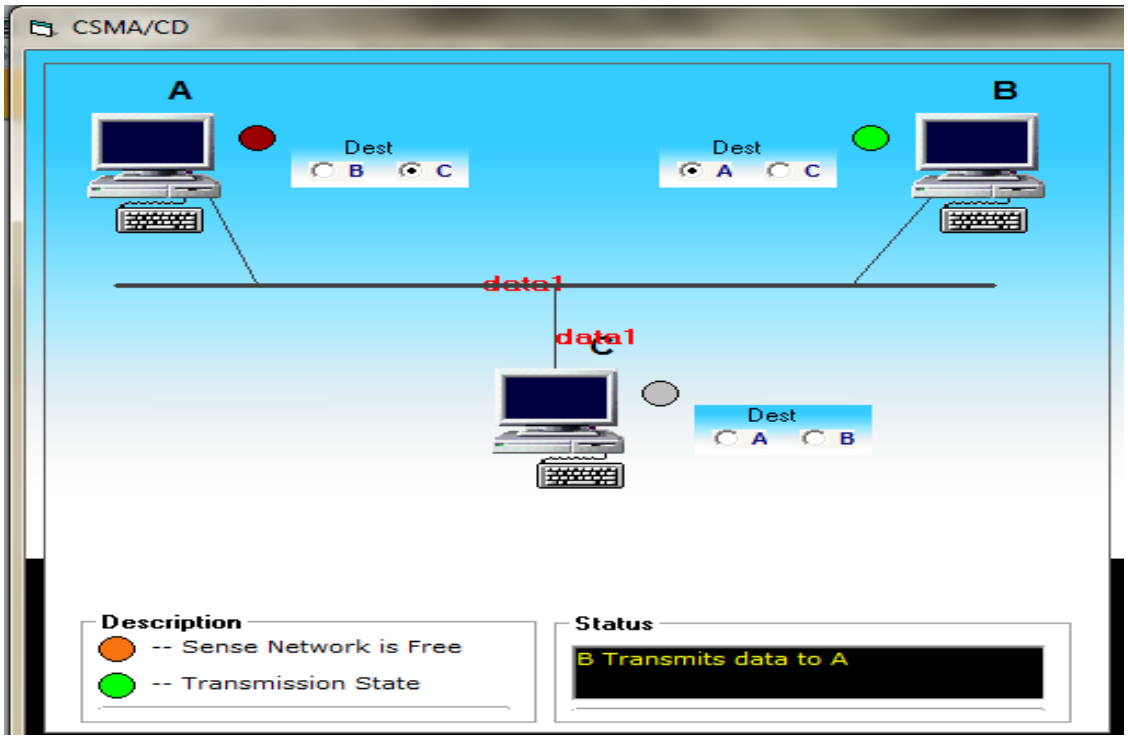
```
include <protocol.h>

void main()

{

        Frame X,Y;

                X=“data1";

                Y=“data2";

                CSMACD_INIT();

                CSMACD_START();

                        CSMA_SEND(B,A,X);

                        CSMA_SEND(A,C,Y);

                R=COLLISION_OCCUR();
```

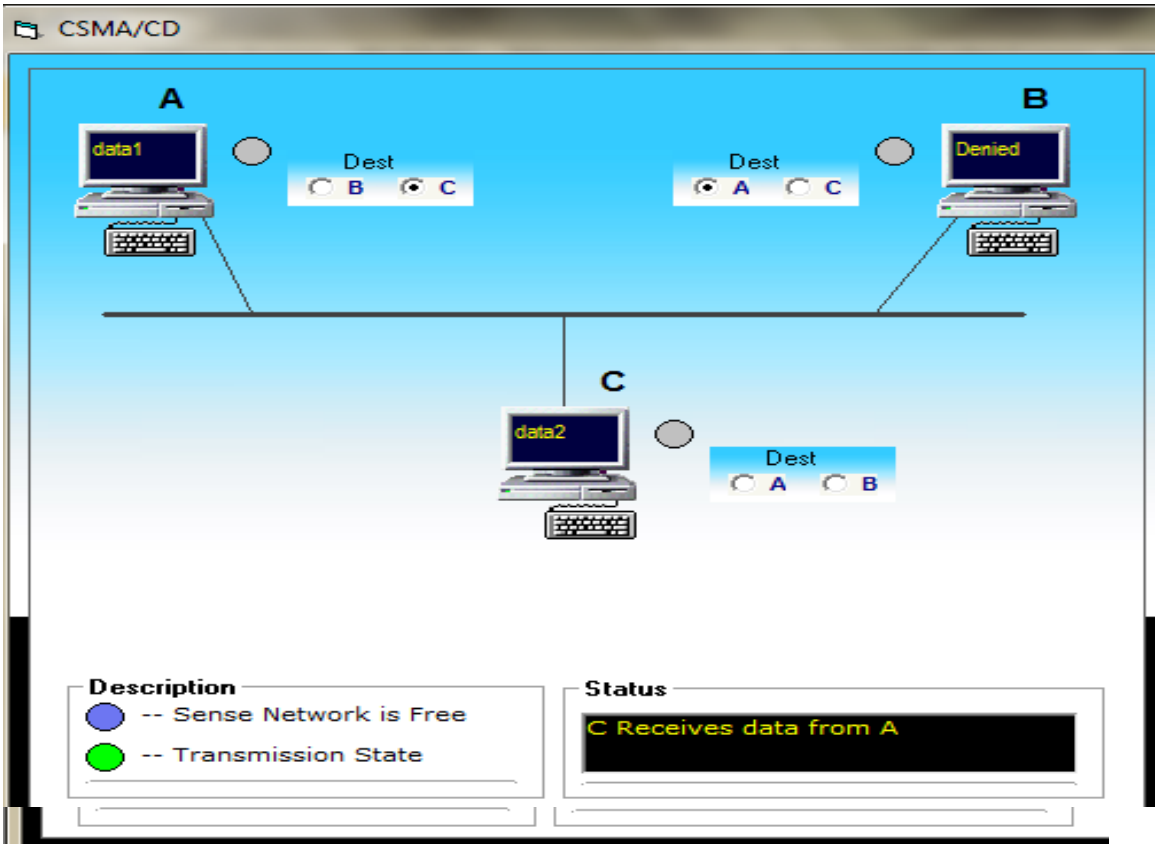
```
        if(R)
        {
            WAIT(1000);
            RETRANSMIT(B,A);
            RETRANSMIT(A,C);
        }
}
```

OUTPUT:




CSMA/CD
A
B
C
Dest
B C
Dest
A C
Dest
A B
Description
-- Sense Network is Free
-- Transmission State
Status
Collision Occurs


CSMA/CD
A
B
C
Dest
B C
Dest
A C
data1
data1
Dest
A B
Description
-- Sense Network is Free
-- Transmission State
Status
B Transmits data to A



RESULT:

Thus, the CSMA/CD,CA is executed and studied.

Ex No : 12 **STUDY OF NETWORK SIMULATOR**

Date : 28/09/2021

AIM:

To write a program and demonstrate the usage of IP address configuration.

OBJECTIVE:

To learn how to implement the IP address configuration.

SOFTWARE REQUIRED:

Network Simulator (NS2) Version 2

INTRODUCTION:

NS is a discrete event simulator targeted at networking research. NS provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite)networks. NS began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT

project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. NS has always included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and SunMicrosystems.

The network simulator ns-2 is a widely accepted discrete event network simulator, actively used for wired and wireless network simulations. It has a highly detailed model of the lower layers (Physicaland MAC) of wireless IEEE 802.11 networks.

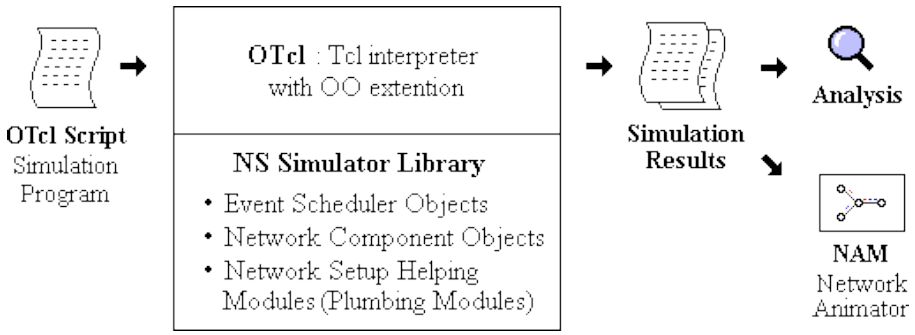
Ns-2 has also an emulation feature, i.e. the ability to introduce the simulator into a live network andto simulate a desired network between real applications in real-time.

Within the scope of this project we developed some methods and extensions to the ns-2 to combinewireless network simulation and network emulation.

OVERVIEW:

NS is an event driven network simulator developed at UC Berkeley that simulates variety of IP networks. It implements network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations. The NS project is now a part of the VINT project that develops

tools for simulation results display, analysis and converters that convert network topologies generated by well-known generators to NS formats. Currently, NS (version 2) written in C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT) is



available. This document talks briefly about the basic structure of NS, and explains in detail how touse NS mostly by giving examples. Most of the figures that are used in describing the NS basic structure and network components are from the 5th VINT/NS Simulator Tutorial/Workshop slides and the NS Manual (formerly called "NS Notes and Documentation"), modified a little bit as needed.

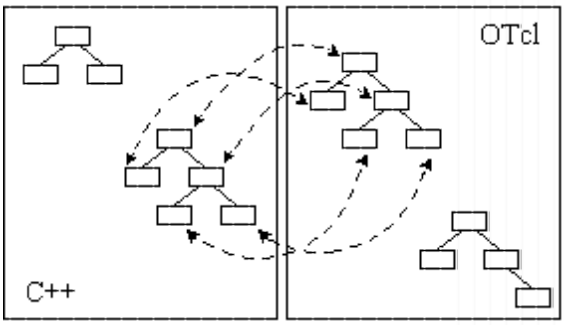
Simplified User's View of NS

NS is Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup (plumbing) module libraries (actually, plumbing modules are implemented as member functions of the base simulator object). In other words, to use NS, you program in OTcl script language. To setup and run a simulation network, a user should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler. The term "plumbing" is used fora network setup, because setting up a network is plumbing possible data paths among network objects by setting the "neighbor" pointer of an object to the address of an appropriate object. Whena user wants to make a new network object, he or she can easily make an object either by writing anew object or by making a compound object from the object library, and plumb the data path through the object. This may sound like complicated job, but the plumbing OTcl modules actuallymake the job very easy. The power of NS comes from this plumbing.

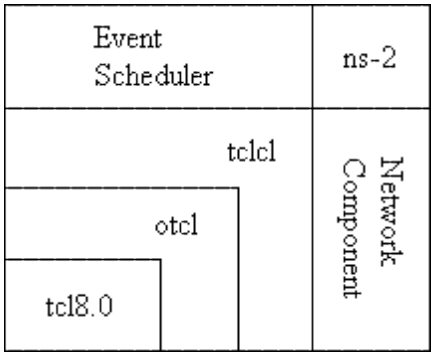
Another major component of NS beside network objects is the event scheduler. An event in NS is apacket ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, which usually are the ones who issued the events, and let them do the appropriate action associated with packet pointed by the event. Network components communicate with one another passing packets,however this does not consume actual simulation time.

All the network components that need to spend some simulation time handling a packet (i.e. need adelay) use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet. For example, a network switch component that simulates a switch with 20 microseconds of switching delay issues an event for a packet to be switched to the scheduler as an event 20 microsecond later. The scheduler after 20 microseconds dequeues the event and fires it to the switch component, which then passes the packet to an appropriate output link component. Another use of an event scheduler is timer.

NS is written not only in OTcl but in C++ also. For efficiency reason, NS separates the data path implementation from control path implementations. In order to reduce packet and event processing time (not simulation time), the event scheduler and the basic network component objects in the datapath are written and compiled using C++. These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objectsand makes the control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object. In this way, the controlsof the C++ objects are given to OTcl. It is also possible to add member functions and variables to aC++ linked OTcl object. The objects in C++ that do not need to be controlled in a simulation or internally used by another object do not need to be linked to OTcl. Likewise, an object (not in the data path) can be entirely implemented in OTcl. Figure 2 shows an object hierarchy example in C++ and OTcl. One thing to note in the figure is that for C++ objects that have an OTcl linkage forming a hierarchy, there is a matching OTcl object hierarchy very similar to that of C++.



C++ and OTcl: The Duality



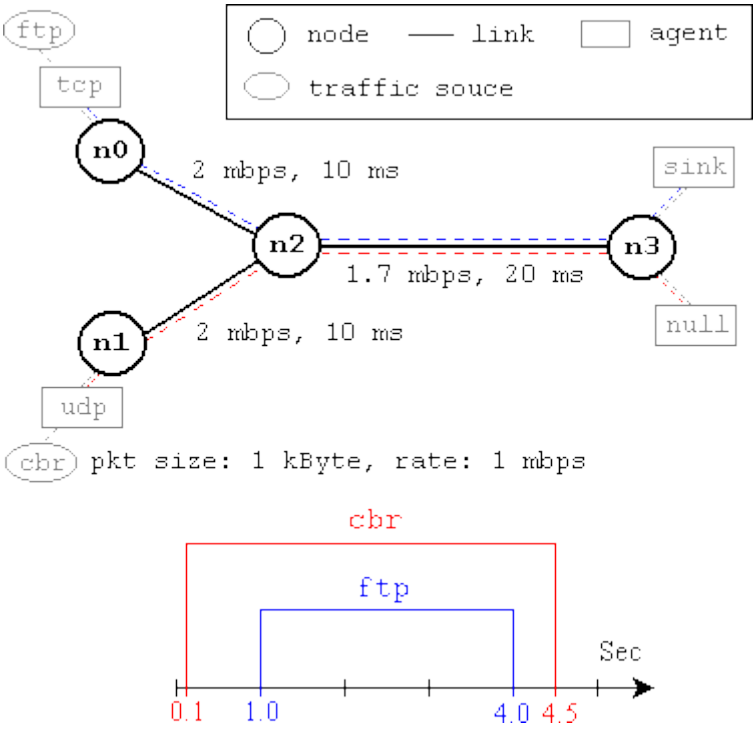
Architectural View of NS

The general architecture of NS. In this figure a general user (not an NS developer) can be thoughtof standing at the left bottom corner, designing and running simulations in Tcl using the simulatorobjects in the OTcl library. The event schedulers and most of the network components are implemented in C++ and available to OTcl through an OTcl linkage that is implemented using tcl.The whole thing together makes NS, which is a OO extended Tcl interpreter with network simulator libraries.

This section briefly examined the general structure and architecture of NS. At this point, one mightbe wondering about how to obtain NS simulation results.

When a simulation is finished, NS produces one or more text-based output files that contain detailed simulation data, if specified to do so in the input Tcl (or more specifically, OTcl) script. The data can be used for simulation analysis (two simulation result analysis examples are presentedin later sections) or as an input to a graphical simulation display tool called Network Animator (NAM) that is developed as a part of VINT project. NAM has a nice graphical user interface similar to that of a CD player (play, fast forward, rewind, pause and so on), and also has a display speed controller. Furthermore, it can graphically present information such as throughput and number of packet drops at each link, although the graphical information cannot be used for accuratesimulation analysis.

This section shows a simple NS simulation script and explains what each line does. OTcl scriptcreates the simple network configuration and runs the simulation scenario.



A Simple Network Topology and Simulation Scenario

This network consists of 4 nodes (n0, n1, n2, n3) as shown in above figure. The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex linkbetween n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "tcp" agent is attached to n0, and a connection is established to a tcp "sink" agent attached to n3. As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "udp" agent that is attached to n1 is connectedto a "null" agent attached to n3. A "null" agent just frees the packets received

The following is the explanation of the script above. In general, an NS script starts with makinga Simulator object instance.

set ns [new Simulator]: generates an NS simulator object instance, and assigns it to variable ns(italics is used for variables and values in this section).

- Initialize the packet format (ignore this for now) o Create a scheduler (default is calendarscheduler)
- Select the default address format (ignore this for now)
  - The "Simulator" object has member functions that do the following: Create compound objectssuch as nodes and links (described later) Connect network component objects created

(ex. attach-agent) Set network component parameters (mostly for compound objects)

- Create connections between agents (ex. make connection between a "tcp" and "sink") ▪

Specify NAM display options

Most of member functions are for simulation setup and scheduling, however some of them are for the NAM display. The "Simulator" object member function implementations are located in the "ns-2/tcl/lib/ns-lib.tcl" file.

$ns color fid color: is to set color of the packets for a flow specified by the flow id (fid). This member function of "Simulator" object is for the NAM display, and has no effect on the actualsimulation.

$ns namtrace-all file-descriptor: This member function tells the simulator to record simulation traces in NAM input format. It also gives the file name that the trace will be written to later by thecommand $ns flush-trace. Similarly, the member function trace-all is for recording the simulationtrace in a general format.

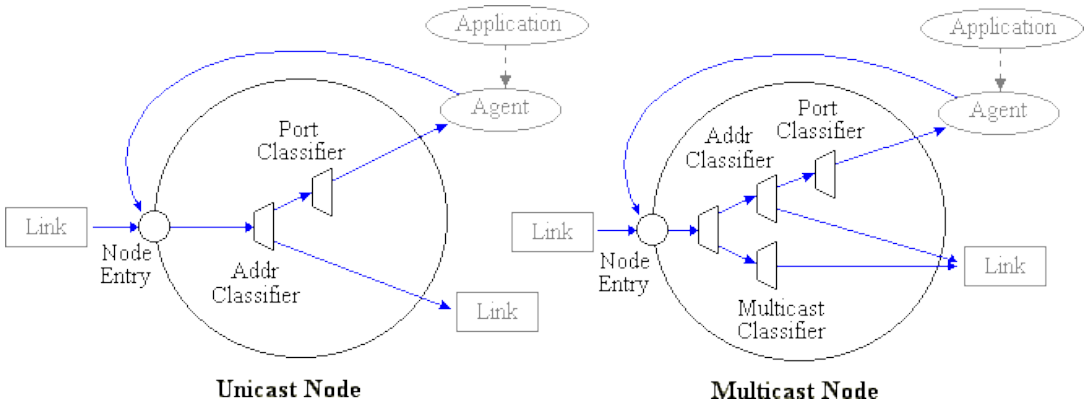
proc finish : is called after this simulation is over by the command $ns at 5.0 "finish". In

this function, post-simulation processes are specified. set n0 [$ns node]: The member functionnode creates a node. A node in NS is a compound object made of address and port classifiers. set tcp [new Agent/TCP]: This line shows how to create a TCP agent.

$ns at time "string": This member function of a Simulator object makes the scheduler(scheduler_ is the variable that points to the scheduler object created by [new Scheduler] command at the beginning of the script) to schedule the execution of the specified string at given simulation time. After all network configuration, scheduling and post-simulation procedurespecifications are done, the only thing left is to run the simulation. This is done by $ns run.

NODE AND ROUTING

A node is a compound object composed of a node entry object and classifiers as shown in Figure 7.There are two types of nodes in NS. A unicast node has an address classifier that does unicast routing and a port classifier. A multicast node, in addition, has a classifier that classify multicast packets from unicast packets and a multicast classifier that performs multicast routing



Node (Unicast and Multicast)

In NS, Unicast nodes are the default nodes. To create Multicast nodes the user must explicitly notify in the input OTcl script, right after creating a scheduler object, that all the nodes that will becreated are multicast nodes. After specifying the node type, the user can also select a specific routing protocol other than using a default one.

Unicast

- $ns rtproto type

- type: Static, Session, DV,

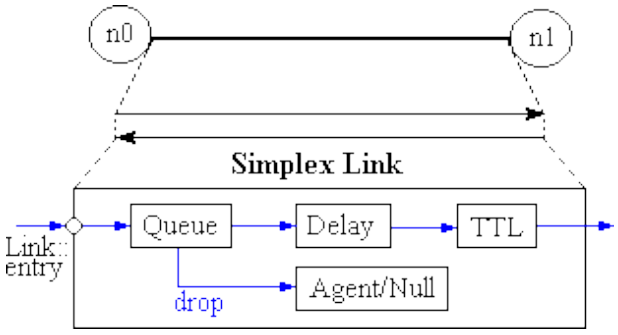
cost, multi-pathMulticast

- $ns multicast (right after set $ns [new Scheduler]) - $ns mrtproto type

- type: CtrMcast, DM, ST, BST

Link

A link is another major compound object in NS. When a user creates a link using a duplex-link member function of a Simulator object, two simplex links in bothdirections are created.

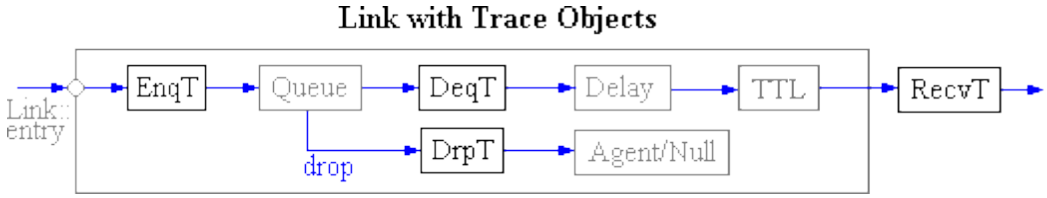


A Link

One thing to note is that an output queue of a node is actually implemented as a part of a simplex link object. Packets dequeued from a queue are passed to the Delay object that simulates the link delay, and packets dropped at a queue are sent to a Null Agent and are freed there. Finally, the TTLobject calculates Time To Live parameters for each packet received and updates the TTL field of the packet.

Tracing:

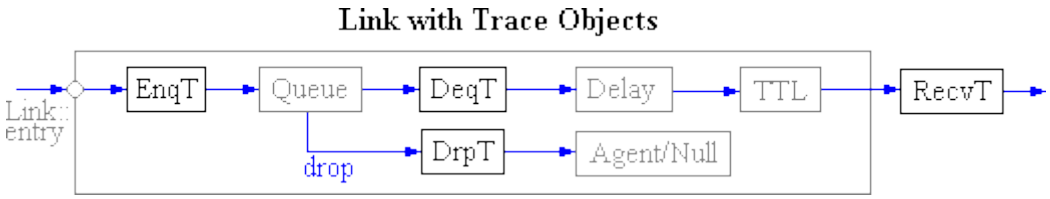
In NS, network activities are traced around simplex links. If the simulator is directed to trace network activities (specified using $ns trace-all file or $ns namtrace-all file), the links created afterthe command will have the following trace objects inserted as shown in Figure 9. Users can also specifically create a trace object of type type between the given src and dst nodes using the createtrace type file src dst command



Inserting Trace Objects

When each inserted trace object (i.e. EnqT, DeqT, DrpT and RecvT) receives a packet, it writes tothe specified trace file without consuming any simulation time, and passes the packet to the next network object.

Queue Monitor: Basically, tracing objects are designed to record packet arrival time at which theyare located. Although a user gets enough information from the trace, he or she might be interested in what is going on inside a specific output queue. For example, a user interested in RED queue behavior may want to measure the dynamics of average queue size and current queue size of a specific RED queue (i.e. need for queue monitoring). Queue monitoring can be achieved using queue monitor objects and snoop queue objects

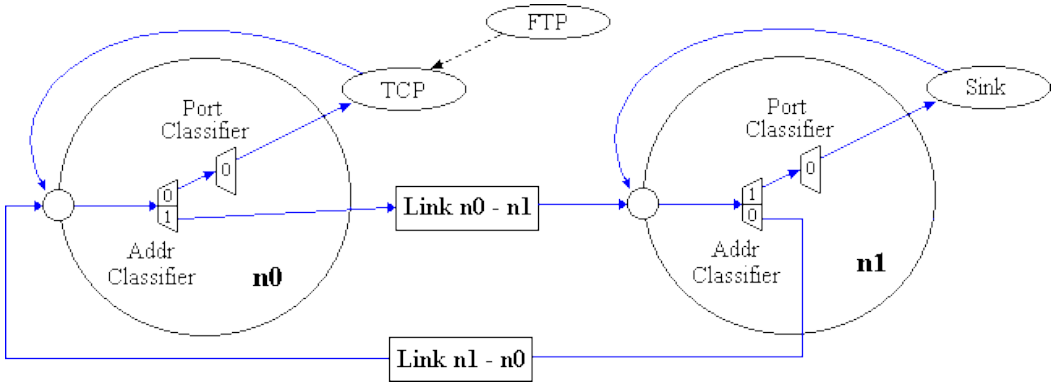


Monitoring Queue:

When a packet arrives, a snoop queue object notifies the queue monitor object of this event. The queue monitor using this information monitors the queue. A RED queue monitoring example is shown in the RED Queue Monitor Example section. Note that snoop queue objects can be used inparallel with tracing objects even though it is not shown in the above figure.

Packet flow example:

Until now, the two most important network components (node and link) were examined. This Figure shows is example simulation network setup and packet flow. The network consist of two nodes (n0 and n1) of which the network addresses are 0 and 1 respectively. A TCP agent attachedto n0 using port 0 communicates with a TCP sink object attached to n1 port 0. Finally, an FTP application (or traffic source) is attached to the TCP agent, asking to send some amount of data.

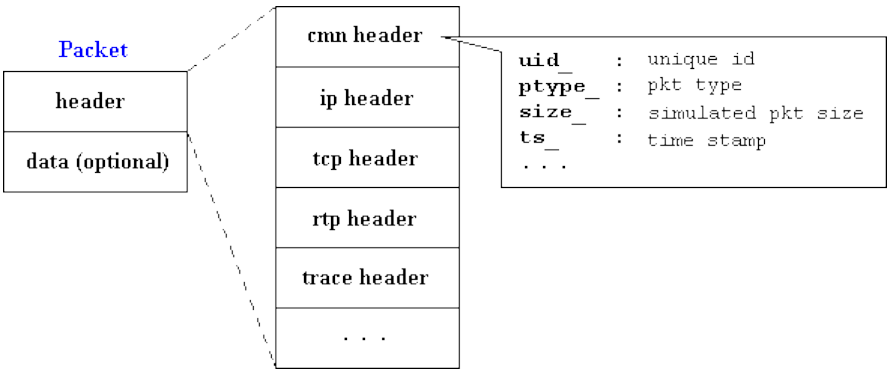


Packet Flow Example

Note that the above figure does not show the exact behavior of a FTP over TCP. It only shows thedetailed internals of simulation network setup and a packet flow.

Packet:

A NS packet is composed of a stack of headers, and an optional data space. The header format isinitialized when a Simulator object is created, where a stack of all registered (or possibly useable)headers, such as the common header that is commonly used by any objects as needed, IP header, TCP header, RTP header (UDP uses RTP header) and trace header, is defined, and the offset of each header in the stack is recorded. What this means is that whether or not a specific header is used, a stack composed of all registered headers is created when a packet is allocated by an agent, and a network object can access any header in the stack of a packet it processes using the corresponding offset value.



NS Packet Format

Usually, a packet only has the header stack (and a data space pointer that is null). Although a packet can carry actual data (from an application) by allocating a data space, very few applicationand agent implementations support this. This is because it is meaningless to carry data around in anon-real-time simulation. However, if you want to implement an application that talks to another application cross the network, you might want to use this feature with a little modification in the underlying agent implementation. Another possible approach would be creating a new header for the application and modifying the underlying agent to write data received from the application to the new header. The second approach is shown as an example in a later section called "Add New Application and Agent".

RESULT:

The NS2 (Network Simulator 2) has been studied.

Ex No:13 **IMPLEMENTATION OF STAR TOPOLOGY**

Date: 04/09/2021

AIM:

To implement star topology using Cisco packet tracer.

OBJECTIVE:

To learn how to implement star topology.

SOFTWARE REQUIRED:

Cisco packet tracer.

PROCEDURE :

STEP 1: Open the Cisco packet tracer.

STEP 2: Construct 2 start network each switch in centre connected to devices like PC's. Both the star networks are connected through HUB.

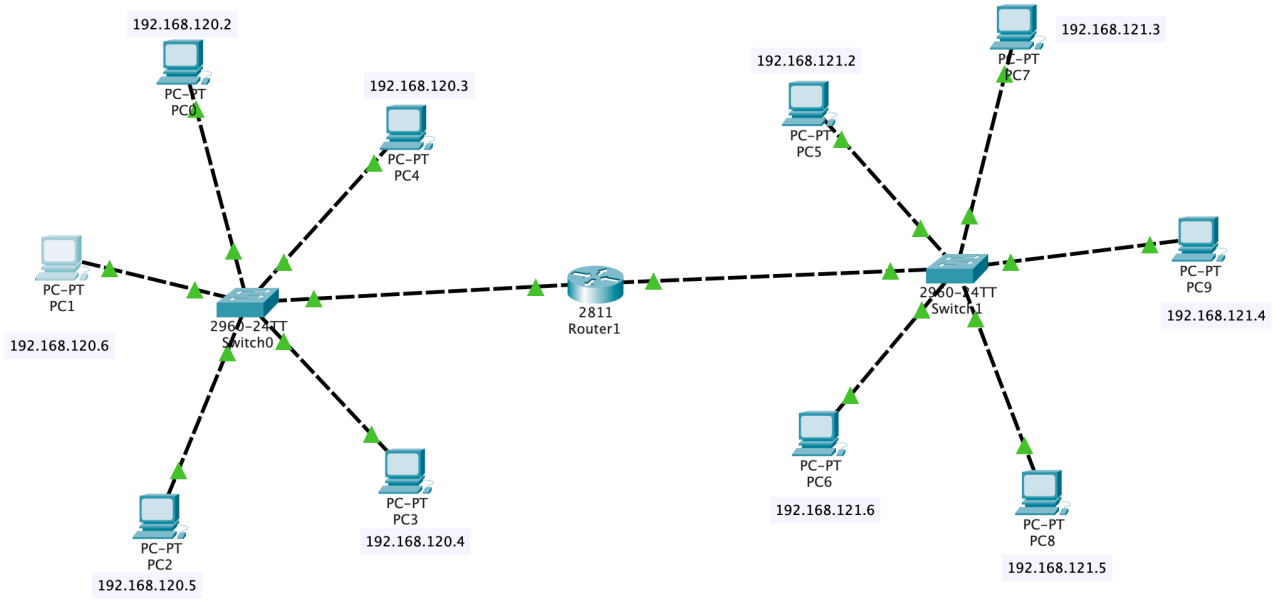
STEP 3: star network on the left side of HUB is 121 network and sight side is 121 network. step 4: click on the PC, select desktop and click on ip configuration Assign ip address

STEP 5: Repeat step 4 for all the devices Connected to switch and dose close.

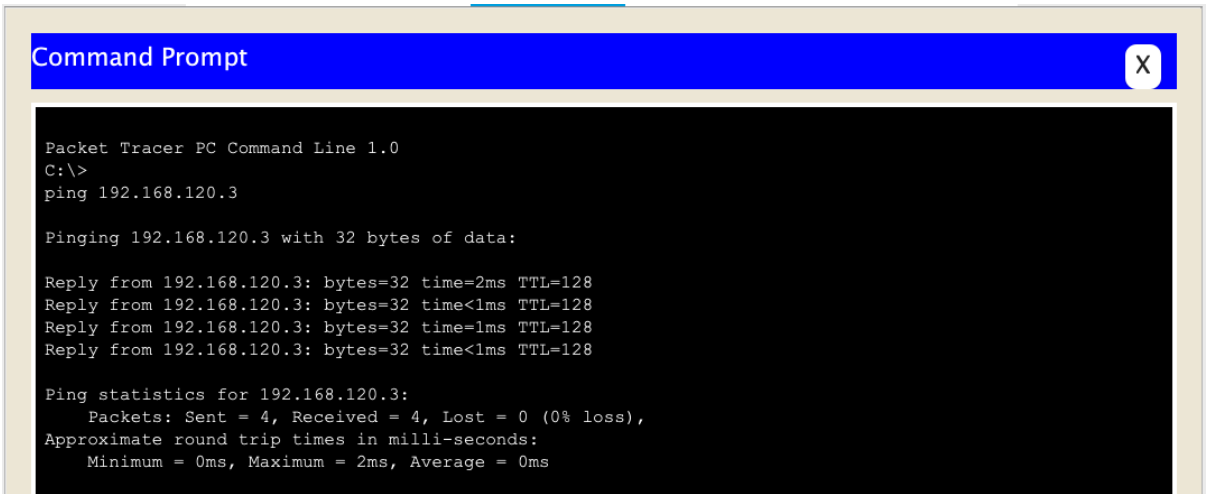
STEP 6: Place a note of ip address near the devices for reference.

STEP 7: Go to command prompt of PC2 and ping PC3 of another network. If there is zero less than the connection is successful.

STAR TOPOLOGY :



OUTPUT :



RESULT:

Thus, the star topology has been implemented using Cisco packet tracer.

Ex No : 14 **IMPLEMENTATION OF BUS TOPOLOGY**

Date : 04/09/2021

AIM:

To implement bus topology using Cisco packet tracer.

OBJECTIVE:

To learn how to implement star topology.

SOFTWARE REQUIRED:

Cisco packet tracer

PROCEDURE :

STEP 1: Open Cisco packet tracer.

STEP 2: Place 4 PC's and 4 switches in horizontal manner and also a router,

STEP 3: Form a connection between all devices.

STEP 4: Select the PC and click on the desktop in menu and go to ip configuration.

STEP 5: Assign ip address to the device and close.

STEP 6: Let there be 2 networks (1.e) network on left side of router be 121 and on right side be 122.

STEP 7: Repeat step 4 and 5 to all devices.

STEP 8: Place a note of ip address near the device for reference.

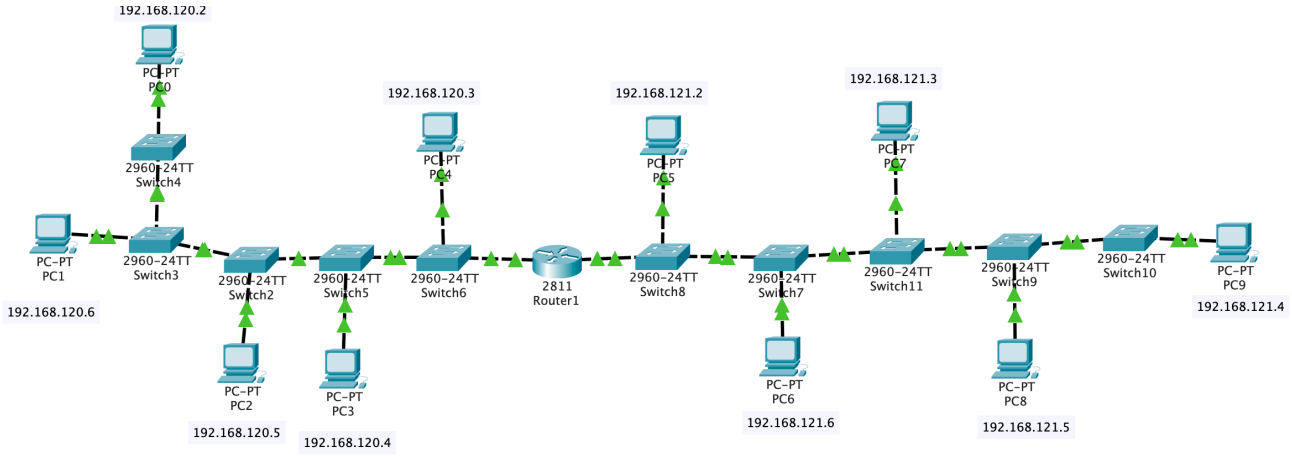
STEP 9: To activate the router, click on the router and select configuration in the menu. Then click interface - Fast Ethernet 0/0 for left network and Fast Ethernet 0/1 for ought network.

STEP 10: Give default gateway ip address to all de vices connected to switch.

STEP 11: select a PC of 121 network, click on desktop in menu and select command prompt.

STEP 12: Ping on a PC of 122 network and check the packet send, received and loss. If the loss is zero then network is correct.

BUS TOPOLOGY:




```
Packet Tracer PC Command Line 1.0
C:\>
ping 192.168.121.4

Pinging 192.168.121.4 with 32 bytes of data:

Request timed out.
Reply from 192.168.121.4: bytes=32 time=21ms TTL=127
Reply from 192.168.121.4: bytes=32 time=8ms TTL=127
Reply from 192.168.121.4: bytes=32 time=23ms TTL=127

Ping statistics for 192.168.121.4:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 23ms, Average = 17ms

C:\>ping 192.168.121.4

Pinging 192.168.121.4 with 32 bytes of data:

Reply from 192.168.121.4: bytes=32 time=1ms TTL=127
Reply from 192.168.121.4: bytes=32 time=23ms TTL=127
Reply from 192.168.121.4: bytes=32 time<1ms TTL=127
Reply from 192.168.121.4: bytes=32 time=22ms TTL=127

Ping statistics for 192.168.121.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 23ms, Average = 11ms

C:\>
```

RESULT:

Thus, the bus topology has been implemented using Cisco packet

Ex No:15 **IMPLEMENTATION OF RING TOPOLOGY**

Date: **04/09/2021**

AIM:

To implement ring topology using Cisco packet tracer.

OBJECTIVE:

To learn how to implement ring topology.

SOFTWARE REQUIRED:

Cisco packet tracer.

PROCEDURE :

STEP 1: Open the Cisco packet tracer

STEP 2: Connect 4 switches in sing format and connect PC's to each switch.

STEP 3: click on the PC, select desktop and select ip configuration and give ip address.

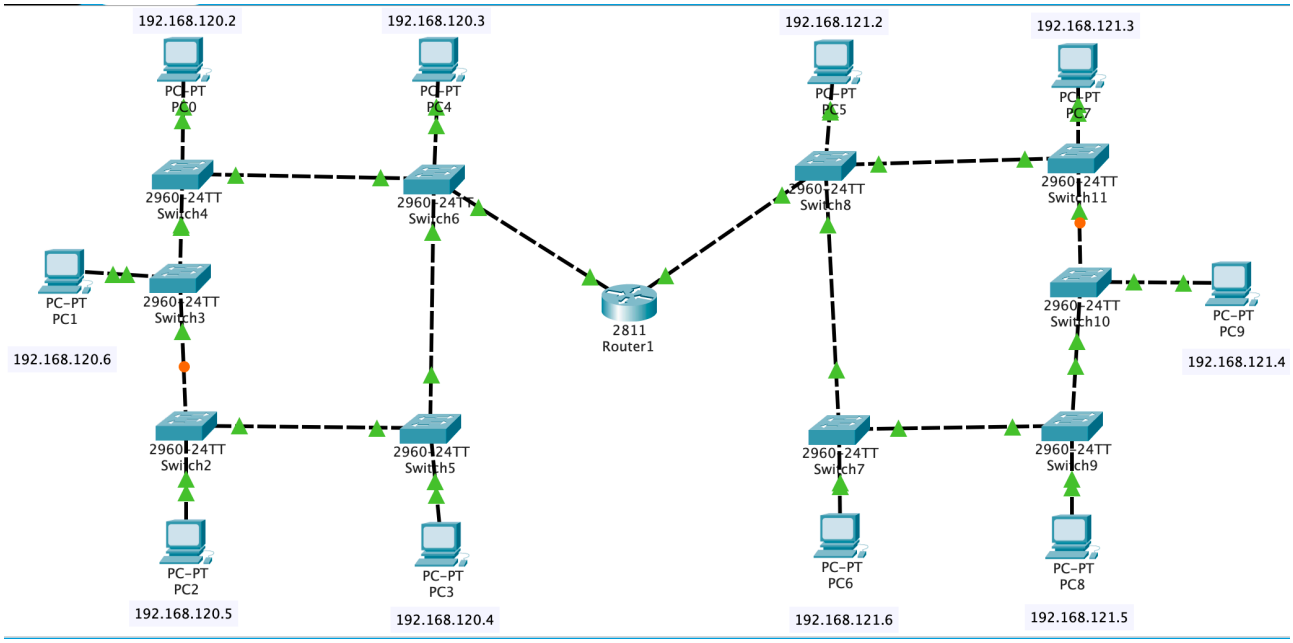
STEP 4: Place the PC a note of ip address near for reference.

STEP 5: Repeat the step 3 and 4 for other devices also.

STEP 6: click on the PCO and go to command-prompt to ping PC6.

STEP 7: If there is zeroes packet less then connection is successful.

RING TOPOLOGY :



OUTPUT :

```
Packet Tracer PC Command Line 1.0
C:\>
ping 192.168.121.6

Pinging 192.168.121.6 with 32 bytes of data:

Request timed out.
Reply from 192.168.121.6: bytes=32 time=1ms TTL=127
Reply from 192.168.121.6: bytes=32 time=1ms TTL=127
Reply from 192.168.121.6: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.121.6:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```

RESULT:

Thus, the bus topology has been implemented using Cisco packet tracer.

Ex No : 16 **IMPLEMENTATION OF**

Date: 05/10/2021 **DISTANCE VECTOR ROUTING**

AIM :

To Implement Distance Vector Routing using virtsim.

OBJECTIVE :

To learn about Distance Vector Routing using virtsim.

SOFTWARE REQUIRED :

VIR-TISM

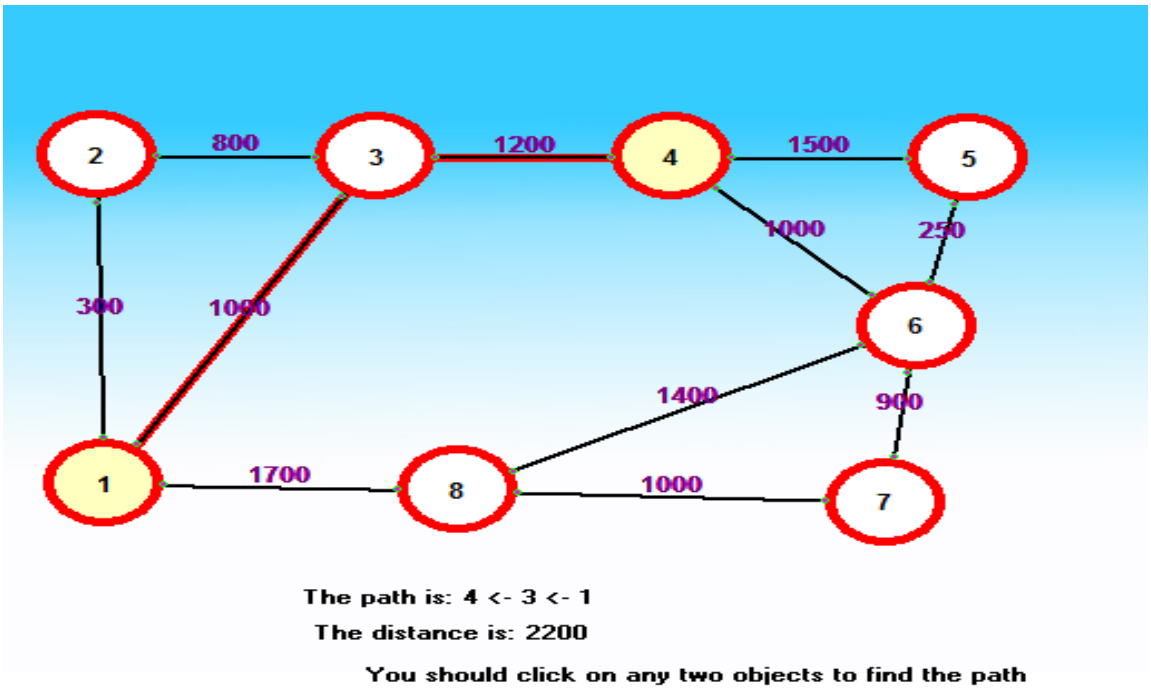
DESCRIPTION:

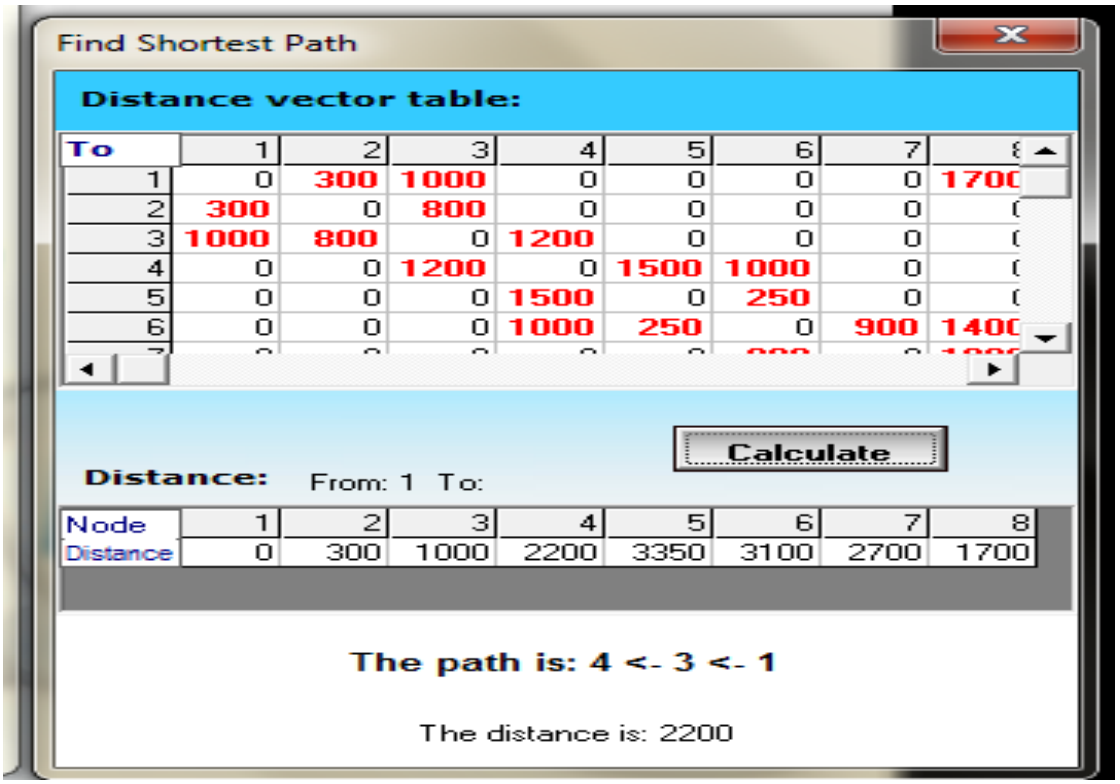
A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology **changes periodically**, known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).Distances, based on a chosen metric, are computed using information from the neighbors' distance vectors.

DISTANCE VECTOR ROUTING ALGORITHM:

Paths from 1 to 4
1. 1 -> 2 -> 3 -> 4
2. 1 -> 3 -> 4
3. 1 -> 8 -> 6 -> 4
4. 1 -> 8 -> 6 -> 5 -> 4
5. 1 -> 8 -> 7 -> 6 -> 4
6. 1 -> 8 -> 7 -> 6 -> 5 -> 4

OUTPUT:





RESULT:

Thus, successfully implement the distance vector routing using VIRTISM.

EX.NO : 17 **SIMULATION OF LINK STATE ROUTING ALGORITHM**

DATE : 05/10/2021

AIM:

To stimulate link state routing algorithm.

OBJECTIVE:

To understand the concepts of link state routing algorithm.

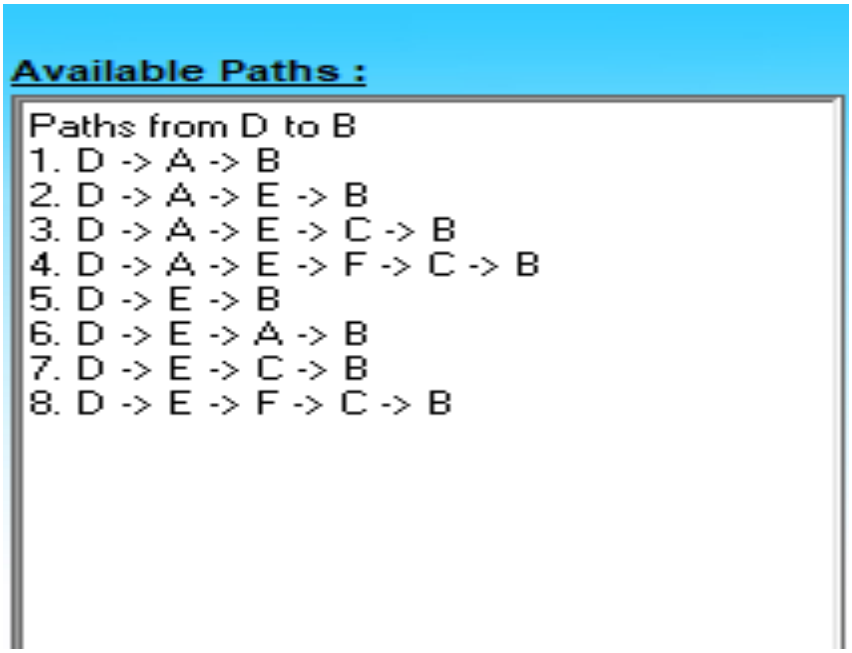
SOFTWARE REQUIRED:

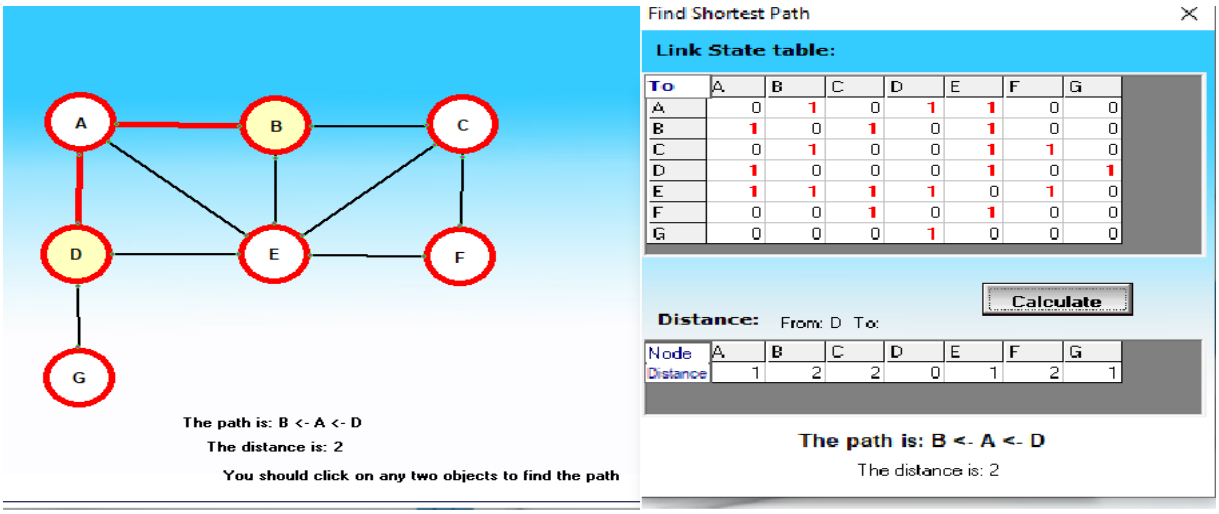
VIR-TISM

PROCEDURE:

1. This algorithm is based on minimum spanning tree.
2. Select vir-tsim application.
3. Select the stimulation menu and then select link state routing algorithm.
4. Select the required source and destination nodes.
5. Click on the find path button.
6. All the possible paths from source to destination node is displayed inavailable section.
7. The shortest path from source to destination node is displayed in available path sector distance shortest path from source node to every other-node isalso displayed.

OUTPUT:





RESULT:

The link state routing algorithm was executed successfully.

Ex No : 19 **IMPLEMENTATION OF ENCRYPTION AND DECRYPTION**

Date : 05/10/2021

AIM:

To implement the encryption and decryption concepts.

OBJECTIVE:

To learn how to implement the encryption and decryption.

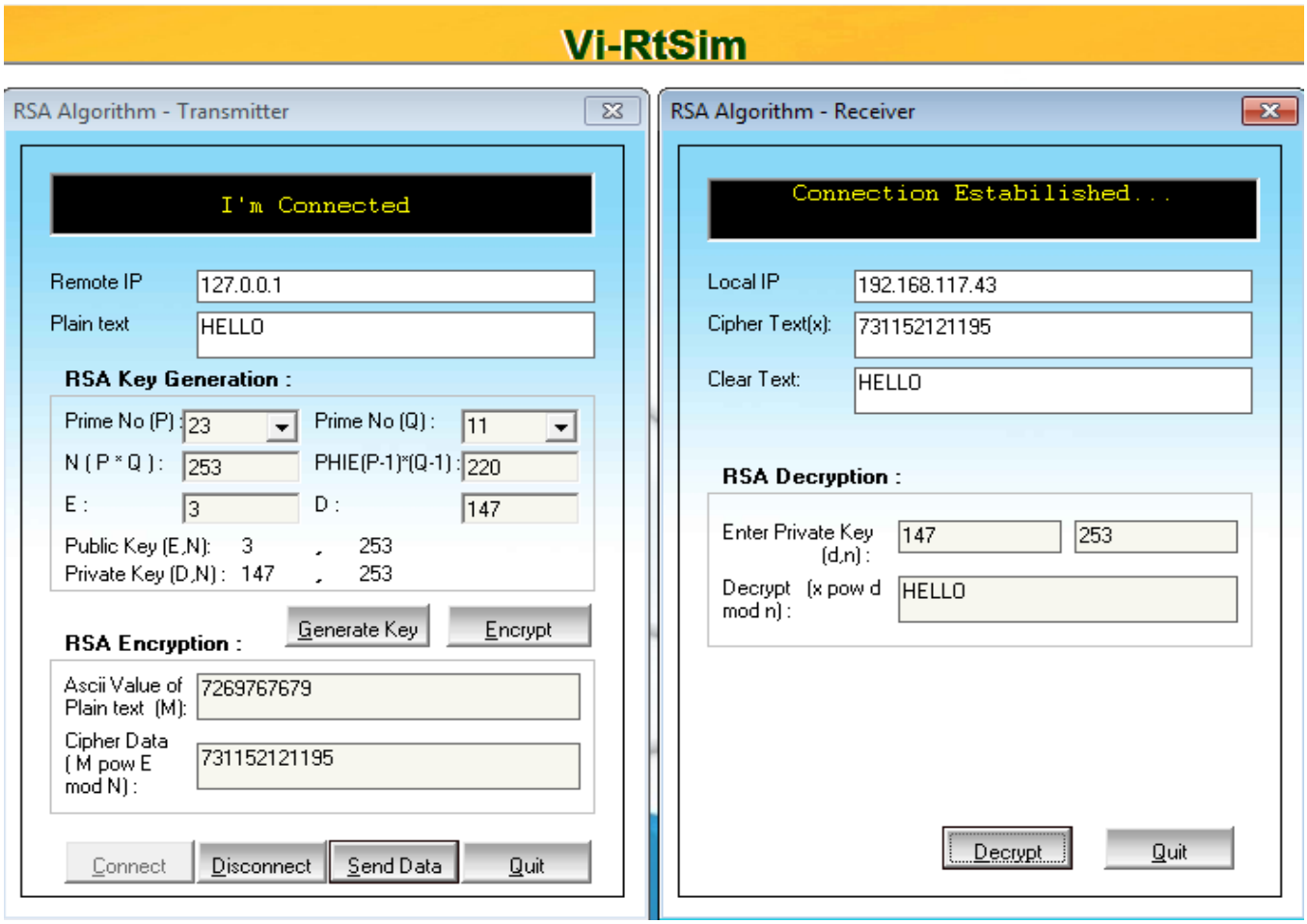
SOFTWARE REQUIRED:

VI-RTISM

DESCRIPTION:

when the sender sends a message or some data to the receiver the data is first encrypted using the encryption algorithm and a key. After that encrypted message or data is decrypted on the receiver sideusing decryption algorithm and a key. And finally the receiver is receiving the original data.

OUTPUT:



RESULT:

Thus, the encryption and decryption are implemented using VIRTISM.