## 1:BIT STUFFING

```
def receiver(frame):
    msg = ''.join(str(frame[i]) for i in range(8, len(frame)-8))
    msg = msg.replace('111110', '11111')
    print("Received message is: ", msg)

def sender():
    data = input("Enter the data bits: ")
    frame = '01111110' + data.replace('11111', '111110') + '01111110'
    print("Length of frame sent: ", len(frame))
    print("Frame sent: ", frame)
    receiver(list(map(int, frame)))

sender()
```

## 2: FRAME COUNT

```
def sender():
    frames = [input(f"Enter the frame {i+1}: ")
for i in range(int(input("Enter the number of frames: ")))]
    return frames

def receiver():
    frames = sender()
    print("Received frames:")
    for frame in frames:
        frame_size = len(frame.encode())
        print(f"Frame: {frame}, Size: {frame_size} bytes")

receiver()
```

## 3:DISTANCE VECTOR

```
n = int(input("Enter the number of nodes: "))
costmat = [list(map(int, input(f"Enter the cost matrix for node {i+1}: ").split()))
for i in range(n)]
via = [[j for j in range(n)] for i in range(n)]

for k in range(n):
    for i in range(n):
        for j in range(n):
            if costmat[i][j] > costmat[i][k] + costmat[k][j]:
                costmat[i][j] = costmat[i][k] + costmat[k][j]
                via[i][j] = k

for i in range(n):
    print("\nFor router:", i+1)
    for j in range(n):
        print("\tnode:", j+1, "via:", via[i][j]+1, "Distance:", costmat[i][j])
```

## 4:LEAKY BUCKET

```
import random
import time
def flow(pktsize, output):
    buketsize = 512
    if pktsize > buketsize:
        print("Bucket overflow")
    else:
        time.sleep(1)
        while pktsize > output:
            print(str(output) + " bytes outflow")
            pktsize = pktsize - output
        if pktsize > 0:
            print(str(pktsize) + " bytes outflow")
def main():
    output = int(input("Enter output rate: "))
    n = int(input("Enter number of packets: "))
    for i in range(1, n+1):
        pktsize = random.randint(0, 999)
        print("Packet No: " + str(i) + " packetsize = " + str(pktsize))
        flow(pktsize, output)
if __name__ == "__main__":
    main()
```

## 5:CRC

```c
#include <stdio.h>
#include <string.h>

int main() {
    char data[100];
    char divisor[30];
    char temp[30];
    char rem[30];
    char zero[30];
    int keylen, msglen, ch, f;

    printf("Enter Data: ");
    scanf("%s", data);

    printf("Enter divisor: ");
    scanf("%s", divisor);

    keylen = strlen(divisor);
    msglen = strlen(data);

    for (int i = 0; i < keylen - 1; i++)
        data[msglen + i] = '0';

    printf("\ndata after appending zeros: %s\n", data);

    for (int i = 0; i < keylen; i++)
        zero[i] = '0';

    for (int i = 0; i < keylen; i++)
        temp[i] = data[i];

    for (int i = 0; i < msglen; i++) {
        if (temp[0] == '0')
            strcpy(rem, zero);
        else
            strcpy(rem, divisor);

        for (int j = 0; j < keylen - 1; j++)
            rem[j] = (temp[j + 1] == rem[j + 1]) ? '0' : '1';

        rem[keylen - 1] = data[i + keylen];
        strcpy(temp, rem);
    }

    strcat(data, rem);
    printf("\nRemainder is %s\n", rem);
    printf("\ndata after appending remainder: %s\n", data);

    printf("\nDo you want to introduce an error (Y/N)? ");
    getchar(); // Consume the newline character
    ch = getchar();
    if (ch == 'Y' || ch == 'y')
        data[msglen / 2] = (data[msglen / 2] == '0') ? '1' : '0';

    for (int i = 0; i < keylen; i++)
        temp[i] = data[i];

    for (int i = 0; i < msglen; i++) {
        if (temp[0] == '0')
            strcpy(rem, zero);
        else
            strcpy(rem, divisor);

        for (int j = 0; j < keylen - 1; j++)
            rem[j] = (temp[j + 1] == rem[j + 1]) ? '0' : '1';

        rem[keylen - 1] = data[i + keylen];
        strcpy(temp, rem);
    }

    printf("\nData obtained: %s\n", data);
    printf("Remainder is %s\n", rem);

    f = 1;
    for (int i = 0; i < keylen - 1; i++) {
        if (rem[i] != '0') {
            f = 0;
            break;
        }
    }

    if (f == 1) {
        printf("No Error Occurred. Final data is: ");
        for (int i = 0; i < msglen; i++)
            printf("%c", data[i]);
        printf("\n");
    } else {
        printf("Error Occurred\n");
    }

    return 0;
}
```

## 6: TCP:

*SERVER.PY*

```
import socket, os
HOST = "localhost" # The server's hostname or IP address
PORT = 5000 # The port used by the server
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
 s.bind((HOST, PORT))
```

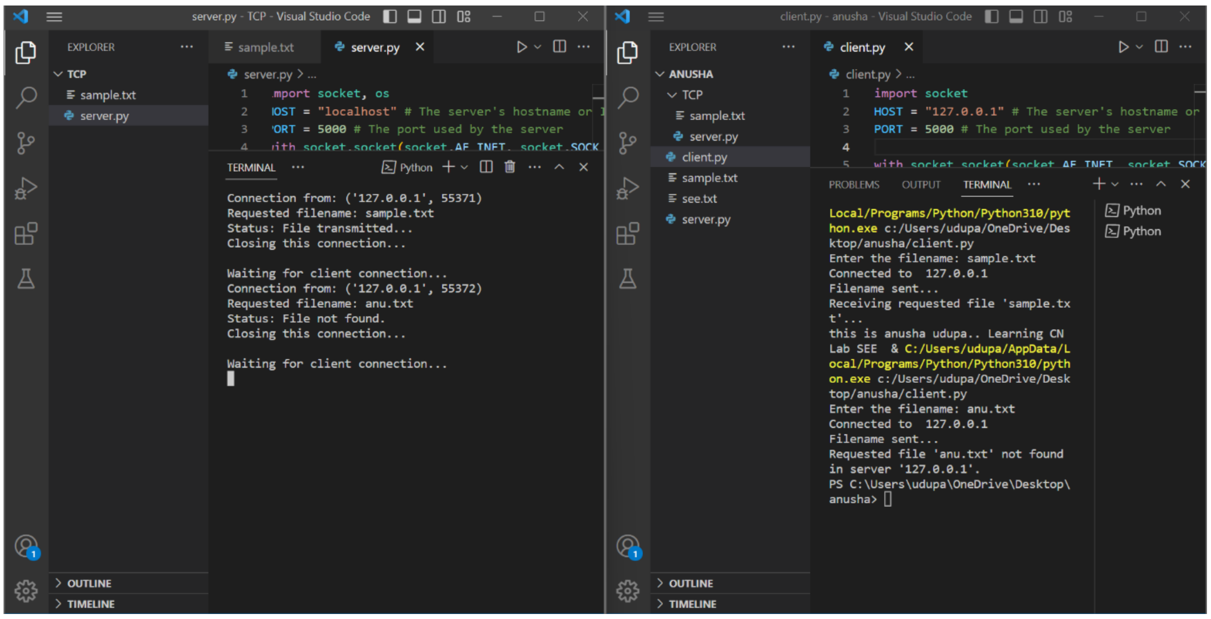
```python
s.listen(1)
while True:
    print('\nWaiting for client connection...')
    conn, addr = s.accept()
    with conn:
        print("Connection from:", addr)
        while True:
            filename = conn.recv(1024).decode()
            if not filename:
                break
            print('Requested filename:', filename)
            if not os.path.exists(filename):
                print('Status: File not found.')
                conn.sendall(b'file not found')
            else:
                with open(filename) as file:
                    conn.sendall(file.read().encode())
                print('Status: File transmitted...')
                break
        print("Closing this connection...")
```

*CLIENT.PY*

```python
import socket
HOST = "127.0.0.1" # The server's hostname or IP address
PORT = 5000 # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    filename = input('Enter the filename: ')
    s.connect((HOST, PORT))
    print('Connected to ', HOST)
    s.sendall(filename.encode())
    print('Filename sent...')
    data = s.recv(1024).decode()
    if data.startswith('file not found'):
        print(f'Requested file {filename!r} not found in server {HOST!r}.')
    else:
        print(f'Receiving requested file {filename!r}...', end='\n')
        with open(filename, 'w') as file:
            while True:
                file.write(data)
                if not data:
                    break
                data = s.recv(1024).decode()

        with open(filename,'r') as filer:
            print(filer.readline())

        print('done.')
    s.close()
```

*create a file: abc.txt in same folder as server.py and client.py*

*terminal-> split terminal*



```
.mport socket, os
IOST = "localhost" # The server's hostname or I
'ORT = 5000 # The port used by the server
vith socket.socket(socket.AF_INET, socket.SOCK
```

```
Connection from: ('127.0.0.1', 55371)
Requested filename: sample.txt
Status: File transmitted...
Closing this connection...

Waiting for client connection...
Connection from: ('127.0.0.1', 55372)
Requested filename: anu.txt
Status: File not found.
Closing this connection...

Waiting for client connection...
```

```
import socket
HOST = "127.0.0.1" # The server's hostname or
PORT = 5000 # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK
```

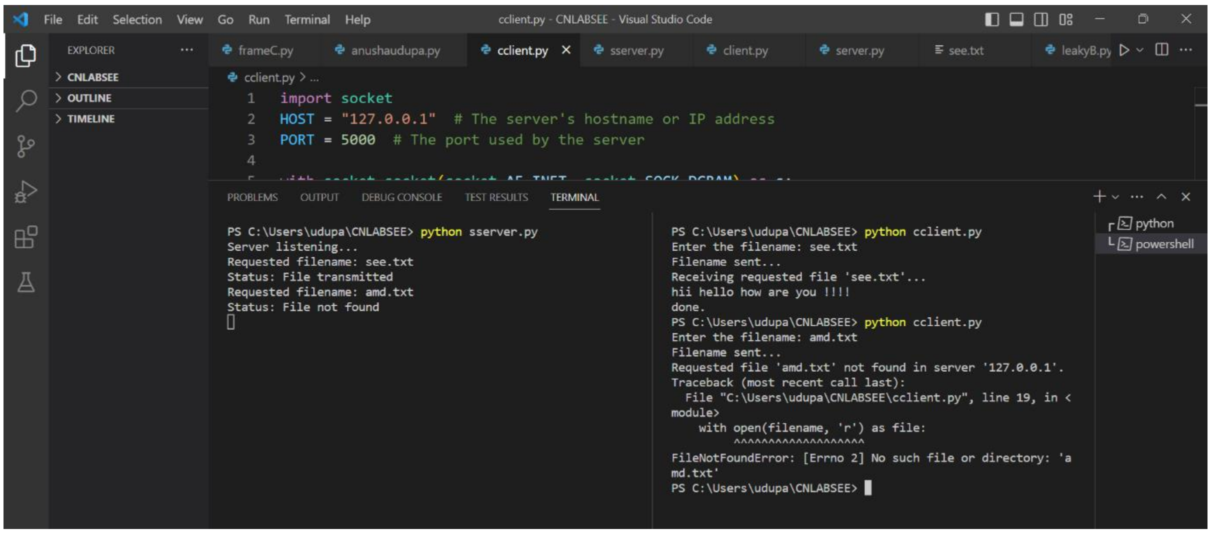
```
Local/Programs/Python/Python310/pyt
hon.exe c:/Users/udupa/OneDrive/Des
ktop/anusha/client.py
Enter the filename: sample.txt
Connected to  127.0.0.1
Filename sent...
Receiving requested file 'sample.tx
t'...
this is anusha udupa.. Learning CN
Lab SEE  & C:/Users/udupa/AppData/L
ocal/Programs/Python/Python310/pyth
on.exe c:/Users/udupa/OneDrive/Desk
top/anusha/client.py
Enter the filename: anu.txt
Connected to  127.0.0.1
Filename sent...
Requested file 'anu.txt' not found
in server '127.0.0.1'.
PS C:\Users\udupa\OneDrive\Desktop\
anusha>
```

***7:UDP*** *Server*

```python
import socket
HOST = "localhost"  # The server's hostname or IP address
PORT = 5000  # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    s.bind((HOST, PORT))
    print('Server listening...')
    while True:
        data, addr = s.recvfrom(1024)
        filename = data.decode()
        print('Requested filename:', filename)
        try:
            with open(filename, 'r') as file:
                file_data = file.read()
                s.sendto(file_data.encode(), addr)
                print('Status: File transmitted')
        except FileNotFoundError:
            s.sendto(b'file not found', addr)
            print('Status: File not found')
```

*client*

```python
import socket
HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 5000  # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    filename = input('Enter the filename: ')
    s.sendto(filename.encode(), (HOST, PORT))
    print('Filename sent...')
    while True:
        data, addr = s.recvfrom(1024)
        if data.startswith(b'file not found'):
            print(f'Requested file {filename!r} not found in
server {HOST!r}.')
            break
        else:
            with open(filename, 'w') as file:
                file.write(data.decode())
            print(f'Receiving requested file {filename!r}...')
            break
    with open(filename, 'r') as file:
        print(file.readline())
    print('done.')
```



```
import socket
HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 5000  # The port used by the server
```

```
PS C:\Users\udupa\CNLABSEE> python sserver.py
Server listening...
Requested filename: see.txt
Status: File transmitted
Requested filename: amd.txt
Status: File not found
```

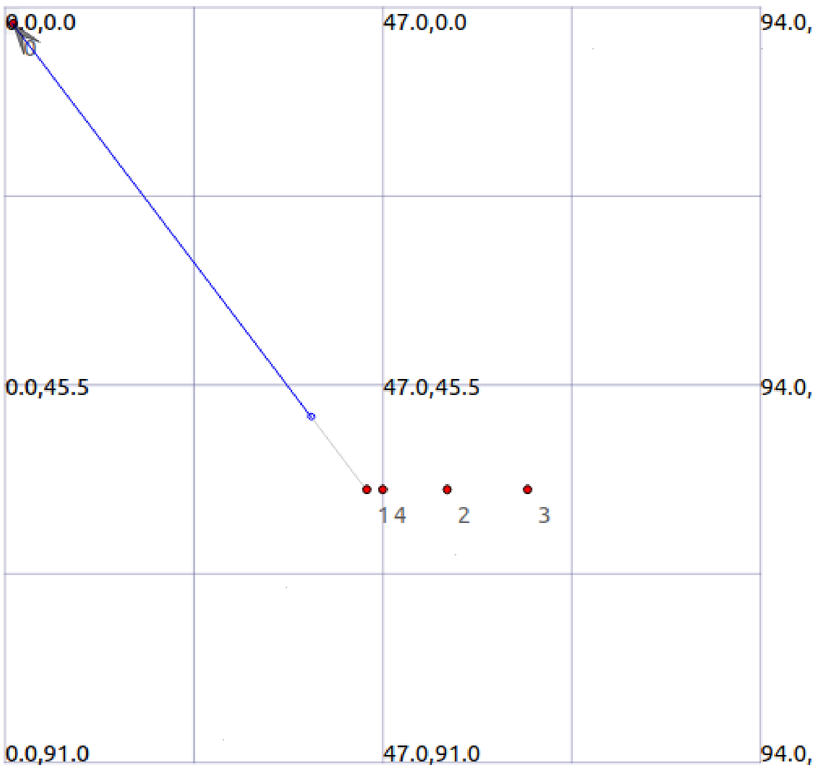
```
PS C:\Users\udupa\CNLABSEE> python cclient.py
Enter the filename: see.txt
Filename sent...
Receiving requested file 'see.txt'...
hii hello how are you !!!!
done.
PS C:\Users\udupa\CNLABSEE> python cclient.py
Enter the filename: amd.txt
Filename sent...
Requested file 'amd.txt' not found in server '127.0.0.1'.
Traceback (most recent call last):
  File "C:\Users\udupa\CNLABSEE\cclient.py", line 19, in <
module>
    with open(filename, 'r') as file:
         ^^^^^^^^^^^^^^^^^^^
FileNotFoundError: [Errno 2] No such file or directory: 'a
md.txt'
PS C:\Users\udupa\CNLABSEE>
```

**PROGRAM 1:**
**Simulate peer-to-peer communication between a client and a server using Point-to-Point protocol. Apply NetAnim software to demonstrate the scenario graphically. Analyze packet parameters by creating trace file usingAscii trace metrics.**

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"

// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1
//    point-to-point
//

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
  CommandLine cmd (__FILE__);
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
  std::string animFile="first.xml";
  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");

  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  AnimationInterface anim(animFile);
  anim.SetConstantPosition(nodes.Get(0),1.0,2.0);
  anim.SetConstantPosition(nodes.Get(1),45.0,60.0);

  AsciiTraceHelper ascii;
  pointToPoint.EnableAsciiAll(ascii.CreateFileStream("first.tr"));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

**PROGRAM 2:**
**Simulate to implement a bus topology using Point-to-Point protocol betweena client and a LAN with 4 nodes. The LAN use CSMA during packet transmission. Apply NetAnim software to demonstrate the scenario graphically. Analyze packet parameters by creating trace file using Ascii trace metrics.**

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 10.1.2.0


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

  cmd.Parse (argc,argv);

  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

std::string animFile="second.xml";
 nCsma = nCsma == 0 ? 1 : nCsma;

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

  NodeContainer csmaNodes;
  csmaNodes.Add (p2pNodes.Get (1));
  csmaNodes.Create (nCsma);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer p2pDevices;
  p2pDevices = pointToPoint.Install (p2pNodes);

  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
  csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
  NetDeviceContainer csmaDevices;
  csmaDevices = csma.Install (csmaNodes);

  InternetStackHelper stack;
  stack.Install (p2pNodes.Get (0));
  stack.Install (csmaNodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces;
  p2pInterfaces = address.Assign (p2pDevices);

  address.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer csmaInterfaces;
  csmaInterfaces = address.Assign (csmaDevices);

  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  //pointToPoint.EnablePcapAll ("second");
  //csma.EnablePcap ("second", csmaDevices.Get (1), true);

  AnimationInterface anim(animFile);
  anim.SetConstantPosition(p2pNodes.Get(0),1.0,2.0);
  anim.SetConstantPosition(p2pNodes.Get(1),45.0,60.0);
  anim.SetConstantPosition(csmaNodes.Get(1),55.0,60.0);
  anim.SetConstantPosition(csmaNodes.Get(2),65.0,60.0);
  anim.SetConstantPosition(csmaNodes.Get(3),75.0,60.0);

  AsciiTraceHelper ascii;
  pointToPoint.EnableAsciiAll(ascii.CreateFileStream("p2p.tr"));
  csma.EnableAsciiAll(ascii.CreateFileStream("csma.tr"));

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

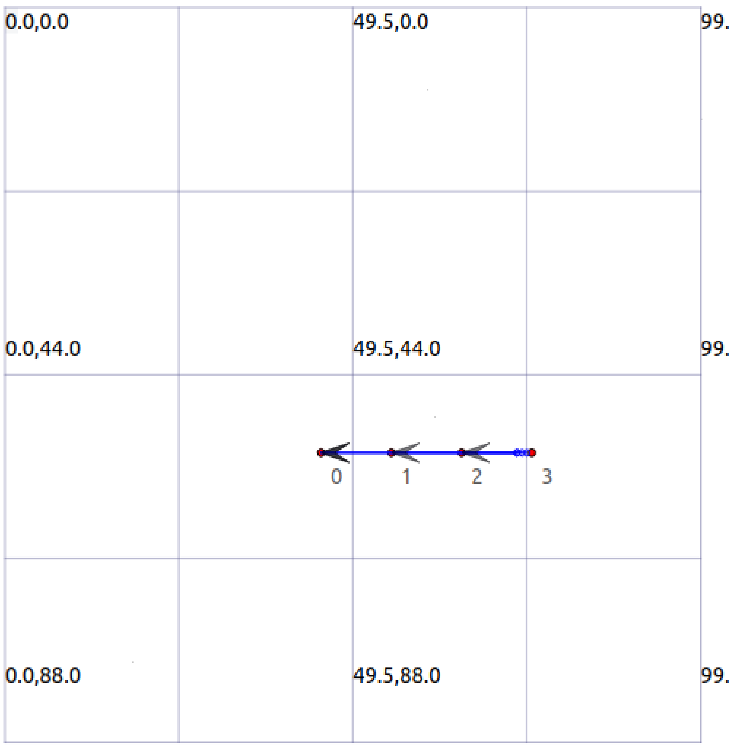


**PROGRAM 3: Simulate peer-to-peer communication between a client and a server using CSMA protocol. Apply NetAnim software to demonstrate the scenario graphically. Analyze packet parameters by creating trace file using Ascii trace metrics.**

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"
```

```
// Default Network Topology
//
//       10.1.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 10.1.2.0


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 4;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

  cmd.Parse (argc,argv);

  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

   std::string animFile="third.xml";
  nCsma = nCsma == 0 ? 1 : nCsma;

  //NodeContainer p2pNodes;
  //p2pNodes.Create (2);

  NodeContainer csmaNodes;
  csmaNodes.Create (nCsma);

  //PointToPointHelper pointToPoint;
  //pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  //pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  //NetDeviceContainer p2pDevices;
 // p2pDevices = pointToPoint.Install (p2pNodes);

  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
  csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

  NetDeviceContainer csmaDevices;
  csmaDevices = csma.Install (csmaNodes);

  InternetStackHelper stack;
  //stack.Install (p2pNodes.Get (0));
  stack.Install (csmaNodes);

  Ipv4AddressHelper address;
  //address.SetBase ("10.1.1.0", "255.255.255.0");
  //Ipv4InterfaceContainer p2pInterfaces;
  //p2pInterfaces = address.Assign (p2pDevices);

  address.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer csmaInterfaces;
  csmaInterfaces = address.Assign (csmaDevices);

  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (3));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (3), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps = echoClient.Install (csmaNodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  //Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  //pointToPoint.EnablePcapAll ("second");

 //csma.EnablePcap ("second", csmaDevices.Get (1), true);
 AnimationInterface anim(animFile);
  //anim.SetConstantPosition(p2pNodes.Get(0),1.0,2.0);
  anim.SetConstantPosition(csmaNodes.Get(0),45.0,60.0);
  anim.SetConstantPosition(csmaNodes.Get(1),55.0,60.0);
  anim.SetConstantPosition(csmaNodes.Get(2),65.0,60.0);
  anim.SetConstantPosition(csmaNodes.Get(3),75.0,60.0);

  AsciiTraceHelper ascii;
  //pointToPoint.EnableAsciiAll(ascii.CreateFileStream("p2p.tr"));
  csma.EnableAsciiAll(ascii.CreateFileStream("csma.tr"));

  Simulator::Run ();
```

```
  Simulator::Destroy ();
  return 0;
}
```



## *4. fifth.cc (changes to second.cc)*

```
#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/csma-module.h"

#include "ns3/internet-module.h"

#include "ns3/internet-apps-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/applications-module.h"

#include "ns3/ipv4-global-routing-helper.h"

#include "ns3/netanim-module.h"

// Default Network Topology

//

// 10.1.1.0

// n0 -------------- n1 n2 n3 n4

// point-to-point | | | |
// ================
// LAN 10.1.2.0
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
int
main (int argc, char *argv[])
{
bool verbose = true;
uint32_t nCsma = 3;
CommandLine cmd (__FILE__);
cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
cmd.Parse (argc,argv);
if (verbose)
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
std::string animFile = "fifth.xml";
nCsma = nCsma == 0 ? 1 : nCsma;
NodeContainer p2pNodes;
p2pNodes.Create (2);
NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
```

```
p2pDevices = pointToPoint.Install (p2pNodes);
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);
InternetStackHelper stack;
stack.Install (p2pNodes.Get (0));
stack.Install (csmaNodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
pointToPoint.EnablePcapAll ("second");
```

```
csma.EnablePcap ("second", csmaDevices.Get (1), true);
V4PingHelper ping = V4PingHelper(csmaInterfaces.GetAddress(2));
NodeContainer pingers;
pingers.Add(csmaNodes.Get(0));
pingers.Add(csmaNodes.Get(1));
ApplicationContainer apps = ping.Install(pingers);
apps.Start(Seconds(2.0));
apps.Stop(Seconds(3.0));
csma.EnablePcapAll("csma-ping", true);
AnimationInterface anim(animFile);
anim.SetConstantPosition(csmaNodes.Get(0), 20.0, 100.0);
anim.SetConstantPosition(csmaNodes.Get(1), 20.0, 60.0);
anim.SetConstantPosition(csmaNodes.Get(2), 55.0, 30.0);
Simulator::Run ();
Simulator::Destroy ();
Return 0;
}
```