CN LAB – 2 INTERNALS

1. Create a simple topology of four nodes (Node1, Node2, Node3, Node4) separated by a point-to-point link. Setup a UdpClient on Node1 and a UdpServer on Node2. Let the data rate be set 5 Mbps and the delay be 2 ms. Now Setup another UdpClient on Node3 & Node4 and a server instance on Node2. Let the data rate be set 7 Mbps and the delay be 1 ms. Set the parameters for the clients. Demonstrate the usage of Flow monitor for the simulation by calculating throughput between Node 1 and Node 2.

CODE:

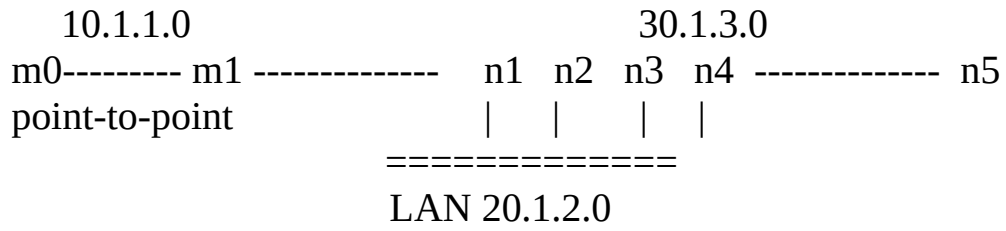
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/csma-module.h"
#include"ns3/flow-monitor-module.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("Second script example");

int main(int argc, char *argv[])
{
LogComponentEnable("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication",LOG_LEVEL_INFO);


NodeContainer p2pNodes;
p2pNodes.Create(4);


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

PointToPointHelper pointToPoint1;
pointToPoint1.SetDeviceAttribute("DataRate",StringValue("7Mbps"));
pointToPoint1.SetChannelAttribute("Delay",StringValue("1ms"));
```

```
PointToPointHelper pointToPoint2;
pointToPoint2.SetDeviceAttribute("DataRate",StringValue("7Mbps"));
pointToPoint2.SetChannelAttribute("Delay",StringValue("1ms"));


NetDeviceContainer p2pDevices;
p2pDevices=pointToPoint.Install(p2pNodes.Get(0),p2pNodes.Get(1));
std::cout<<"installed no and n1"<<std::endl;

NetDeviceContainer p2pDevices1;
p2pDevices1=pointToPoint1.Install(p2pNodes.Get(1),p2pNodes.Get(2));
std::cout<<"installed n1 and n2"<<std::endl;

NetDeviceContainer p2pDevices2;
p2pDevices2=pointToPoint2.Install(p2pNodes.Get(2),p2pNodes.Get(3));
std::cout<<"installed n2 and n3"<<std::endl;

InternetStackHelper stack;
stack.Install(p2pNodes);


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


Ipv4AddressHelper address1;
address1.SetBase("20.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces1;
p2pInterfaces1=address1.Assign(p2pDevices1);


address1.SetBase("20.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces2;
p2pInterfaces2=address1.Assign(p2pDevices2);


UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps = echoServer.Install(p2pNodes.Get(1));
serverApps.Start(Seconds(1.0));
```

```
serverApps.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient(p2pInterfaces.GetAddress(1),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));

UdpEchoServerHelper echoServer1(10);
ApplicationContainer serverApps1 = echoServer1.Install(p2pNodes.Get(1));
serverApps1.Start(Seconds(1.0));
serverApps1.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient1(p2pInterfaces1.GetAddress(0),10);
echoClient1.SetAttribute("MaxPackets", UintegerValue(1));
echoClient1.SetAttribute("Interval",TimeValue(Seconds(1.0)));
echoClient1.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps1=echoClient1.Install(p2pNodes.Get(2));
clientApps1.Start(Seconds(3.0));
clientApps1.Stop(Seconds(10.0));

UdpEchoServerHelper echoServer2(11);
ApplicationContainer serverApps2 = echoServer2.Install(p2pNodes.Get(1));
serverApps2.Start(Seconds(1.0));
serverApps2.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient2(p2pInterfaces1.GetAddress(0),11);
echoClient2.SetAttribute("MaxPackets", UintegerValue(1));
echoClient2.SetAttribute("Interval",TimeValue(Seconds(1.0)));
echoClient2.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps2=echoClient2.Install(p2pNodes.Get(3));
clientApps2.Start(Seconds(4.0));
clientApps2.Stop(Seconds(10.0));



FlowMonitorHelper flowmon;
```

```
Ptr<FlowMonitor>monitor=flowmon.InstallAll();

Ipv4GlobalRoutingHelper::PopulateRoutingTables();
NS_LOG_INFO("Run Simulation");
Simulator::Stop(Seconds(11.0));

Simulator::Run ();

monitor->CheckForLostPackets();
Ptr<Ipv4FlowClassifier>classifier=DynamicCast<Ipv4FlowClassifier>(flowmon.GetCla
ssifier());

std::map<FlowId,FlowMonitor::FlowStats>stats=monitor->GetFlowStats();

for(std::map<FlowId,FlowMonitor::FlowStats>::const_iterator i=stats.begin();i!
=stats.end();++i)
{
Ipv4FlowClassifier::FiveTuple t=classifier->FindFlow(i->first);
std::cout<<"Flow:"<<i->first<<"\nSourceAddress="<<t.sourceAddress<<"
DestinationAddress="<<t.destinationAddress<<" Source Port:"<<t.sourcePort<<" "<<"
Destination Port:"<<t.destinationPort<<"\n";
std::cout<<"Flow"<<i->first<<"("<<t.sourceAddress<<"->"<<t.destinationAddress<<")\
n";

std::cout<<"TxBytes:"<<i->second.txBytes<<"\n";
std::cout<<"RxBytes:"<<i->second.rxBytes<<"\n";
std::cout<<"TxPackets:"<<i->second.txPackets<<"\n";
std::cout<<"RxPackets:"<<i->second.rxPackets<<"\n";
std::cout<<"Total time taken for the transmission"<<i-
>second.timeLastRxPacket.GetSeconds()-i-
>second.timeFirstTxPacket.GetSeconds()<<std::endl;

std::cout<<"Throughput:"<<i->second.rxBytes*8.0/(i-
>second.timeLastRxPacket.GetSeconds()-i->second.timeFirstTxPacket.GetSeconds())/
1000/1000<<"mbps\n";
}

Simulator::Destroy();
NS_LOG_INFO("Done");

return 0;
}
```

2. Create a topology as given below



Choose m0 as client and n5 as server. Animate the simulation using NetAnim and visualize the packet flow.

CODE:

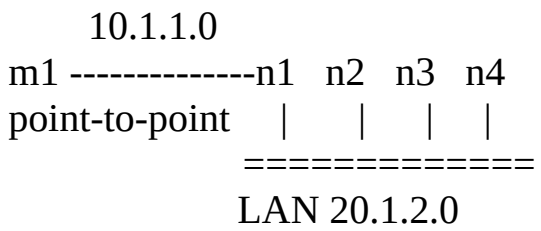
```
#include"ns3/core-module.h"
#include"ns3/network-module.h"
#include"ns3/csma-module.h"
#include"ns3/internet-module.h"
#include"ns3/point-to-point-module.h"
#include"ns3/applications-module.h"
#include"ns3/ipv4-global-routing-helper.h"
#include"ns3/netanim-module.h"
#include"ns3/mobility-module.h"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


using namespace ns3;
NS_LOG_COMPONENT_DEFINE("Second script example");

int main(int argc, char *argv[])
{

uint32_t nCsma=2;


LogComponentEnable("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication",LOG_LEVEL_INFO);


NodeContainer p2pNodes;
```

```
p2pNodes.Create(3);
NodeContainer p2pNodes1;
p2pNodes1.Create(2);


NodeContainer csmaNodes;
csmaNodes.Add(p2pNodes.Get(2));
csmaNodes.Create(nCsma);
csmaNodes.Add(p2pNodes1.Get(0));


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


PointToPointHelper pointToPoint1;
pointToPoint1.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
pointToPoint1.SetChannelAttribute("Delay",StringValue("2ms"));



PointToPointHelper pointToPoint2;
pointToPoint2.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
pointToPoint2.SetChannelAttribute("Delay",StringValue("2ms"));



PointToPointHelper pointToPoint3;
pointToPoint3.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
pointToPoint3.SetChannelAttribute("Delay",StringValue("2ms"));



CsmaHelper csma;
csma.SetChannelAttribute("DataRate",StringValue("10Mbps"));
csma.SetChannelAttribute("Delay",TimeValue(NanoSeconds(6560)));



NetDeviceContainer p2pDevices;
p2pDevices=pointToPoint.Install(p2pNodes.Get(0),p2pNodes.Get(1));
```

```
std::cout<<"installed no and n1"<<std::endl;
NetDeviceContainer p2pDevices1;
p2pDevices1=pointToPoint1.Install(p2pNodes.Get(1),p2pNodes.Get(2));
std::cout<<"installed n1 and n2"<<std::endl;
NetDeviceContainer p2pDevices2;
p2pDevices2=pointToPoint2.Install(p2pNodes1.Get(0),p2pNodes1.Get(1));
std::cout<<"installed n4 and n5"<<std::endl;
NetDeviceContainer csmaDevices;
csmaDevices=csma.Install(csmaNodes);


InternetStackHelper stack;
stack.Install(p2pNodes);
stack.Install(csmaNodes.Get(1));
stack.Install(csmaNodes.Get(2));
stack.Install(p2pNodes1);

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 p2pInterfaces1;
p2pInterfaces1=address.Assign(p2pDevices1);


Ipv4AddressHelper address1;
address1.SetBase("30.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces2;
p2pInterfaces2=address1.Assign(p2pDevices2);


Ipv4AddressHelper address3;
address3.SetBase("20.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces=address3.Assign(csmaDevices);



UdpEchoServerHelper echoServer(9);
```

```
ApplicationContainer serverApps = echoServer.Install(p2pNodes1.Get(1));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));



UdpEchoClientHelper echoClient(p2pInterfaces2.GetAddress(1),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("newAnim.xml");
anim.SetConstantPosition(p2pNodes.Get(0),5,5);
anim.SetConstantPosition(csmaNodes.Get(0),10,10);
anim.SetConstantPosition(csmaNodes.Get(1),15,15);
anim.SetConstantPosition(csmaNodes.Get(2),20,20);
anim.SetConstantPosition(csmaNodes.Get(3),25,25);
anim.SetConstantPosition(p2pNodes1.Get(1),30,30);
anim.EnablePacketMetadata(true);

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

3. Create a topology as given below



Choose m1 as client and n3 as server. Demonstrate the usage of flow monitor for the simulation by calculating Throughput.

CODE:

```
#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/ipv4-global-routing-helper.h"
#include "ns3/csma-module.h"
#include"ns3/flow-monitor-module.h"


using namespace ns3;

NS_LOG_COMPONENT_DEFINE("Second script example");

int main(int argc, char *argv[])
{

uint32_t nCsma=3;


LogComponentEnable("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication",LOG_LEVEL_INFO);
```

```
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"));


CsmaHelper csma;
csma.SetChannelAttribute("DataRate",StringValue("10Mbps"));
csma.SetChannelAttribute("Delay",TimeValue(NanoSeconds(6560)));


NetDeviceContainer p2pDevices;
p2pDevices=pointToPoint.Install(p2pNodes.Get(0),p2pNodes.Get(1));
std::cout<<"installed no and n1"<<std::endl;


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


InternetStackHelper stack;
stack.Install(p2pNodes);
stack.Install(csmaNodes.Get(1));
stack.Install(csmaNodes.Get(2));
stack.Install(csmaNodes.Get(3));
```

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


Ipv4AddressHelper address3;
address3.SetBase("20.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces=address3.Assign(csmaDevices);


UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(2));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));


UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(2),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();


FlowMonitorHelper flowmon;
Ptr<FlowMonitor>monitor=flowmon.InstallAll();
```

```
NS_LOG_INFO("Run Simulation");
Simulator::Stop(Seconds(11.0));
Simulator::Run ();

monitor->CheckForLostPackets();
Ptr<Ipv4FlowClassifier>classifier=DynamicCast<Ipv4FlowClassifier>(flowmon.
GetClassifier());

std::map<FlowId,FlowMonitor::FlowStats>stats=monitor->GetFlowStats();

for(std::map<FlowId,FlowMonitor::FlowStats>::const_iterator i=stats.begin();i!
=stats.end();++i)
{
Ipv4FlowClassifier::FiveTuple t=classifier->FindFlow(i->first);
std::cout<<"Flow:"<<i->first<<"\nSourceAddress="<<t.sourceAddress<<"
DestinationAddress="<<t.destinationAddress<<" Source Port:"<<t.sourcePort<<"
"<<" Destination Port:"<<t.destinationPort<<"\n";
std::cout<<"Flow"<<i->first<<"("<<t.sourceAddress<<"-
>"<<t.destinationAddress<<")\n";

std::cout<<"TxBytes:"<<i->second.txBytes<<"\n";
std::cout<<"RxBytes:"<<i->second.rxBytes<<"\n";
std::cout<<"TxPackets:"<<i->second.txPackets<<"\n";
std::cout<<"RxPackets:"<<i->second.rxPackets<<"\n";
std::cout<<"Total time taken for the transmission"<<i-
>second.timeLastRxPacket.GetSeconds()-i-
>second.timeFirstTxPacket.GetSeconds()<<std::endl;

std::cout<<"Throughput:"<<i->second.rxBytes*8.0/(i-
>second.timeLastRxPacket.GetSeconds()-i-
>second.timeFirstTxPacket.GetSeconds())/1000/1000<<"mbps\n";
}

Simulator::Destroy();
NS_LOG_INFO("Done");

return 0;
}
```

4.Create a simple topology of four nodes (Node1, Node2, Node3, Node4) separated by a point-to-point link. Setup a UdpClient on Node1 and Node2. Let the data rate be set 5 Mbps and the delay be 2 ms. Now Setup another UdpClient on Node3 and a server instance on Node4. Let the data rate be set 7 Mbps and the delay be 1 ms. Set the parameters for the clients.Animate the simulation using NetAnim and visualize the packet flow.

CODE:

```
#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/ipv4-global-routing-helper.h"
#include "ns3/csma-module.h"
#include"ns3/flow-monitor-module.h"
#include"ns3/netanim-module.h"
#include"ns3/mobility-module.h"
#include"ns3/animation-interface.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("Second script example");

int main(int argc, char *argv[])
{
LogComponentEnable("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication",LOG_LEVEL_INFO);


NodeContainer p2pNodes;
p2pNodes.Create(4);



PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
```

```
pointToPoint.SetChannelAttribute("Delay",StringValue("2ms"));

PointToPointHelper pointToPoint1;
pointToPoint1.SetDeviceAttribute("DataRate",StringValue("5Mbps"));
pointToPoint1.SetChannelAttribute("Delay",StringValue("2ms"));

PointToPointHelper pointToPoint2;
pointToPoint2.SetDeviceAttribute("DataRate",StringValue("7Mbps"));
pointToPoint2.SetChannelAttribute("Delay",StringValue("1ms"));


NetDeviceContainer p2pDevices;
p2pDevices=pointToPoint.Install(p2pNodes.Get(0),p2pNodes.Get(1));
std::cout<<"installed no and n1"<<std::endl;

NetDeviceContainer p2pDevices1;
p2pDevices1=pointToPoint1.Install(p2pNodes.Get(1),p2pNodes.Get(2));
std::cout<<"installed n1 and n2"<<std::endl;

NetDeviceContainer p2pDevices2;
p2pDevices2=pointToPoint2.Install(p2pNodes.Get(2),p2pNodes.Get(3));
std::cout<<"installed n2 and n3"<<std::endl;



InternetStackHelper stack;
stack.Install(p2pNodes);


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


Ipv4AddressHelper address1;
address1.SetBase("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces1;
```

```
p2pInterfaces1=address1.Assign(p2pDevices1);


address1.SetBase("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces2;
p2pInterfaces2=address1.Assign(p2pDevices2);


UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps = echoServer.Install(p2pNodes.Get(3));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));


UdpEchoClientHelper echoClient(p2pInterfaces2.GetAddress(1),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));

UdpEchoServerHelper echoServer1(10);
ApplicationContainer serverApps1 = echoServer1.Install(p2pNodes.Get(3));
serverApps1.Start(Seconds(1.0));
serverApps1.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient1(p2pInterfaces2.GetAddress(1),10);
echoClient1.SetAttribute("MaxPackets", UintegerValue(1));
echoClient1.SetAttribute("Interval",TimeValue(Seconds(1.0)));
echoClient1.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps1=echoClient1.Install(p2pNodes.Get(1));
clientApps1.Start(Seconds(2.0));
clientApps1.Stop(Seconds(10.0));

UdpEchoServerHelper echoServer2(11);
```

```
ApplicationContainer serverApps2 = echoServer2.Install(p2pNodes.Get(3));
serverApps2.Start(Seconds(1.0));
serverApps2.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient2(p2pInterfaces2.GetAddress(1),11);
echoClient2.SetAttribute("MaxPackets", UintegerValue(1));
echoClient2.SetAttribute("Interval",TimeValue(Seconds(1.0)));
echoClient2.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps2=echoClient2.Install(p2pNodes.Get(2));
clientApps2.Start(Seconds(2.0));
clientApps2.Stop(Seconds(10.0));


Ipv4GlobalRoutingHelper::PopulateRoutingTables();
NS_LOG_INFO("Run Simulation");
Simulator::Stop(Seconds(11.0));

AnimationInterface anim("test2.xml");
anim.SetConstantPosition(p2pNodes.Get(0),10,10);
anim.SetConstantPosition(p2pNodes.Get(1),20,20);
anim.SetConstantPosition(p2pNodes.Get(2),30,30);
anim.SetConstantPosition(p2pNodes.Get(3),40,40);
anim.EnablePacketMetadata(true);

Simulator::Run ();
Simulator::Destroy();
NS_LOG_INFO("Done");

return 0;
}
```

5. Using TCP/IP sockets, write a client-server program to make client sending the file name and a word and the server to search for the presence of the word in the file and return the number of times the word being repeated in the file contents back to client.

CODE:

client.py

```
from socket import*
serverName=gethostname()
serverPort=12000
clientSocket=socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
file1=input("Enter file name and the word : ")
clientSocket.send(file1.encode())
wordcount=clientSocket.recv(1024).decode('utf8')
print(wordcount)
clientSocket.close()
```

server.py

```
from socket import*
serverName=gethostname()
serverPort=12000
serverSocket=socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
print("The Server is ready to recieve")
while(1):
    count=0
    conn,addr=serverSocket.accept()
    file2=conn.recv(1024).decode()
    list1=[]
    list1=file2.split(" ")
    file=open(list1[0],"r")
    words=file.read(1024)
    list2=[]
    list2=words.split(" ")
    for i in list2:
        if list1[1]==i or list1[i] +'\n'==i:
            count+=1
    print(count)
    conn.send(str(count).encode('utf8'))
    file.close()
conn.close()
```

6.Using UDP sockets, write a client-server program to make client send a word and the server to search for the presence of the word in a list of 5 files and return the filename/s in which the word is present back to client.

CODE:

Client

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET,SOCK_DGRAM)
sentence = input("Enter a word : ")
clientSocket.sendto(bytes(sentence,"utf-8"),(serverName,serverPort))
filecontents,aerverAddress = clientSocket.recvfrom(2048)
print("from server : ",filecontents.decode('utf-8'))
clientSocket.close()
```

Server

```
from socket import *
st = ""
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_DGRAM)
serverSocket.bind(("127.0.0.1",serverPort))
files = ['file1.txt','file2.txt','file3.txt','file4.txt','file5.txt']
print("server is ready to receive")
while 1:
        sentence,clientAddress = serverSocket.recvfrom(2048)
        for i in files:
                fil = open(i,"r")
                y = fil.read().replace("."," ").replace(","," ")
                x = y.strip().split()
                p = sentence.decode('utf-8')
                if p in x:
                        st = st+str(i)+" "
                fil.close()
        if(len(st)!=0):
                serverSocket.sendto(bytes(st,"utf-8"),clientAddress)
        else:
                serverSocket.sendto(bytes("Not found","utf-8"),clientAddress)
        print("sent")
        break
```

7. Implement the client server program using IPC through FIFO wherein the client sends a series of sentences to the server. The server writes all these sentences to a file and returns the contents of the file back to client.

CODE:
client.c

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666
char fname[512];
int main( )
 {
        int readfd, writefd, fd;
        ssize_t n;
        char buff[512];
        char end[5]="end";
        int cmp = 0;
        int le;
        char st[80];
        writefd=open(FIFO1, O_WRONLY, 0);
                readfd =open(FIFO2, O_RDONLY, 0);
        printf("Connected..\n");
        while(1)
        {
                printf("Enter thr string " );
                fgets(st,sizeof(st),stdin);
                le = strlen(st);
                st[le-1]='\0';
                cmp = strcmp(st,"end");
                if(cmp==0)
                {
                        write(writefd,st,strlen(st)+1);
```

```
                printf("end");
                break;
            }
            else
            {
                write(writefd,st,strlen(st)+1);
                printf("%s\n",st);
            }
        }
    printf("Waiting for reply from server ......\n");
    while((n=read(readfd,buff,512))>0)
    {
       write(1,buff,n);
    }
        printf("\n");
        close(readfd); unlink(FIFO1);
        close(writefd); unlink(FIFO2);
}
```

server.c

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666
char fname[80];
int sam = 0;
char fil[20]="hello.txt";
int main( )
 {
        int readfd, writefd, fd;
        ssize_t n;
        char buff[512];
        char end[10];
        int siz = 0;
        FILE *f = fopen("hello.txt","w");
        fclose(f);
        f = fopen("hello.txt","a");
```

```
        if (mkfifo(FIFO1, PERMS)<0)
                printf("Cant Create FIFO Files\n");
        if (mkfifo(FIFO2, PERMS)<0)
                 printf("Cant Create FIFO Files\n");
        printf("Waiting for connection Request..\n");
        readfd =open(FIFO1, O_RDONLY, 0);
        writefd=open(FIFO2, O_WRONLY, 0);
        printf("Connection Established..\n");
        strcpy(end,"end");
        while(1)
        {
                siz = read(readfd, fname,sizeof(fname));
                fname[siz]='\0';
                sam = strcmp(fname,end);
                if(sam!=0)
                {
                        printf("received = %s\n",fname);
                        fprintf(f,"%s ",fname);
                }
                else
                {
                        fclose(f);
                        break;
                }

        }
        if ((fd=open(fil,O_RDWR))<0)
        {
                 strcpy(buff,"File does not exist..\n");
                 write(writefd, buff, strlen(buff));
        }
        else
        {
                while((n=read(fd, buff,512))>0)
                {
                 printf("reading");
                 write(writefd, buff, n);
                }
        }
         close(readfd); unlink(FIFO1);
         close(writefd); unlink(FIFO2);
}
```

8. Implement the leaky bucket algorithm for the following data.
Bucket Size: 256Mbits
Output Rate: 50Mbps
Number of packets: 10
Packet Size in Mbits: 100, 345, 230,78, 980, 130, 7, 89, 670, 256
The interarrival time between each packet is 5s.

CODE:

```
#include<iostream>
#include<unistd.h>
#include<stdlib.h>
using namespace std;
#define bucketsize 256
void bktinput(int a, int b);
int main()
{
	int op,pktsize,pktsize[]={100, 345, 230,78, 980, 130, 7, 89, 670, 256};
	cout<<"Enter output rate:";
	cin>>op;    //50(Mbps)
	for(int i=0;i<10;i++)
	{
		usleep(rand()%1000);
		cout<<"\nPacket no"<<i+1<<"\tPacket size="<<pktsize[i];
		bktinput(pktsize[i],op);
	}
return 0;
}
void bktinput(int a,int b)
{
	if(a>bucketsize)
		cout<<"\n\t\tBucket overflow";
	else
	{
		usleep(500);
		while(a>b)
		{
			cout<<"\n\t\t"<<b<<"bytes outputted";
```

```
                a-=b;
                usleep(500);
            }
        if(a>0)
            cout<<"\n\t\tlast"<<a<<"bytes sent\t";
        cout<<"\n\t\t Bucket output successful";
        }
}
```

9. Implement Dijkstra's algorithm to compute the shortest path through a graph.

CODE:

```
#include<iostream>
#include<string.h>
#include<math.h>
#define IN 99
#define N 5
using namespace std;

int dijkstra(int cost[][N],int source,int target)
{
        int dist[N],prev[N],selected[N]={0},i,m,min,start,d,j;
        char path[N];
        for(i=1;i<N;i++)
        {
            dist[i]=IN;
            prev[i]=-1;
        }
        start=source;
        selected[start]=1;
        dist[start]=0;
        while(selected[target]==0)
        {
            min=IN;
            m=0;
            for(i=1;i< N;i++)
            {
                d = dist[start] +cost[start][i];
```

```
                if(d<dist[i]&&selected[i]==0)
                {
                        dist[i] = d;
                        prev[i] = start;
                }
                if(min>dist[i] && selected[i]==0)
                {
                        min = dist[i];
                        m = i;
                }
        }
        start = m;
        selected[start] = 1;
    }
    start = target;
    j = 0;
    while(start != -1)
    {
        path[j++] = start+64;
        start = prev[start];
    }
    path[j]='\0';
    for(i=strlen(path)-1;i>0;i--)
        cout<<path[i]<<"--->";
    cout<<path[0];
    return dist[target];
}
int main()
{
    int cost[N][N],i,j,w,ch,co;
    int source, target,x,y;
    cout<<"\tShortest Path Algorithm using Dijkstra's Algorithm\n\n";
    for(i=1;i< N;i++)
        for(j=1;j< N;j++)
            cost[i][j] = IN;
    for(x=1;x< N;x++)
    {
        for(y=x+1;y< N;y++)
        {
```

```
cout<<"Enter the weight of the path between node "<<char(x+64)<<" -->
"<<char(y+64)<<":";
                        cin>>w;
                        cost [x][y] = cost[y][x] = w;
                }
                cout<<"\n";
        }
        cout<<"\nEnter The Source:";
        cin>>source;
        cout<<"\nEnter The target:";
        cin>>target;
        co = dijkstra(cost,source,target);
        cout<<"\nShortest Path is "<<co<<endl;
}

/*
OUTPUT:
Shortest Path Algorithm using Dijkstra's Algorithm

Enter the weight of the path between node A --> B:8
Enter the weight of the path between node A --> C:2
Enter the weight of the path between node A --> D:15

Enter the weight of the path between node B --> C:12
Enter the weight of the path between node B --> D:6

Enter the weight of the path between node C --> D:11


Enter The Source:1

Enter The target:4
A--->C--->D
Shortest Path is 13
*/
```

10. Write a program for distance vector algorithm to find suitable path for transmission.

CODE:

```
#include<stdio.h>
struct node
{
   int dist[20];
```

```
    int from[20];

}rt[10];

int main()
{
    int costmat[20][20];
    int nodes,i,j,k,count=0;
   // clrscr();
    printf("\nEnter the number of nodes : ");
    scanf("%d",&nodes);
    printf("\nEnter the cost matrix :\n");
    for(i=0;i<nodes;i++)
    {
        for(j=0;j<nodes;j++)
        {
            scanf("%d",&costmat[i][j]);
            costmat[i][i]=0;
            rt[i].dist[j]=costmat[i][j];
            rt[i].from[j]=j;
        }
    }
        do
        {
            count=0;
            for(int i=0;i<nodes;i++)
            for(int j=0;j<nodes;j++)
            for(int k=0;k<nodes;k++)
                if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
                {
                    rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
                    rt[i].from[j]=k;
                    count++;
                }
        }while(count!=0);

        for(int i=0;i<nodes;i++)
        {
            printf("\n For router %d \n",i);
```

```
            for(int j=0;j<nodes;j++)
            {
                printf("\n node %d via %d Distance %d ",j,rt[i].from[j],rt[i].dist[j]);
            }
        }
    printf("\n\n");
    //getc();
    return 0;
}
/*
OUTPUT:
Enter the number of nodes : 4

Enter the cost matrix :
0 2 7 3
2 0 1 4
7 1 0 6
3 4 6 0

For router 0

node 0 via 0 Distance 0
node 1 via 1 Distance 2
node 2 via 1 Distance 3
node 3 via 3 Distance 3
For router 1

node 0 via 0 Distance 2
node 1 via 1 Distance 0
node 2 via 2 Distance 1
node 3 via 3 Distance 4
For router 2

node 0 via 1 Distance 3
node 1 via 1 Distance 1
node 2 via 2 Distance 0
node 3 via 1 Distance 5
For router 3

node 0 via 0 Distance 3
node 1 via 1 Distance 4
node 2 via 1 Distance 5
node 3 via 3 Distance 0
*/
```