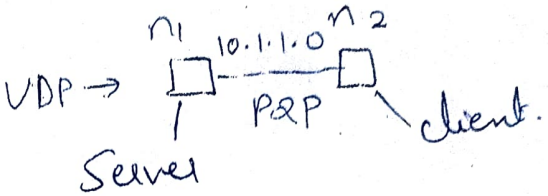
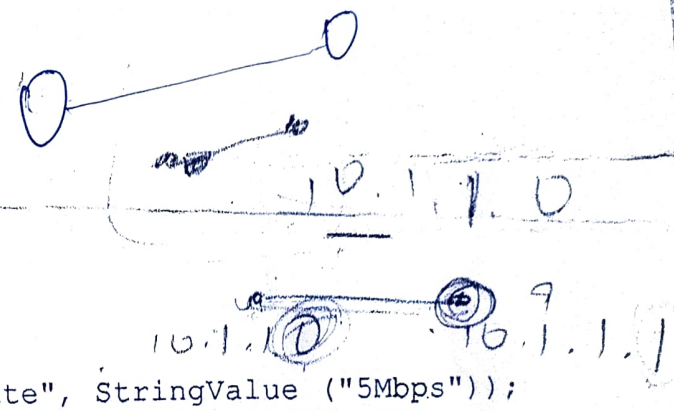
Create UDP echo client & Server application on P2P connection.

Program-1

AKshatha.G.C

```
/*
 * 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/netanim-module.h"
#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"

using namespace ns3;
int
main (int argc, char *argv[])
{
  Time::SetResolution (Time::NS);
  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));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
```





Create Nodes

Create Link

Join link to node.

Install TCP/IP protocol stack

set base addr

Assign addr

IP addr Subnet mask

point-to-point connection is a dedicated comm. link b/w 2 S/m's @ protocol

Application starts

UdpEchoServerHelper echoServer (9); — Start Server at port 9.

— Install Server on node 1

Start & Stop the Server.

These are 6 steps to be followed.

Creating topology

1. Create Node
- End system.
- Create an object for the class NodeContainer
- Through the object call Create function with #nodes a parameter.

NIC → gives the MAC addr & is unique to every system (DL).
NIC / N/w Interface card

2. Create Link
- Decide the type of link (P2P, CSMA).
- Create an object for the class of the link. (PointToPointHelper, CsmaHelper)
- Through the object call the functions - SetDeviceAttr to set data rate & SetChannelAttribute to set delay

3. Join link to the node.
- Create an object to the calss NetDeviceContainer
- Call Through the link object Call It covers NIC & the function Install with node object Device drivers s/w as Parameter & assign it to the NetDevice object

Device driver s/w is a prog which run to understand what type of device connected & how it sends & receives data & at what rate.

4. TCP/IP Stack.
- Create an object for the class InternetStackHelper
- Through the object call Install function passing node object as parameter.

5. Give IP address [N/w base addr, Subnet mask]
- Decide type of IP addr - IPv4 or IPv6 is used.
- Create an object for the class IPv4AddressHelper.
- Through the object call SetBase function with N/w base addr & subnet mask as parameters
- Through the object call the function Assign with NetDevice object as parameter & assign it to the object created for Ipv4InterfaceContainer class.

6. Application.
- Decide the port number from unreserved numbers.
- Start the UDP echo server at the specified port.
- Create an server object for the class UDPEchoServerHelper to specify the port number.
- ~~Create an object for ApplicationContainer~~
- Through the object call Install passing the node number as parameter & assign it to the server object of ApplicationContainer.

Can be obtained from nodes.Get function.

The object of AC Start & stop the server calling respe

```
  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));
 AnimationInterface anim ("first.xml");
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

- Start the UDP echo client & it should access the server installed in node 1 & port 9.
  - Create an object for UdpEchoClientHelper with server addr & port number as parameter.
    - server addr: obtained using interfaces.GetAddress(1) ← node no.
    - interfaces: object of Ipv4InterfaceContainer.
  - Through the object call ~~SetAttribute function~~ & set
    - MaxPackets that can be sent
    - Interval b/w each packet to send.
    - packet size
  - Through the object call Install function & pass node number as parameter
    - node number: nodes.Get(0).
  - Create an client object for Application container class & assign it to
  - Start & stop the client.
- Step up is ready to run the simulator the animation is stored in first.xml

Program-2 Create UDP echo client server application on

```
/* -*- 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
 */

// Network topology  UDP echoserver
//
//       n0     n1   n2   n3
//       |3     |2   |1   |0   UDP
//       ================
//              LAN
//
// - UDP flows from n0 to n1 and back
// - DropTail queues
// - Tracing of queues and packet receptions to file "udp-echo.tr"

#include <fstream>
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"

using namespace ns3;

int
main (int argc, char *argv[])
{
  Address serverAddress;
  NodeContainer n;
  n.Create (4);
  InternetStackHelper internet;
  internet.Install (n);
  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate
(5000000)));
  csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
  csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));
  NetDeviceContainer d = csma.Install (n);
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (d);
  serverAddress = Address(i.GetAddress (1));
```

CSMA
Carrier sense multiple access.

```
uint16_t port = 9;  // well-known echo port number
UdpEchoServerHelper server (port);
ApplicationContainer apps = server.Install (n.Get (1));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));
```

} server App.

```
uint32_t packetSize = 1024;
uint32_t maxPacketCount = 1;
Time interPacketInterval = Seconds (1.);
UdpEchoClientHelper client (serverAddress, port);
client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
client.SetAttribute ("Interval", TimeValue (interPacketInterval));
client.SetAttribute ("PacketSize", UintegerValue (packetSize));
apps = client.Install (n.Get (0));
apps.Start (Seconds (2.0));
apps.Stop (Seconds (10.0));
```

} client App.

addr of client node, data to be sent

```
#if 0
client.SetFill (apps.Get (0), "Hello World");
client.SetFill (apps.Get (0), 0xa5, 1024);
uint8_t fill[] = { 0, 1, 2, 3, 4, 5, 6};
  client.SetFill (apps.Get (0), fill, sizeof(fill), 1024);
#endif
AnimationInterface anim ("second.xml");
  Simulator::Run ();
  Simulator::Destroy ();
}
```

packet size is auto adjusted to reflect the size of the string

fill, fillsize, datasize.

- Set the data fill of the packet to the contents of the fill buffer, repeated as many times as required.
- ⊛
- fillsize – The #byts in the provided fill pattern.

⊛ Packet size is auto adjusted to reflect dataSize Parameter specified next

desired size of the resulting echo packet.

Program-3

& Server application combination of P2P CSMA Connection

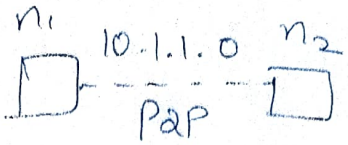
```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"


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

UDP

```
using namespace ns3;
int
main (int argc, char *argv[])
{

  uint32_t nCsma = 3;
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 ("third.xml");
Simulator::Run ();
  Simulator::Destroy ();
  return 0;
```

Program - 4 Create TCP Source & Sink application on P2P connection.




```
#include <string>
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/network-module.h"
#include "ns3/packet-sink.h"
#include "ns3/netanim-module.h"

using namespace ns3;
int
main (int argc, char *argv[])
{

  uint32_t maxBytes = 0;
  NodeContainer nodes;
  nodes.Create (2);
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue
("500Kbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));
  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);
  InternetStackHelper internet;
  internet.Install (nodes);
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (devices);
  uint16_t port = 9;  // well-known echo port number
  BulkSendHelper source ("ns3::TcpSocketFactory",
  InetSocketAddress (i.GetAddress (1), port));
   source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));
   ApplicationContainer sourceApps = source.Install (nodes.Get (0));
   sourceApps.Start (Seconds (0.0));
   sourceApps.Stop (Seconds (10.0));
  PacketSinkHelper sink ("ns3::TcpSocketFactory",
    InetSocketAddress (Ipv4Address::GetAny (), port));
    ApplicationContainer sinkApps = sink.Install (nodes.Get (1));
    sinkApps.Start (Seconds (0.0));
    sinkApps.Stop (Seconds (10.0));
  Simulator::Stop (Seconds (10.0));
  AnimationInterface anim ("fourth.xml");
  anim.EnablePacketMetadata(true);
   Simulator::Run ();
    Simulator::Destroy ();


  }
```

SYN

S=0 A=0 → S

S=0, A=1 ←

S=1 A=1 →

S=1, A=513