First Example

```
#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"

// 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);

  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);
  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;
}
```

Full Code Explanation

1. Including NS-3 Modules

```
#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"
```

These headers give access to the classes required:

- **core-module** provides basic NS-3 functions like time, logging and events.
- **network-module** handles nodes, devices and channels.
- **internet-module** installs protocols like IP, UDP, TCP.
- **point-to-point-module** supports wired links.
- **applications-module** provides ready-made apps like UDP Echo Client/Server.

2. Logging Setup

```
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
```

This tags log messages so you can enable or disable them easily.

3. Main Function Starts

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

The simulation starts here.

4. Command Line Parsing

```
CommandLine cmd (__FILE__);
cmd.Parse (argc, argv);
```

This allows you to pass arguments when running the program.

5. Time Resolution and Logging

```
Time::SetResolution (Time::NS);
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

- Simulation runs in nanoseconds.
- Logging for client and server is turned on so you can see results.

6. Create Two Nodes

```
NodeContainer nodes;
nodes.Create (2);
```

This creates n0 and n1. Think of them as two computers.

7. Configure Point-to-Point Link

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

Specifies a wired link with 5 Mbps bandwidth and 2 ms propagation delay.

8. Install Devices on Nodes

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

The link is installed on both nodes. Each gets a network interface card (NIC).

9. Install Internet (TCP/IP Stack)

```
InternetStackHelper stack;
stack.Install (nodes);
```

This enables IPv4, routing, UDP, TCP.

10. Assign IP Addresses

```
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

The link is given the subnet **10.1.1.0/24**.
Each node gets an IP address.

11. Set Up UDP Echo Server on Node 1

```
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

- Port **9** is used.
- Server starts at **1 second** and stops at **10 seconds**.

12. Set Up UDP Echo Client on Node 0

```
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));
```

- Client sends **one 1024-byte packet** to server at node 1.
- Starts at **2 seconds**.
- Uses server's IP and port.

13. Run and Destroy Simulation

```
Simulator::Run ();
Simulator::Destroy ();
```

This starts the event scheduler and runs until all events finish.

Full Code With Visualisation using Netanim

```
#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"

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);

  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);
  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));

  // NetAnim Visualization
  AnimationInterface anim ("first.xml");
  anim.SetConstantPosition (nodes.Get (0), 10, 20);
  anim.SetConstantPosition (nodes.Get (1), 30, 20);

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

Star Topology with Netanim visualization

```
#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"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("StarTopologyWithNetAnim");

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

  uint32_t nLeaf = 4;

  Time::SetResolution (Time::NS);
  LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer hubNode;
  hubNode.Create(1);

  NodeContainer leafNodes;
  leafNodes.Create(nLeaf);

  InternetStackHelper stack;
  stack.Install(hubNode);
  stack.Install(leafNodes);

  PointToPointHelper p2p;
  p2p.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
  p2p.SetChannelAttribute("Delay", StringValue("2ms"));

  NetDeviceContainer devices[nLeaf];
  Ipv4InterfaceContainer interfaces;
  char subnet[20];

  for (uint32_t i = 0; i < nLeaf; i++)
  {
    NodeContainer pair(hubNode.Get(0), leafNodes.Get(i));
    devices[i] = p2p.Install(pair);

    sprintf(subnet, "10.1.%d.0", i + 1);
    Ipv4AddressHelper address;
    address.SetBase(subnet, "255.255.255.0");

    Ipv4InterfaceContainer iface = address.Assign(devices[i]);
    interfaces.Add(iface);
  }

UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApp = echoServer.Install(hubNode.Get(0));
serverApp.Start(Seconds(1.0));
serverApp.Stop(Seconds(20.0));

for (uint32_t i = 0; i < nLeaf; i++)
  {
    Address hubAddress = interfaces.Get(i * 2);

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

    ApplicationContainer clientApp =
      echoClient.Install(leafNodes.Get(i));

    clientApp.Start(Seconds(2.0 + i));
    clientApp.Stop(Seconds(20.0));
  }

// -----------------------------
// NetAnim Visualization Section
// -----------------------------
AnimationInterface anim("star-topology.xml");

// Set positions manually to form a star layout
anim.SetConstantPosition(hubNode.Get(0), 50, 50);

double radius = 30.0;
double angleStep = 360.0 / nLeaf;
```

```
  for (uint32_t i = 0; i < nLeaf; i++)
    {
      double angle = angleStep * i;
      double rad = angle * M_PI / 180.0;

      double x = 50 + radius * std::cos(rad);
      double y = 50 + radius * std::sin(rad);

      anim.SetConstantPosition(leafNodes.Get(i), x, y);
    }

  // Optional node descriptions and colors
  anim.UpdateNodeDescription(hubNode.Get(0), "Hub");
  anim.UpdateNodeColor(hubNode.Get(0), 255, 0, 0);

  for (uint32_t i = 0; i < nLeaf; i++)
    {
      std::string desc = "Leaf " + std::to_string(i + 1);
      anim.UpdateNodeDescription(leafNodes.Get(i), desc);
      anim.UpdateNodeColor(leafNodes.Get(i), 0, 0, 255);
    }

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

Bus Topology Code with NetAnim Visualization

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("BusTopologyWithNetAnim");

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);

  // Create 4 nodes on the bus
  NodeContainer nodes;
  nodes.Create(4);

  // CSMA models a shared bus cable
  CsmaHelper csma;
  csma.SetChannelAttribute("DataRate", StringValue("10Mbps"));
  csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));

  NetDeviceContainer devices = csma.Install(nodes);

  InternetStackHelper stack;
  stack.Install(nodes);

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

  Ipv4InterfaceContainer interfaces = address.Assign(devices);

  // Server on Node 0
  UdpEchoServerHelper echoServer(9);
  ApplicationContainer serverApp = echoServer.Install(nodes.Get(0));
  serverApp.Start(Seconds(1.0));
  serverApp.Stop(Seconds(10.0));

  // Clients on Node 1, 2, 3
  for (uint32_t i = 1; i < nodes.GetN(); i++)
    {
      UdpEchoClientHelper echoClient(interfaces.GetAddress(0), 9);
      echoClient.SetAttribute("MaxPackets", UintegerValue(1));
      echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
      echoClient.SetAttribute("PacketSize", UintegerValue(1024));

      ApplicationContainer clientApp = echoClient.Install(nodes.Get(i));
      clientApp.Start(Seconds(2.0 + i));
      clientApp.Stop(Seconds(10.0));
    }

  // ----------------------------
  // NetAnim Visualization Section
  // ----------------------------
  AnimationInterface anim("bus-topology.xml");

  // Arrange nodes in a horizontal bus layout
  int x_start = 10;
  int y_pos = 30;

  for (uint32_t i = 0; i < nodes.GetN(); i++)
    {
      anim.SetConstantPosition(nodes.Get(i), x_start + i * 20, y_pos);

      std::string desc = "Node " + std::to_string(i);
      anim.UpdateNodeDescription(nodes.Get(i), desc);

      // Server is red, clients are blue
      if (i == 0)
        anim.UpdateNodeColor(nodes.Get(i), 255, 0, 0);
      else
        anim.UpdateNodeColor(nodes.Get(i), 0, 0, 255);
    }

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