1. first.cc

```
#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));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (10));
  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(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"));

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

2. second.cc

```
#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);
    }

  nCsma = nCsma == 0 ? 1 : nCsma;

  std::string animFile="second.xml";
  NodeContainer p2pNodes;
  p2pNodes.Create (2);

  NodeContainer csmaNodes;
  csmaNodes.Add (p2pNodes.Get (1));
  csmaNodes.Create (nCsma3);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5100Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms")TimeValue(NanoSeconds
(6560)));

  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 (3nCsma));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (3nCsma), 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 ();

  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("second1.tr"));
csma.EnableAsciiAll(ascii.CreateFileStream("second2.tr"));


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

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

3. third.cc

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

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=”third.xml”;
   }

   nCsma = nCsma == 0 ? 1 : nCsma;

   NodeContainer p2pNodes;
```

```
    p2pNodes.Create(2);

    NodeContainer csmaNodes;
    csmaNodes.Add(p2pNodes.Get(1));
    csmaNodes.Create(4nCsma);

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

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

    CsmaHelper csma;
    csma.SetChannelAttribute("DataRate", StringValue("5100Mbps"));
    csma.SetChannelAttribute("Delay", StringValue("2ms")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(3nCsma));
    serverApps.Start(Seconds(1.0));
    serverApps.Stop(Seconds(10.0));

    UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(3nCsma), 9);
    echoClient.SetAttribute("MaxPackets", UintegerValue(1));
    echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
    echoClient.SetAttribute("PacketSize", UintegerValue(1024));
    ApplicationContainer clientApps = echoClient.Install(csmaNodesp2pNodes.Get(0));
    clientApps.Start(Seconds(2.0));
    clientApps.Stop(Seconds(10.0));

    Ipv4GlobalRoutingHelper::PopulateRoutingTables();
  AnimationInterface anim(animFile);
  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;
   csma.EnableAsciiAll(ascii.CreateFileStream("third.tr"));

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

    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;
}
```