Sudo usermod -aG sudo username

Groups username

Sudo visudo

Username ALL=(ALL:ALL) ALL

1st prg

```
#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);

  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));

    Ptr<Node> n0 = nodes.Get(0);
    Ptr<Node> n1 = nodes.Get(1);
```

```
    AnimationInterface anim("myfirst.xml");
    anim.SetConstantPosition(n0, 100, 400);
    anim.SetConstantPosition(n1, 400, 400);

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

2nd peer to peer

```
#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"
#include "ns3/flow-monitor-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("PeerToPeerExample");

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 a container for three nodes
  NodeContainer nodes;
  nodes.Create (3);

  // Setup point-to-point links for all nodes to connect to the server
(node 0)
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue
("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices1, devices2;
  devices1 = pointToPoint.Install (nodes.Get(0), nodes.Get(1)); // Link
between server and client 1
  devices2 = pointToPoint.Install (nodes.Get(0), nodes.Get(2)); // Link
between server and client 2
```

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

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

  Ipv4InterfaceContainer interfaces1 = address.Assign (devices1);
  Ipv4InterfaceContainer interfaces2 = address.Assign (devices2);

  // Create a UDP Echo Server on node 0
  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (nodes.Get
(0));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  // Create UDP Echo Clients on node 1 and node 2
  UdpEchoClientHelper echoClient1 (interfaces1.GetAddress (0), 9);
// Client 1
  echoClient1.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient1.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient1.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps1 = echoClient1.Install (nodes.Get
(1));
  clientApps1.Start (Seconds (2.0));
```

```
  clientApps1.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient2 (interfaces2.GetAddress (0), 9);
// Client 2
  echoClient2.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient2.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient2.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps2 = echoClient2.Install (nodes.Get
(2));
  clientApps2.Start (Seconds (3.0));
  clientApps2.Stop (Seconds (10.0));

  // Animation interface for visualization
  AnimationInterface anim("second.xml");
  anim.SetConstantPosition(nodes.Get(0), 100, 400); // Server
  anim.SetConstantPosition(nodes.Get(1), 400, 300); // Client 1
  anim.SetConstantPosition(nodes.Get(2), 400, 500); // Client 2

  // Optional: ASCII trace for packet metrics
  AsciiTraceHelper ascii;
  pointToPoint.EnableAsciiAll(ascii.CreateFileStream("trace.tr"));

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

```
  return 0;
}
```

3rd star topology

```
#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"
#include "ns3/flow-monitor-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE
("StarTopologyBroadcastExample");



void BroadcastPacket(Ptr<Socket> broadcastSocket) {
    Ptr<Packet> packet = Create<Packet>(1024);
    broadcastSocket->SendTo(packet, 0,
InetSocketAddress(Ipv4Address("255.255.255.255"), 9));
}
```

```
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 (8);

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

  NetDeviceContainer devices[7];
  for (int i = 0; i < 7; ++i) {
    devices[i] = pointToPoint.Install(nodes.Get(0), nodes.Get(i + 1));
```

```
    }

    InternetStackHelper stack;
    stack.Install(nodes);

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



    Ipv4InterfaceContainer interfaces[7];
    for (int i = 0; i < 7; ++i) {
        interfaces[i] = address.Assign(devices[i]);
    }



    Ptr<Socket> broadcastSocket =
Socket::CreateSocket(nodes.Get(0),
UdpSocketFactory::GetTypeId());
    broadcastSocket-
>Bind(InetSocketAddress(Ipv4Address::GetAny(), 9));
    broadcastSocket->SetAllowBroadcast(true);



    Simulator::Schedule(Seconds(2.0), &BroadcastPacket,
broadcastSocket);
```

```
    for (int i = 0; i < 7; ++i) {
        UdpEchoServerHelper echoServer(9);
        ApplicationContainer serverApps =
echoServer.Install(nodes.Get(i + 1));
        serverApps.Start(Seconds(1.0));
        serverApps.Stop(Seconds(10.0));
    }


    AnimationInterface anim("output/star_topology_broadcast.xml");
    anim.SetConstantPosition(nodes.Get(0), 300, 300);

    anim.SetConstantPosition(nodes.Get(1), 100, 200);
    anim.SetConstantPosition(nodes.Get(2), 100, 300);
    anim.SetConstantPosition(nodes.Get(3), 100, 400);
    anim.SetConstantPosition(nodes.Get(4), 100, 500);
    anim.SetConstantPosition(nodes.Get(5), 100, 600);
    anim.SetConstantPosition(nodes.Get(6), 100, 700);
    anim.SetConstantPosition(nodes.Get(7), 100, 800);


    AsciiTraceHelper ascii;
    pointToPoint.EnableAsciiAll(ascii.CreateFileStream("trace.tr"));
```

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

4th bus-topology

```
#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-address-helper.h"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("BusTopologyExample");

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

```
  NodeContainer nodes;
  nodes.Create (5);


  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute("DataRate",
StringValue("10Mbps"));
  pointToPoint.SetChannelAttribute("Delay",
TimeValue(NanoSeconds(100)));


  NetDeviceContainer devices01;
  devices01 = pointToPoint.Install (nodes.Get(0), nodes.Get(1));


  NetDeviceContainer devices12;
  devices12 = pointToPoint.Install (nodes.Get(1), nodes.Get(2));

  NetDeviceContainer devices23;
  devices23 = pointToPoint.Install (nodes.Get(2), nodes.Get(3));


  NetDeviceContainer devices34;
  devices34 = pointToPoint.Install (nodes.Get(3), nodes.Get(4));
```

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


Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces01 = address.Assign (devices01);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces12 = address.Assign (devices12);
address.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces23 = address.Assign (devices23);
address.SetBase ("10.1.4.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces34 = address.Assign (devices34);


UdpServerHelper udpServer (9);
ApplicationContainer serverApp = udpServer.Install (nodes.Get (1));
serverApp.Start (Seconds (1.0));
serverApp.Stop (Seconds (10.0));


Ipv4Address remoteAddress = Ipv4Address("10.1.1.2");
UdpClientHelper udpClient (InetSocketAddress(remoteAddress, 9));
```

```
udpClient.SetAttribute ("MaxPackets", UintegerValue (320));
udpClient.SetAttribute ("Interval", TimeValue (MilliSeconds (50)));
udpClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApp = udpClient.Install (nodes.Get (0));
clientApp.Start (Seconds (2.0));
clientApp.Stop (Seconds (10.0));



pointToPoint.EnablePcapAll ("bus-topology");



LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);



AnimationInterface anim ("bus-topology.xml");



anim.SetConstantPosition (nodes.Get(0), 100, 100);
anim.SetConstantPosition (nodes.Get(1), 200, 100);
anim.SetConstantPosition (nodes.Get(2), 300, 100);
anim.SetConstantPosition (nodes.Get(3), 400, 100);
anim.SetConstantPosition (nodes.Get(4), 500, 100);
```

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

  return 0;
}
```

5th.....

```
#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/network-module.h"
#include "ns3/netanim-module.h"
#include "ns3/ipv4-global-routing-helper.h"
using namespace ns3;

int main(int argc, char *argv[])
{
CommandLine cmd(__FILE__);
cmd.Parse(argc,argv);
Time::SetResolution(Time::NS);
```

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

NodeContainer routers;
routers.Create(4);

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

NetDeviceContainer d0,d1,d2,d3,d4;
d0 = p2p.Install(nodes.Get(0),routers.Get(0));
d1 = p2p.Install(routers.Get(0),routers.Get(1));
d2 = p2p.Install(routers.Get(1),routers.Get(2));
d3 = p2p.Install(routers.Get(2),routers.Get(3));
d4 = p2p.Install(routers.Get(3),nodes.Get(1));


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

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

```
Ipv4InterfaceContainer i0 = address.Assign(d0);
address.SetBase("10.1.2.0","255.255.255.0");
Ipv4InterfaceContainer i1 = address.Assign(d1);
address.SetBase("10.1.3.0","255.255.255.0");
Ipv4InterfaceContainer i2 = address.Assign(d2);
address.SetBase("10.1.4.0","255.255.255.0");
Ipv4InterfaceContainer i3 = address.Assign(d3);
address.SetBase("10.1.5.0","255.255.255.0");
Ipv4InterfaceContainer i4 = address.Assign(d4);


Ipv4GlobalRoutingHelper::PopulateRoutingTables();

UdpEchoServerHelper server(9);
ApplicationContainer ServerApp = server.Install(nodes.Get(1));
ServerApp.Start(Seconds(1.0));
ServerApp.Stop(Seconds(10.0));

UdpEchoClientHelper client(i4.GetAddress(1),9);
client.SetAttribute("MaxPackets",UintegerValue(1));
client.SetAttribute("Interval",TimeValue(Seconds(1)));
client.SetAttribute("PacketSize",UintegerValue(1024));

ApplicationContainer clientApp = client.Install(nodes.Get(0));
clientApp.Start(Seconds(2.0));
```

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

AnimationInterface anim("5.xml");
anim.SetConstantPosition(nodes.Get(0),100,400);
anim.SetConstantPosition(nodes.Get(1),400,400);

for(int i=0;i<4; ++i) anim.SetConstantPosition(routers.Get(i),200 +
100*i, 300);

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

6th...

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

```
// 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;

  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 ();

  AnimationInterface anim ("6.xml");

  pointToPoint.EnablePcapAll ("6");
  csma.EnablePcap ("6", csmaDevices.Get (1), true);

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

}