**Aim: Installation of NetAnim & Animate a simple network using NetAnim in Network**

Simulator

**Code:**

**TO Install NETANIM** sudo apt install python3-pip

pip install pybindgen

sudo apt install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools

make sure you are in the folder /ns-allinone-3.41/netanim-3.109

make clean

qmake NetAnim.pro

make

./NetAnim

**Animate a simple network using NetAnim in Network Simulator**

#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);

}

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);

AnimationInterface anim ("Animation1.xml"); anim.SetConstantPosition(p2pNodes.Get(0), 10.0, 10.0);

anim.SetConstantPosition(csmaNodes.Get(0), 20.0, 20.0);

anim.SetConstantPosition(csmaNodes.Get(1), 30.0, 30.0);

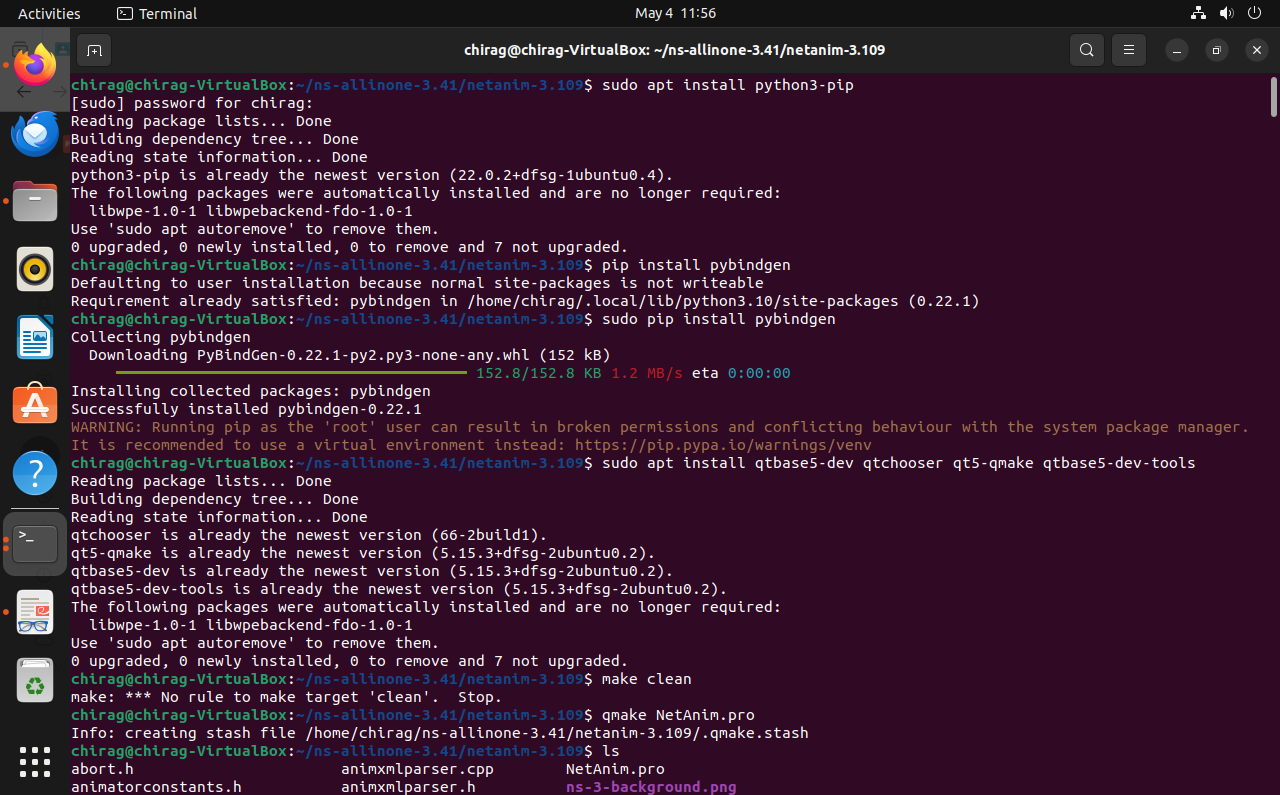
anim.SetConstantPosition(csmaNodes.Get(2), 40.0, 40.0);

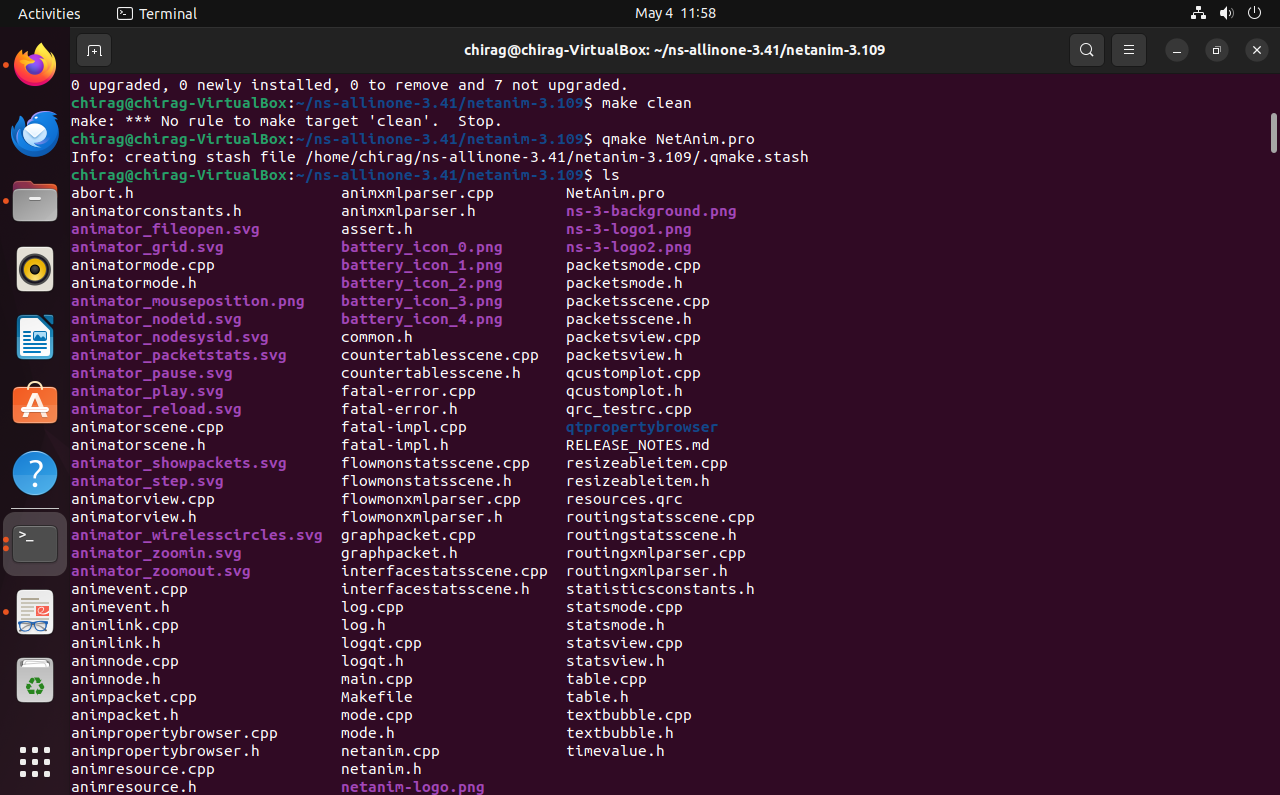
anim.SetConstantPosition(csmaNodes.Get(3), 50.0, 50.0); Simulator::Run();

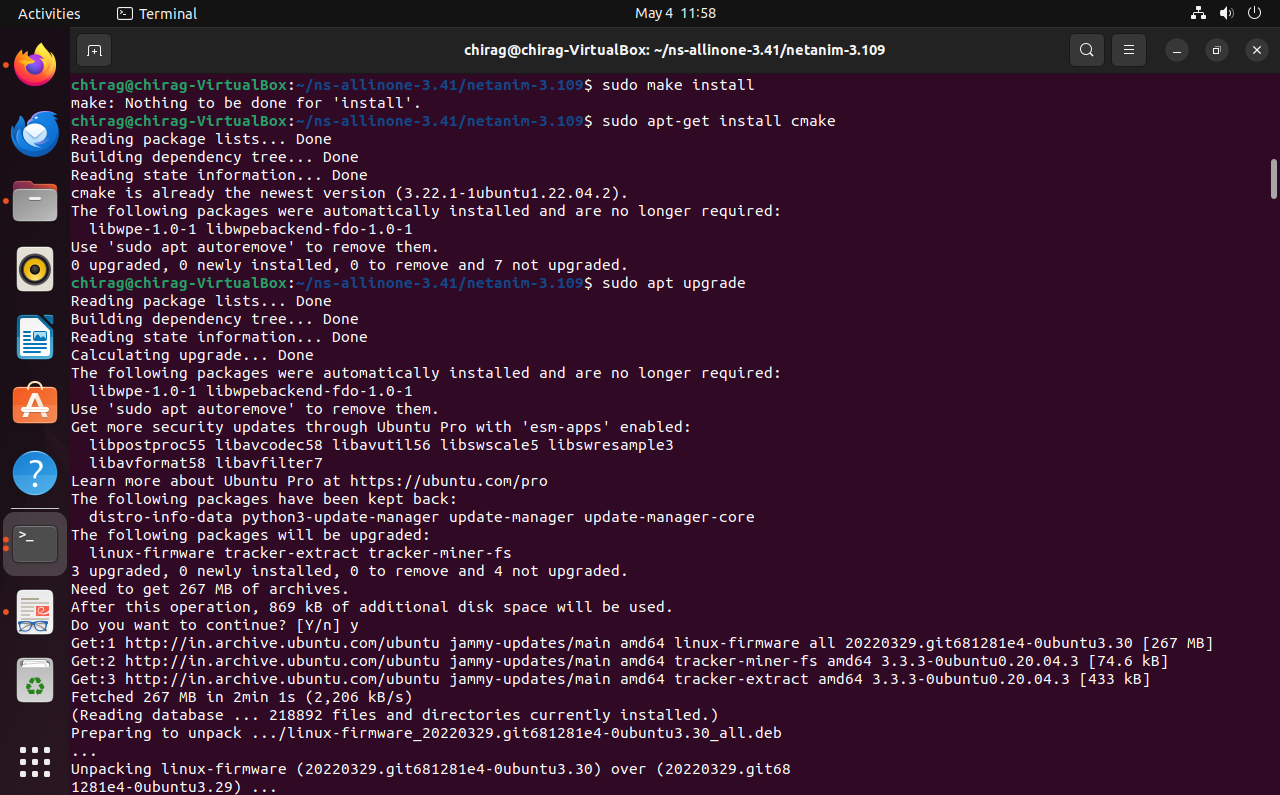
Simulator::Destroy();

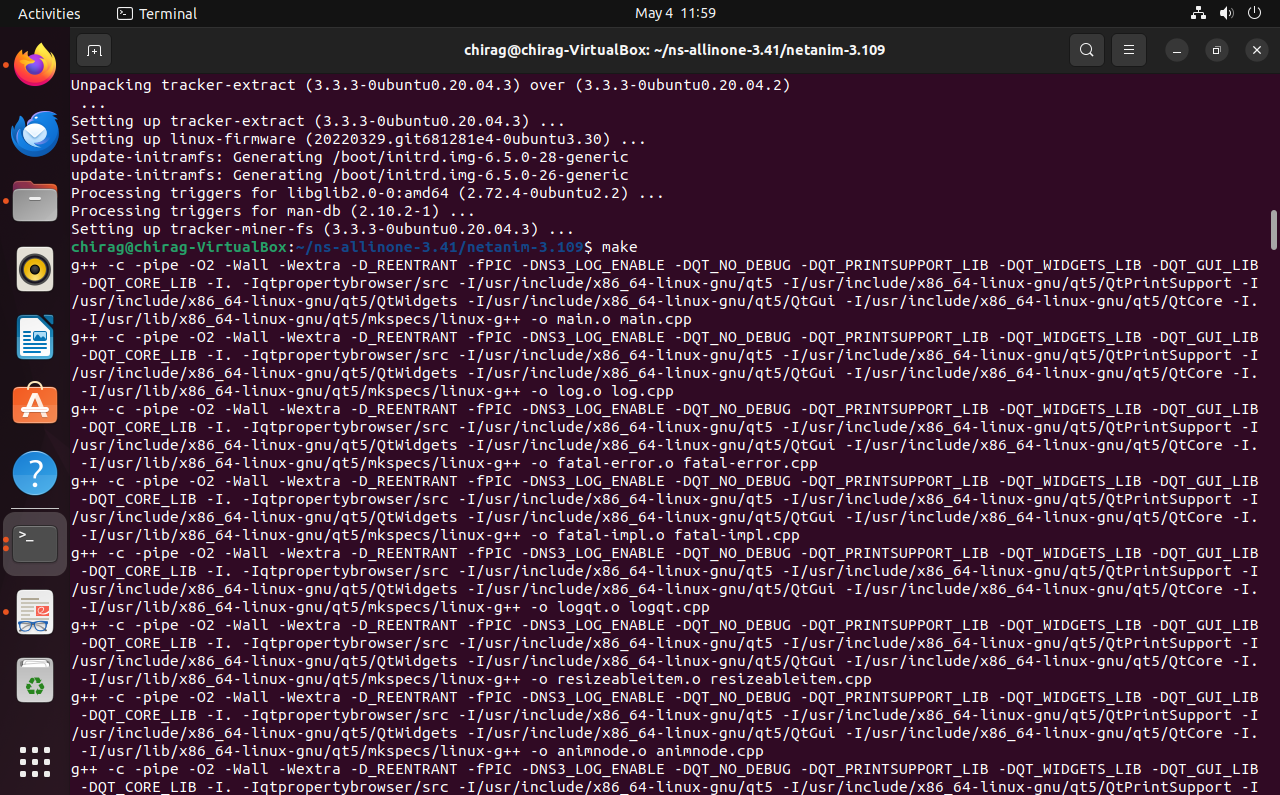
**Output:**

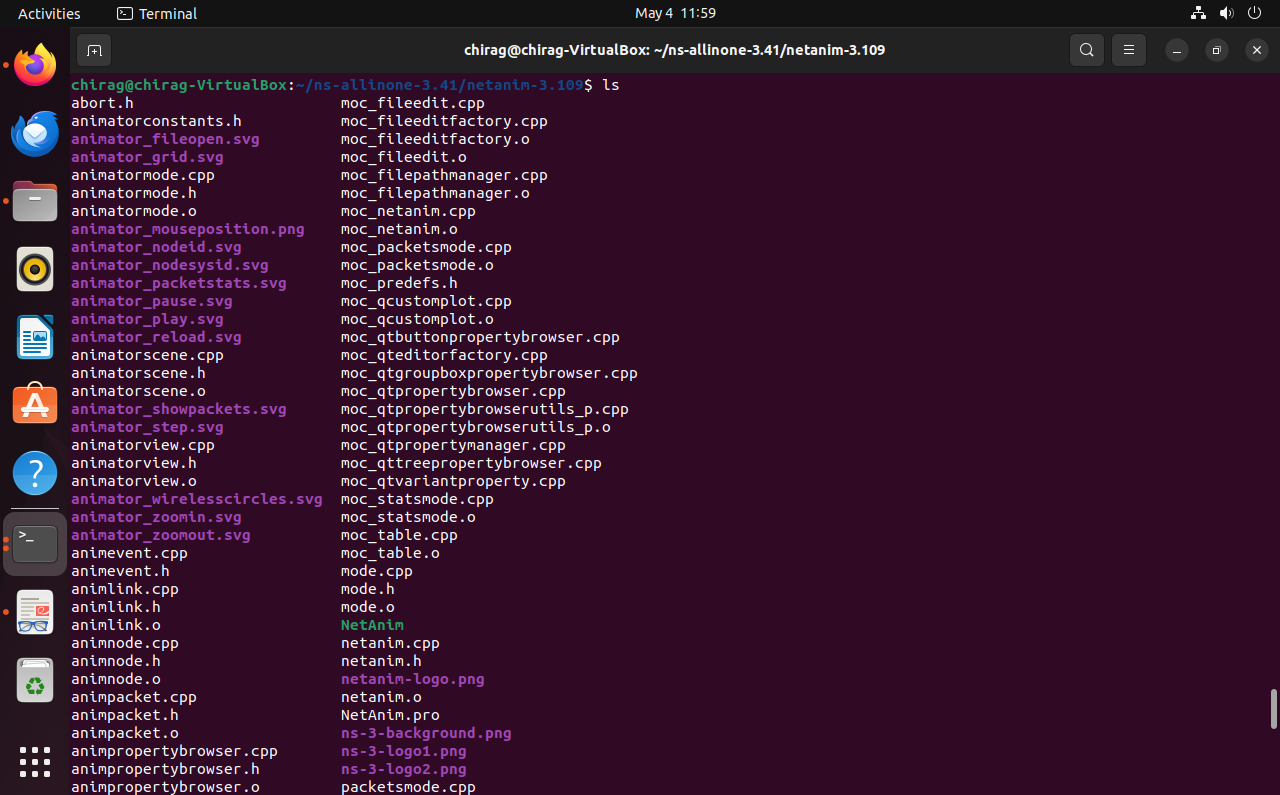
NetAnim

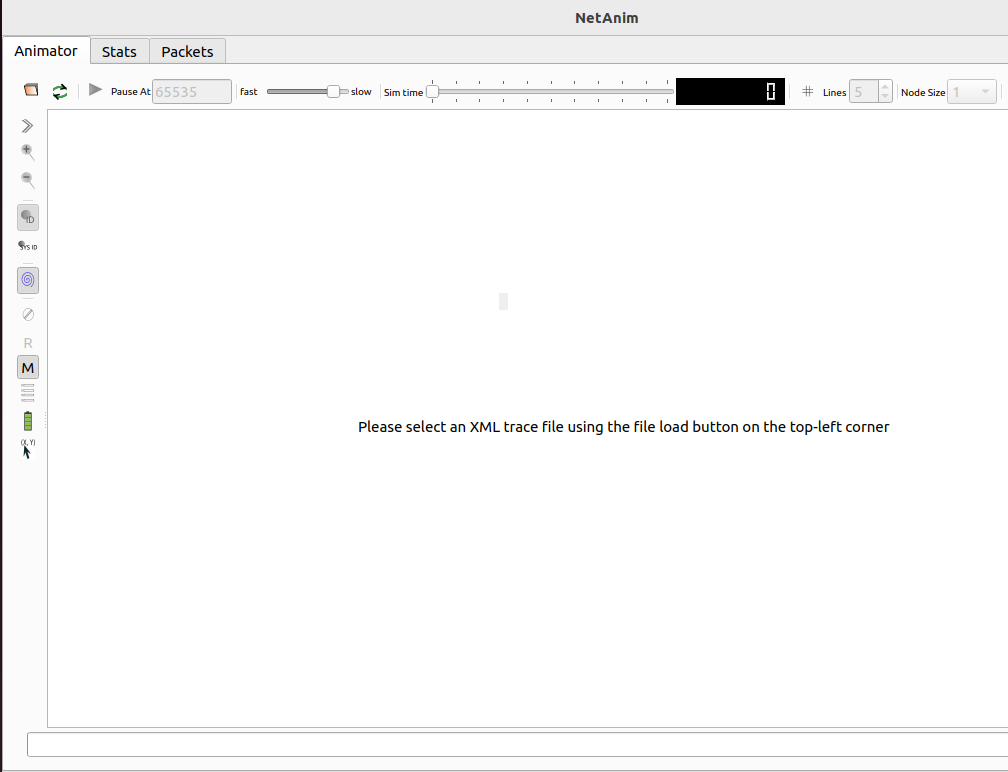








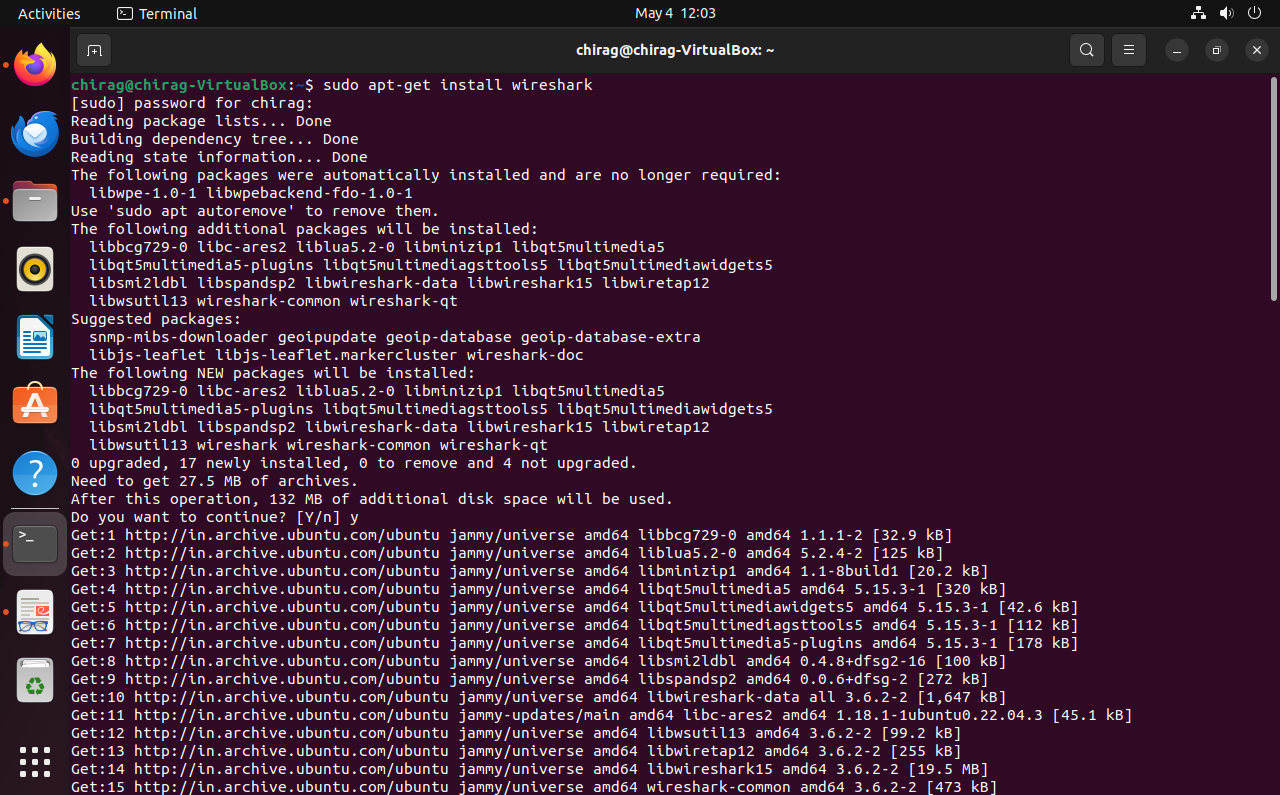


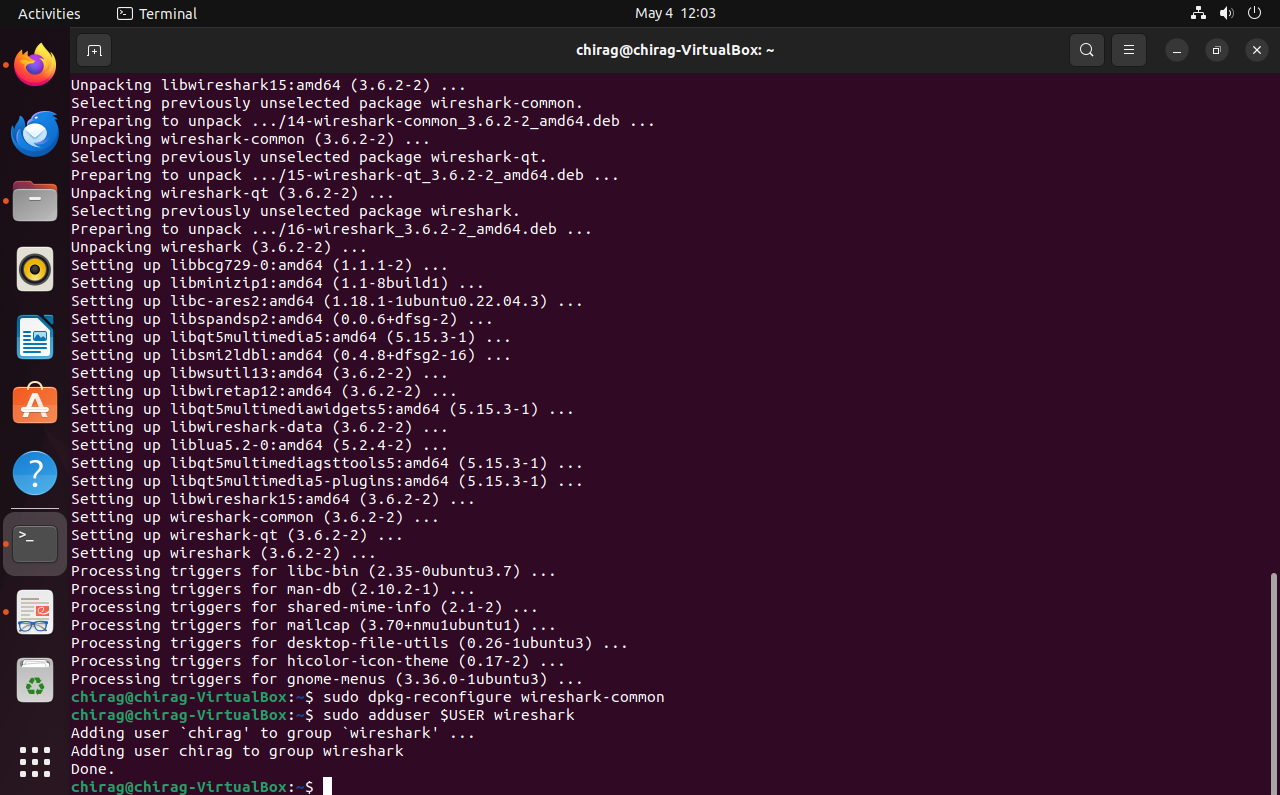


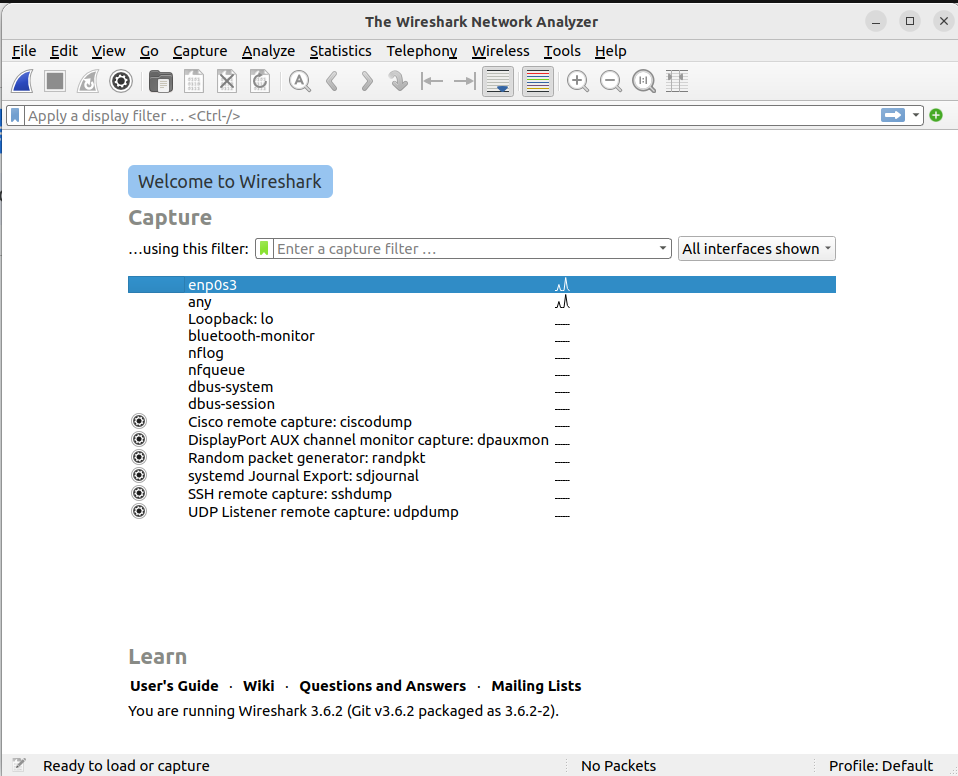
sudo apt-get install wireshark

sudo dpkg-reconfigure wireshark-common

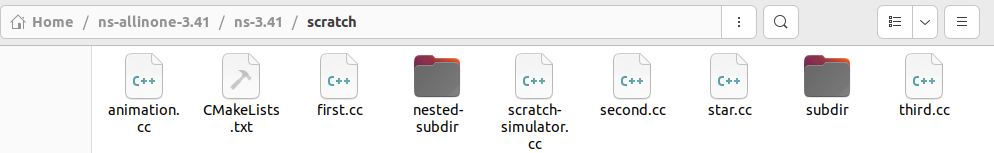
sudo adduser $USER wireshark







Part 2



animation.cc

/\*

 \* 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/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);

}

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);

    AnimationInterface anim ("Animation1.xml");

    anim.SetConstantPosition(p2pNodes.Get(0), 10.0, 10.0);

    anim.SetConstantPosition(csmaNodes.Get(0), 20.0, 20.0);

    anim.SetConstantPosition(csmaNodes.Get(1), 30.0, 30.0);

    anim.SetConstantPosition(csmaNodes.Get(2), 40.0, 40.0);

    anim.SetConstantPosition(csmaNodes.Get(3), 50.0, 50.0);

Simulator::Run();

Simulator::Destroy();

return 0;

}

