Networking with Linux Lab
Module 2 & 3: Client Server Network topology using NS-3 and Animating the Network

Assignment 4: Create a hybrid topology, and animate a simple network using NetAnim in Network Simulator.

1. Change the CSMA node to 15 through the command line and analyze the modifications.

Aim: To modify the CSMA node to 15 through the command line and analyze the impact of this change

Theory: **Hybrid Topology**

Hybrid Topology is a network topology that combines two or more different types of topologies to form a more robust and flexible network. It typically incorporates elements from star, bus, ring, and/or mesh topologies to leverage the strengths of each topology and mitigate their weaknesses. This approach allows for better scalability, reliability, and fault tolerance in complex network environments.

Command-Line Arguments

Command-line arguments are parameters supplied to a program when it is executed from the command line or terminal. They allow users to customize the behavior of a program without modifying its source code.

Code:

> third.cc

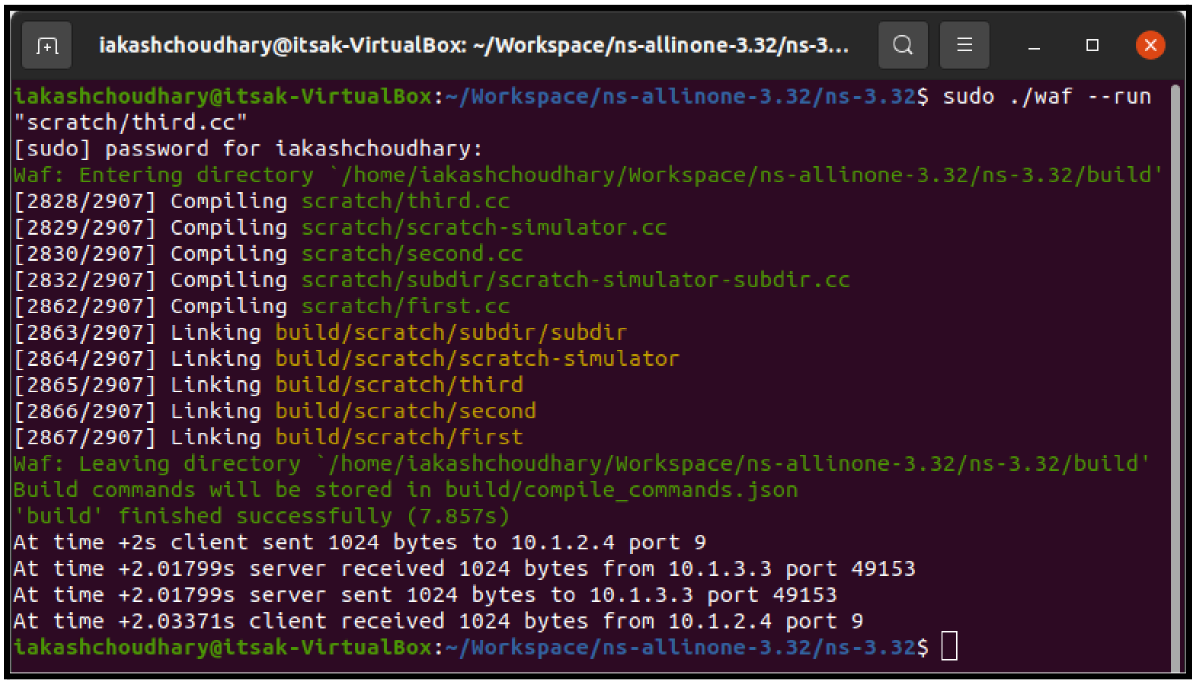
```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"

// Default Network Topology
//
```

```
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
  bool tracing = false;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

  cmd.Parse (argc,argv);

  if (nWifi > 18)
    {
      std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" <<
std::endl;
      return 1;
    }

  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

  PointToPointHelper pointToPoint;
```

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

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

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

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
          "Ssid", SsidValue (ssid),
          "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac",
          "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UintegerValue (3),
                               "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

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);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Stop (Seconds (10.0));

if (tracing == true)
  {
    pointToPoint.EnablePcapAll ("third");
    phy.EnablePcap ("third", apDevices.Get (0));
    csma.EnablePcap ("third", csmaDevices.Get (0), true);
  }

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

Command & Screenshot:

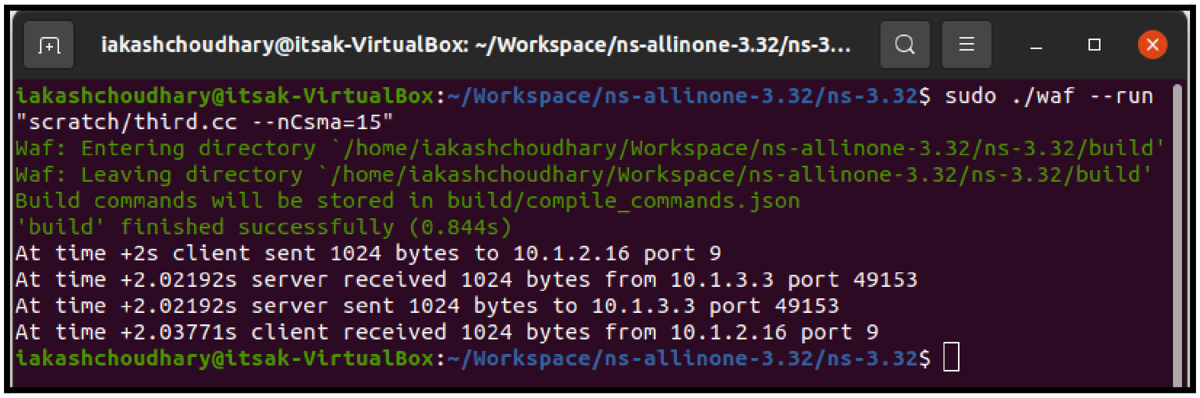
> $ ./waf --run "scratch/third.cc"



iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3...

```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run
"scratch/third.cc"
[sudo] password for iakashchoudhary:
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
[2828/2907] Compiling scratch/third.cc
[2829/2907] Compiling scratch/scratch-simulator.cc
[2830/2907] Compiling scratch/second.cc
[2832/2907] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2862/2907] Compiling scratch/first.cc
[2863/2907] Linking build/scratch/subdir/subdir
[2864/2907] Linking build/scratch/scratch-simulator
[2865/2907] Linking build/scratch/third
[2866/2907] Linking build/scratch/second
[2867/2907] Linking build/scratch/first
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (7.857s)
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.01799s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01799s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03371s client received 1024 bytes from 10.1.2.4 port 9
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$
```

> **$ ./waf --run "scratch/third.cc --nCsma=15"**


iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3...

```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run
"scratch/third.cc --nCsma=15"
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.844s)
At time +2s client sent 1024 bytes to 10.1.2.16 port 9
At time +2.02192s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.02192s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03771s client received 1024 bytes from 10.1.2.16 port 9
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$
```

Conclusion: Hence, the CSMA node has been successfully changed to 15 through the command line, indicating a modification in the system configuration.

2. Perform animation using PyGraphviz and NetAnim on a hybrid topology.

Aim: To demonstrate and visualize network communication in a hybrid topology using PyGraphviz and NetAnim through animation

Theory:

PyGraphviz is used to visualize network topologies. Network topology refers to the arrangement of nodes and connections in a network. Graph theory helps in modeling and understanding these topologies.

NetAnim is a network animation tool used with ns-3 for simulating and visualizing network behavior. It employs Discrete Event Simulation (DES) to model events like packet transmissions and node movements at discrete time points. The tool focuses on packet-level visualization, providing insights into data flow and network issues. Its real-time analysis feature aids in debugging and optimizing network protocols by offering a dynamic visual representation of the simulation.

Code:

> third.cc

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

// Default Network Topology
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0
```

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
  bool tracing = false;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

  cmd.Parse (argc,argv);

  // The underlying restriction of 18 is due to the grid position
  // allocator's configuration; the grid layout will exceed the
  // bounding box if more than 18 nodes are provided.
  if (nWifi > 18)
    {
      std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" <<
std::endl;
      return 1;
    }

  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

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

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

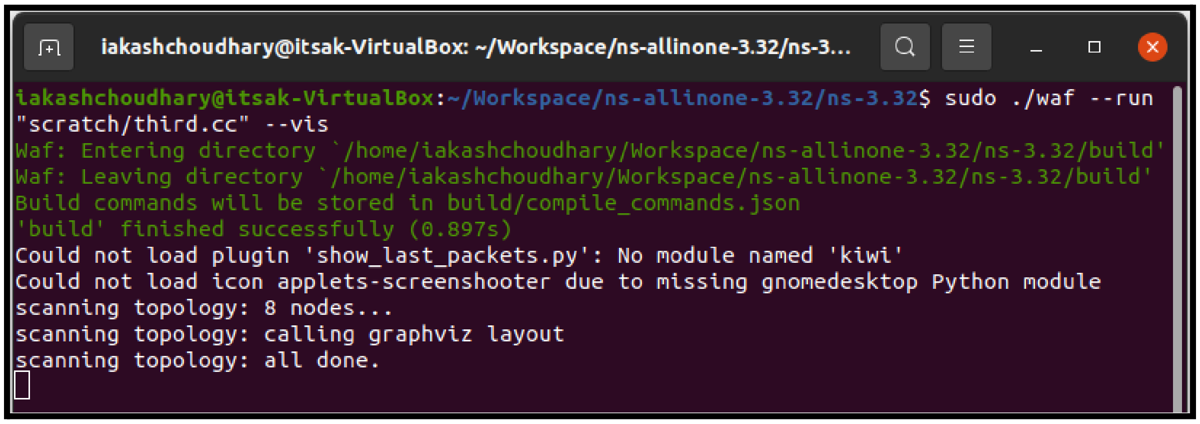
```
NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
          "Ssid", SsidValue (ssid),
          "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac",
          "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                         "MinX", DoubleValue (0.0),
                         "MinY", DoubleValue (0.0),
                         "DeltaX", DoubleValue (5.0),
                         "DeltaY", DoubleValue (10.0),
                         "GridWidth", UintegerValue (3),
                         "LayoutType", StringValue ("RowFirst"));
```

```
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                          "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

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);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Stop (Seconds (10.0));
```

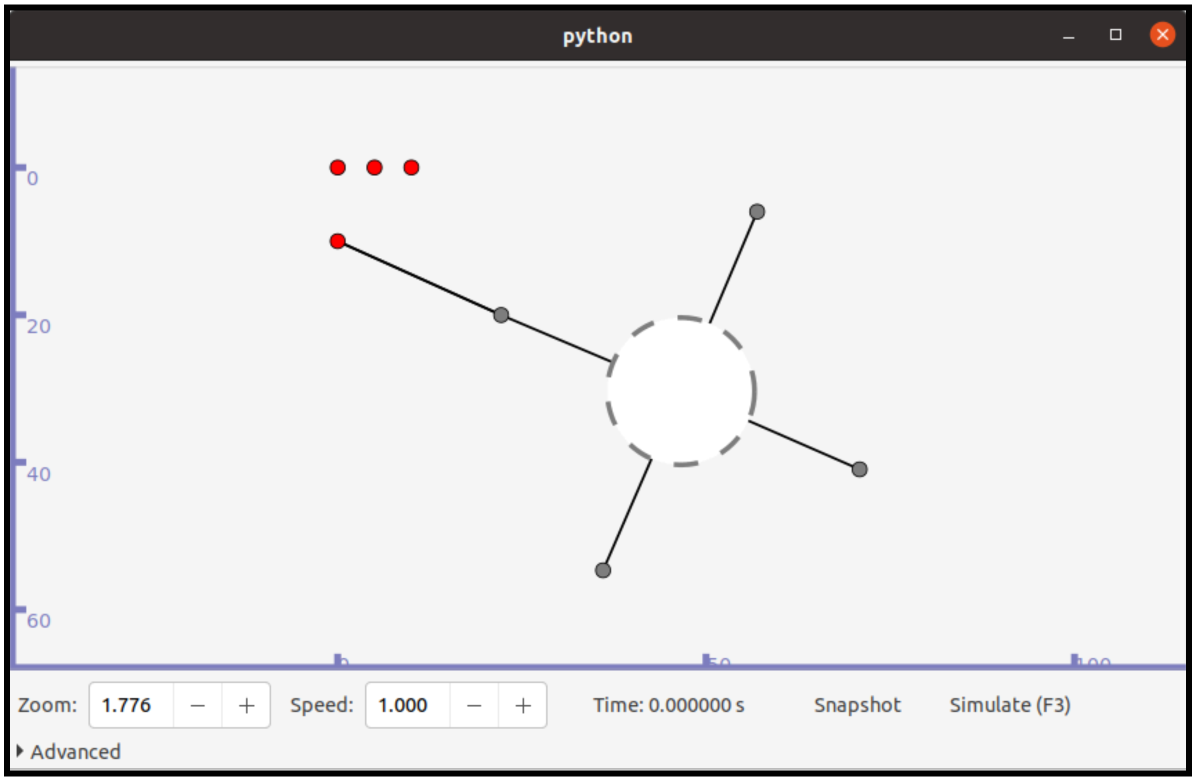
```
  // Creating pcap files for Wireshark-pcap: packet capture information
  if (tracing == true)
    {
      pointToPoint.EnablePcapAll ("third");
      phy.EnablePcap ("third", apDevices.Get (0));
      csma.EnablePcap ("third", csmaDevices.Get (0), true);
    }

  // NetAnimation
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(p2pNodes);
  mobility.Install(csmaNodes);

  AnimationInterface anim("third.xml");
  AnimationInterface::SetConstantPosition(p2pNodes.Get(0), 5, 15);
  AnimationInterface::SetConstantPosition(p2pNodes.Get(1), 10, 15);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(1), 15, 10);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(2), 20, 15);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(3), 15, 20);
  anim.EnablePacketMetadata(true);

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

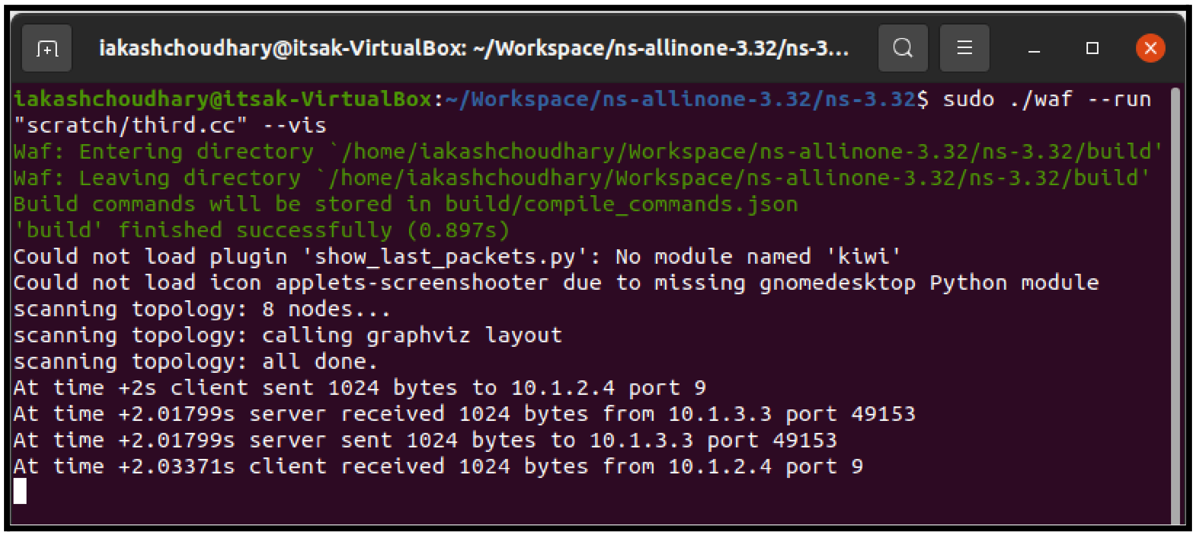
Command & Screenshot:

> Perform animation using pygraphviz: **$ ./waf --run "scratch/third.cc" --vis**

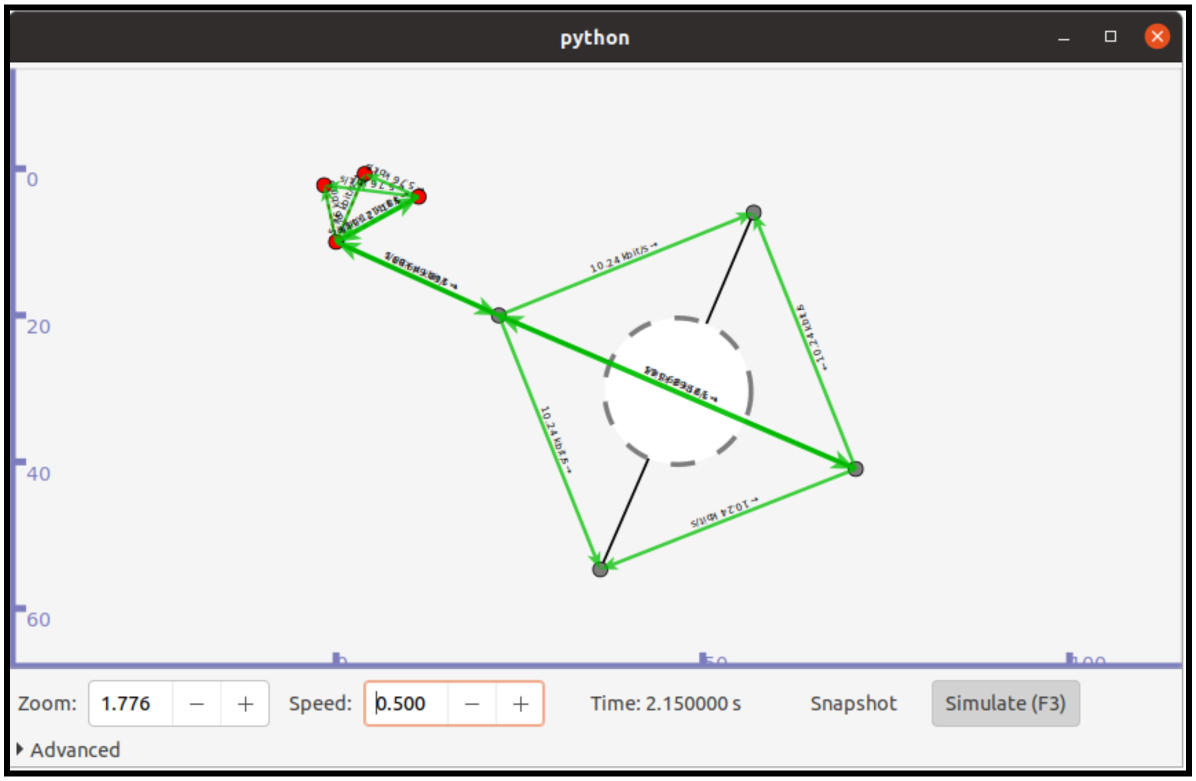


```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run
"scratch/third.cc" --vis
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.897s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 8 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
```



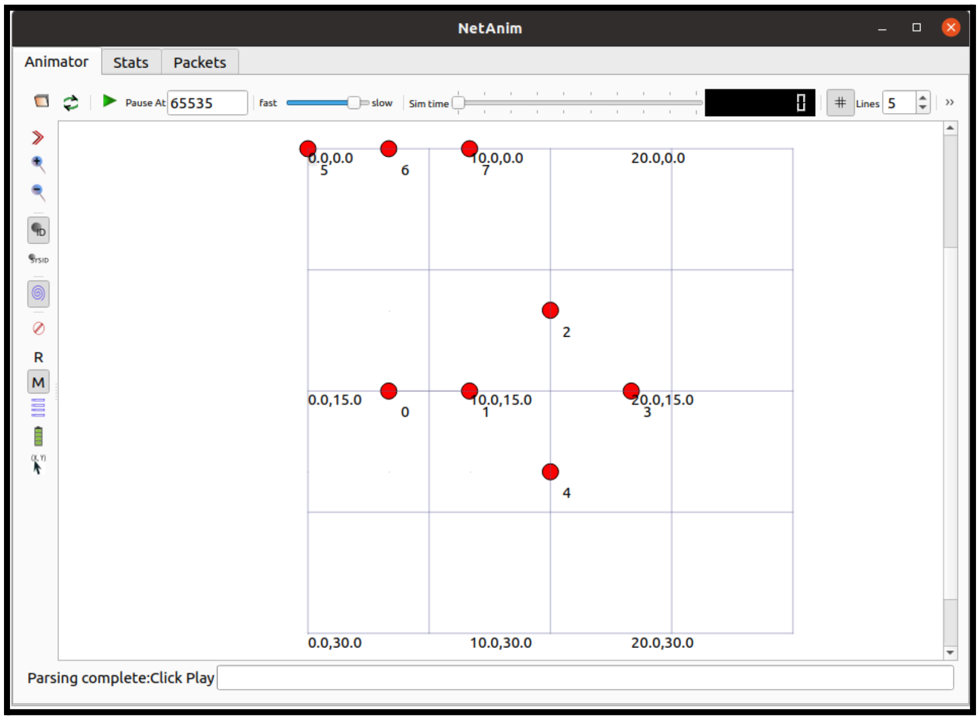

iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3...

```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run
"scratch/third.cc" --vis
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.897s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 8 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.01799s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01799s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03371s client received 1024 bytes from 10.1.2.4 port 9
```

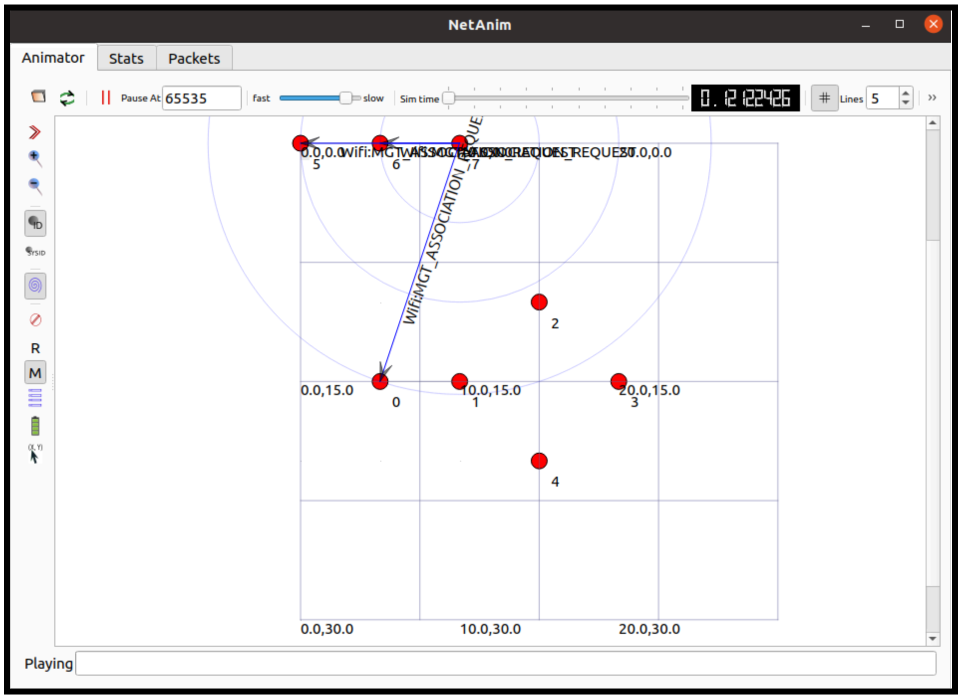


Zoom: 1.776 Speed: 0.500 Time: 2.150000 s Snapshot Simulate (F3)

> Perform animation using NetAnim: **$ ./NetAnim**



Parsing complete:Click Play



Playing

Conclusion: PyGraphviz and NetAnim can be effectively used to animate and visualize a hybrid topology in a network simulation.

3. Substitute LOG_LEVEL_INFO with any two of the following options:

> LOG_LEVEL_ALL
> LOG_PREFIX_FUNC
> LOG_PREFIX_TIME
> LOG_PREFIX_NODE
> LOG_PREFIX_LEVEL
> LOG_PREFIX_ALL

Aim: To substitute LOG_LEVEL_INFO with LOG_LEVEL_ALL and LOG_PREFIX_TIME, respectively, to explore and understand how different logging configurations impact the information displayed, focusing on log levels and function prefixes

Theory:

In a logging system, the choice of log levels and prefixes is essential for managing and understanding log messages. Let's substitute LOG_LEVEL_INFO with two options: LOG_LEVEL_ALL and LOG_PREFIX_TIME.

LOG_LEVEL_ALL

This log level includes all available log messages. It provides the highest level of detail and is useful for comprehensive debugging, but it may generate a significant amount of log data. Developers might use LOG_LEVEL_ALL during the development phase or when troubleshooting complex issues to capture a complete picture of the system's behavior.

LOG_PREFIX_TIME

This prefix adds a timestamp to each log message, indicating when the log entry was generated. It helps in tracking the chronological order of events and diagnosing time-related issues. Including the timestamp as a prefix is particularly valuable in distributed systems or applications where understanding the temporal sequence of events is crucial.

Code:

> third.cc

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

```
#include "ns3/ssid.h"

// Default Network Topology
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
  bool tracing = false;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

  cmd.Parse (argc,argv);

  if (nWifi > 18)
    {
      std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" <<
std::endl;
      return 1;
    }

  if (verbose)
    {
      // Uncomment the following lines as needed based on the requirements
      //LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_ALL);
      //LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_ALL);
      //LogComponentEnable ("UdpEchoClientApplication", LOG_PREFIX_TIME);
```

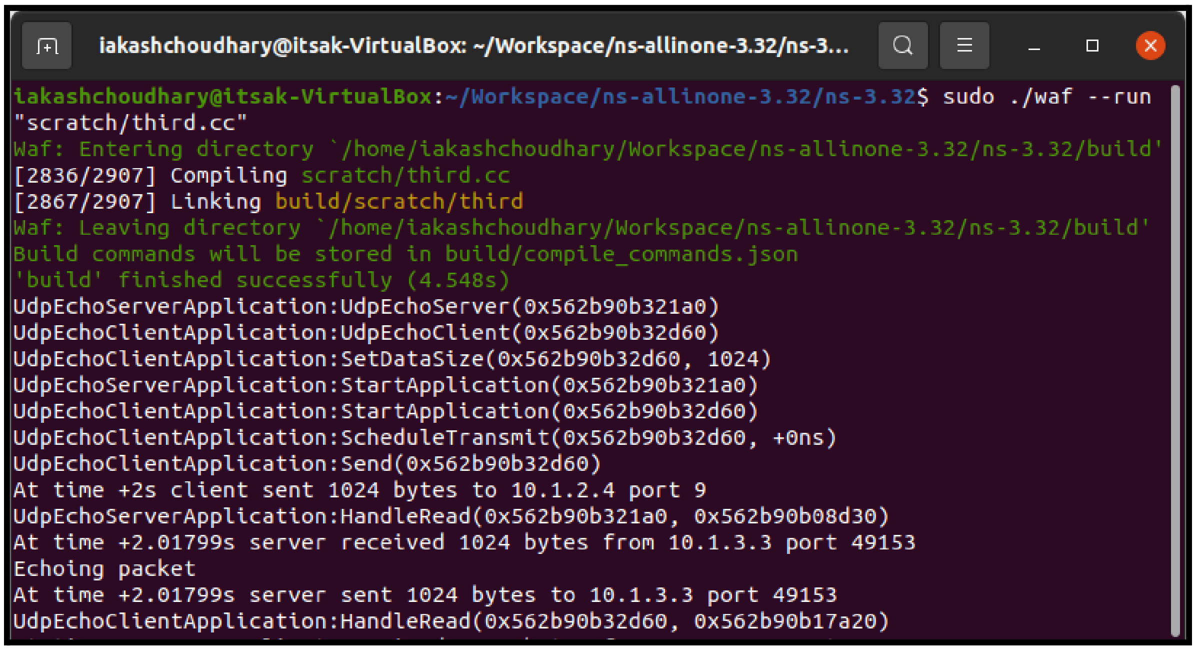
```
   //LogComponentEnable ("UdpEchoServerApplication", LOG_PREFIX_TIME);
  }

 NodeContainer p2pNodes;
 p2pNodes.Create (2);

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

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

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

 CsmaHelper csma;
 csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
 csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

 NetDeviceContainer csmaDevices;
 csmaDevices = csma.Install (csmaNodes);

 NodeContainer wifiStaNodes;
 wifiStaNodes.Create (nWifi);
 NodeContainer wifiApNode = p2pNodes.Get (0);

 YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
 YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
 phy.SetChannel (channel.Create ());

 WifiHelper wifi;
 wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

 WifiMacHelper mac;
 Ssid ssid = Ssid ("ns-3-ssid");
 mac.SetType ("ns3::StaWifiMac",
           "Ssid", SsidValue (ssid),
           "ActiveProbing", BooleanValue (false));

 NetDeviceContainer staDevices;
 staDevices = wifi.Install (phy, mac, wifiStaNodes);

 mac.SetType ("ns3::ApWifiMac",
           "Ssid", SsidValue (ssid));
```

```
NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UintegerValue (3),
                               "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

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);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (wifiStaNodes.Get (nWifi - 1));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  Simulator::Stop (Seconds (10.0));

  if (tracing == true)
    {
      pointToPoint.EnablePcapAll ("third");
      phy.EnablePcap ("third", apDevices.Get (0));
      csma.EnablePcap ("third", csmaDevices.Get (0), true);
    }

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

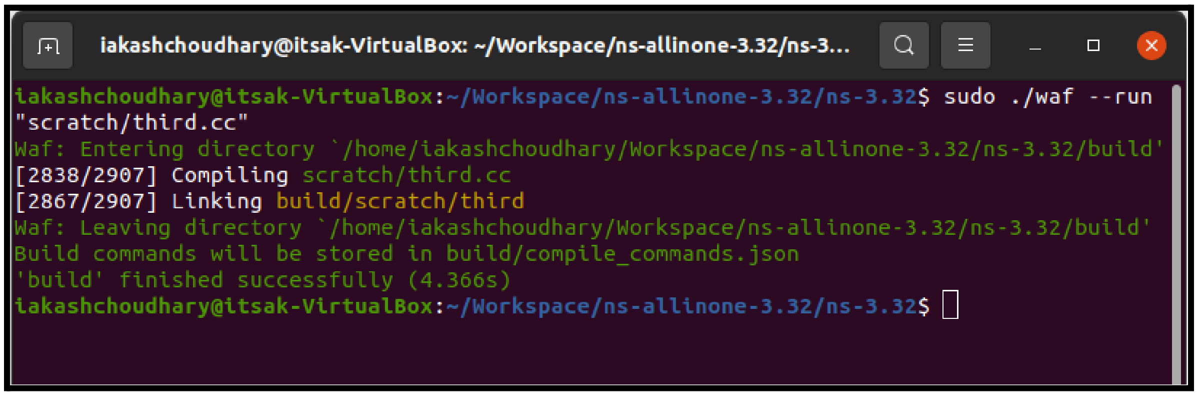
Command & Screenshot:

> For LOG_LEVEL_ALL messages: **$ ./waf --run "scratch/third.cc"**


iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3...

```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run
"scratch/third.cc"
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
[2836/2907] Compiling scratch/third.cc
[2867/2907] Linking build/scratch/third
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.548s)
UdpEchoServerApplication:UdpEchoServer(0x562b90b321a0)
UdpEchoClientApplication:UdpEchoClient(0x562b90b32d60)
UdpEchoClientApplication:SetDataSize(0x562b90b32d60, 1024)
UdpEchoServerApplication:StartApplication(0x562b90b321a0)
UdpEchoClientApplication:StartApplication(0x562b90b32d60)
UdpEchoClientApplication:ScheduleTransmit(0x562b90b32d60, +0ns)
UdpEchoClientApplication:Send(0x562b90b32d60)
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
UdpEchoServerApplication:HandleRead(0x562b90b321a0, 0x562b90b08d30)
At time +2.01799s server received 1024 bytes from 10.1.3.3 port 49153
Echoing packet
At time +2.01799s server sent 1024 bytes to 10.1.3.3 port 49153
UdpEchoClientApplication:HandleRead(0x562b90b32d60, 0x562b90b17a20)
```

> For LOG_PREFIX_TIME messages: **$ ./waf --run "scratch/third.cc"**


iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3...

```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run
"scratch/third.cc"
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
[2838/2907] Compiling scratch/third.cc
[2867/2907] Linking build/scratch/third
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.366s)
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$
```

Conclusion: Hence, we were successfully able to substitute LOG_LEVEL_INFO with LOG_LEVEL_ALL and LOG_PREFIX_TIME for logging purposes.

4. Send four packets from the client to the server through the command prompt.

Aim: To test and establish a hybrid network connection between a client and server by sending four packets through the command prompt

Theory:

Command-Line Arguments

Command-line arguments are parameters supplied to a program when it is executed from the command line or terminal. They allow users to customize the behavior of a program without modifying its source code.

Code:

> third.cc

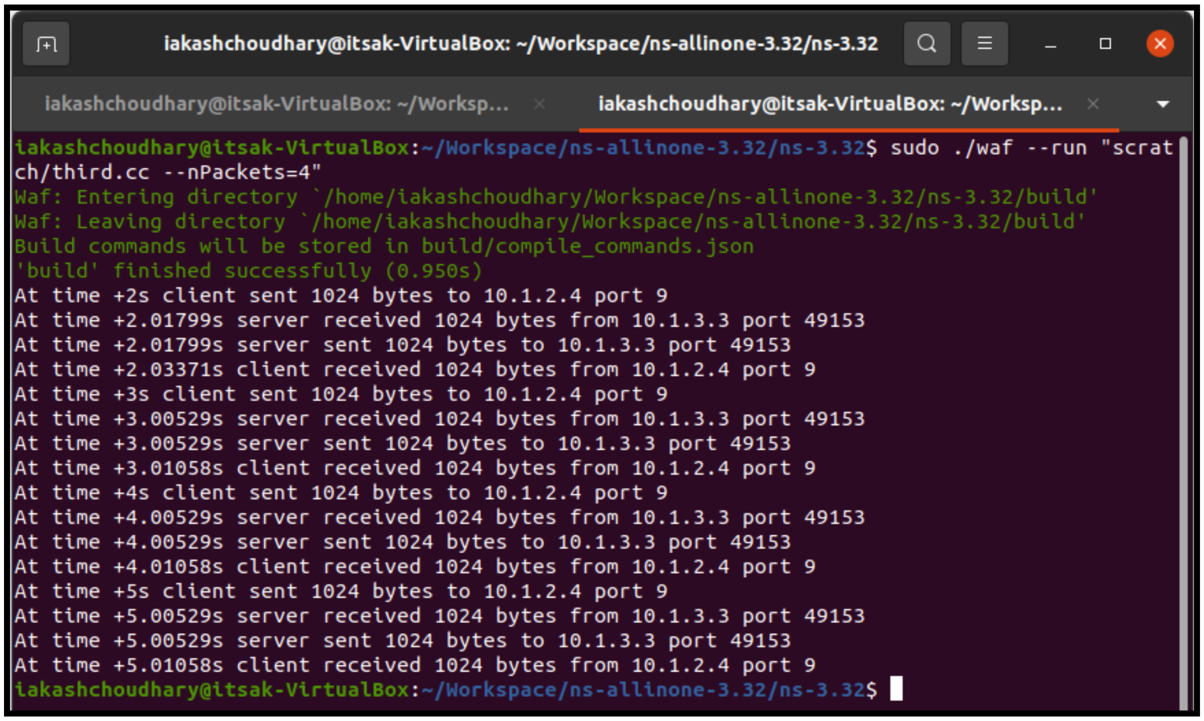
```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include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

// Default Network Topology
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

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

```
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
  uint32_t nPackets = 1;
  bool tracing = false;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
  cmd.AddValue ("nPackets", "Number of packets to echo", nPackets);

  cmd.Parse (argc,argv);

  // The underlying restriction of 18 is due to the grid position
  // allocator's configuration; the grid layout will exceed the
  // bounding box if more than 18 nodes are provided.
  if (nWifi > 18)
    {
      std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" <<
std::endl;
      return 1;
    }

  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

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

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

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

```
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
        "Ssid", SsidValue (ssid),
        "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac",
        "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                "MinX", DoubleValue (0.0),
                "MinY", DoubleValue (0.0),
                "DeltaX", DoubleValue (5.0),
                "DeltaY", DoubleValue (10.0),
                "GridWidth", UintegerValue (3),
                "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
               "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

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);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (nPackets));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps =
  echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Stop (Seconds (10.0));

// Creating pcap files for Wireshark-pcap: packet capture information
if (tracing == true)
  {
    pointToPoint.EnablePcapAll ("third");
    phy.EnablePcap ("third", apDevices.Get (0));
    csma.EnablePcap ("third", csmaDevices.Get (0), true);
  }

 // NetAnimation
 mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
 mobility.Install(p2pNodes);
 mobility.Install(csmaNodes);

 AnimationInterface anim("third.xml");
 AnimationInterface::SetConstantPosition(p2pNodes.Get(0), 5, 15);
 AnimationInterface::SetConstantPosition(p2pNodes.Get(1), 10, 15);
 AnimationInterface::SetConstantPosition(csmaNodes.Get(1), 15, 10);
 AnimationInterface::SetConstantPosition(csmaNodes.Get(2), 20, 15);
 AnimationInterface::SetConstantPosition(csmaNodes.Get(3), 15, 20);
 anim.EnablePacketMetadata(true);

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

Command & Screenshot:

> $ ./waf --run "scratch/third.cc --nPackets=4"



```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scrat
ch/third.cc --nPackets=4"
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.950s)
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.01799s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01799s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03371s client received 1024 bytes from 10.1.2.4 port 9
At time +3s client sent 1024 bytes to 10.1.2.4 port 9
At time +3.00529s server received 1024 bytes from 10.1.3.3 port 49153
At time +3.00529s server sent 1024 bytes to 10.1.3.3 port 49153
At time +3.01058s client received 1024 bytes from 10.1.2.4 port 9
At time +4s client sent 1024 bytes to 10.1.2.4 port 9
At time +4.00529s server received 1024 bytes from 10.1.3.3 port 49153
At time +4.00529s server sent 1024 bytes to 10.1.3.3 port 49153
At time +4.01058s client received 1024 bytes from 10.1.2.4 port 9
At time +5s client sent 1024 bytes to 10.1.2.4 port 9
At time +5.00529s server received 1024 bytes from 10.1.3.3 port 49153
At time +5.00529s server sent 1024 bytes to 10.1.3.3 port 49153
At time +5.01058s client received 1024 bytes from 10.1.2.4 port 9
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$
```

Conclusion: Successful transmission of packets indicates a functional network connection between the client and server.

5. Obtain IP address information using the Wireshark tool.

Aim: To analyze network traffic using Wireshark and obtain IP address information

Theory:

Wireshark

Wireshark is a popular open-source network protocol analyzer that allows users to capture and inspect data traveling back and forth on a network in real-time. It is commonly used for network troubleshooting, analysis, software and protocol development, and education. Wireshark supports a wide range of protocols and provides a comprehensive set of features for capturing, analyzing, and displaying network traffic.

Code:

> third.cc

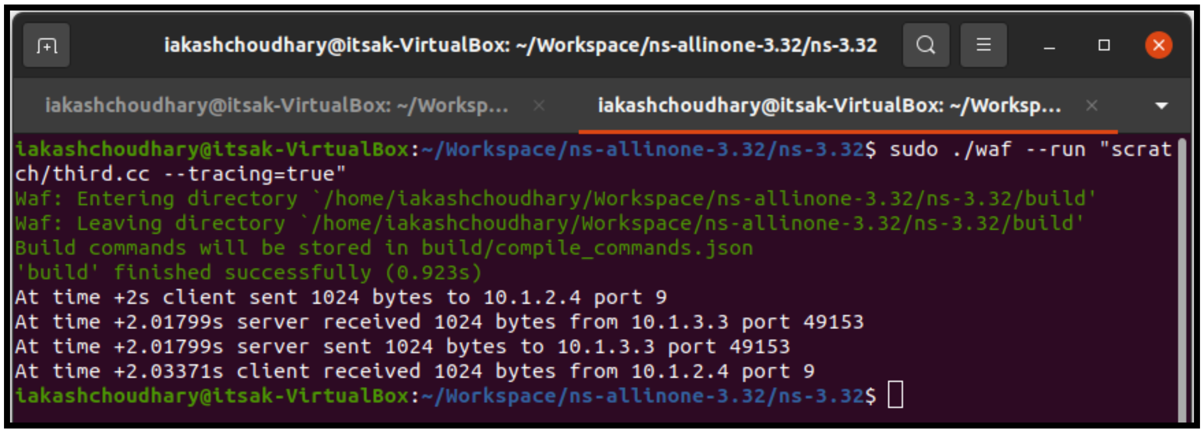
```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include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

// Default Network Topology
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

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

```
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
  uint32_t nPackets = 1;
  bool tracing = false;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
  cmd.AddValue ("nPackets", "Number of packets to echo", nPackets);

  cmd.Parse (argc,argv);

  // The underlying restriction of 18 is due to the grid position
  // allocator's configuration; the grid layout will exceed the
  // bounding box if more than 18 nodes are provided.
  if (nWifi > 18)
    {
      std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box" <<
std::endl;
      return 1;
    }

  if (verbose)
    {
      LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

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

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

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

```
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
          "Ssid", SsidValue (ssid),
          "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac",
          "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                          "MinX", DoubleValue (0.0),
                          "MinY", DoubleValue (0.0),
                          "DeltaX", DoubleValue (5.0),
                          "DeltaY", DoubleValue (10.0),
                          "GridWidth", UintegerValue (3),
                          "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                       "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

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);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (nPackets));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps =
  echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Simulator::Stop (Seconds (10.0));

// Creating pcap files for Wireshark-pcap: packet capture information
if (tracing == true)
  {
      pointToPoint.EnablePcapAll ("third");
      phy.EnablePcap ("third", apDevices.Get (0));
      csma.EnablePcap ("third", csmaDevices.Get (0), true);
    }

  // NetAnimation
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(p2pNodes);
  mobility.Install(csmaNodes);

  AnimationInterface anim("third.xml");
  AnimationInterface::SetConstantPosition(p2pNodes.Get(0), 5, 15);
  AnimationInterface::SetConstantPosition(p2pNodes.Get(1), 10, 15);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(1), 15, 10);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(2), 20, 15);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(3), 15, 20);
  anim.EnablePacketMetadata(true);

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

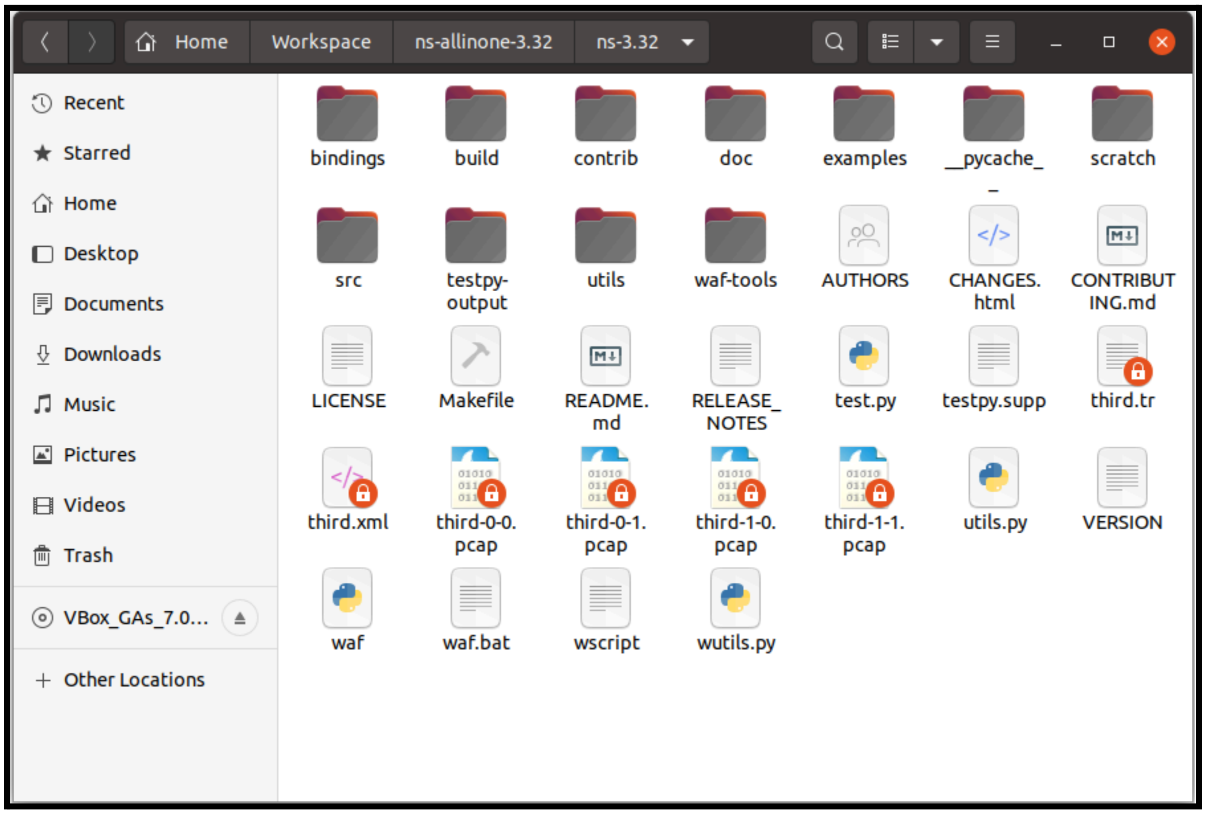
Command & Screenshot:

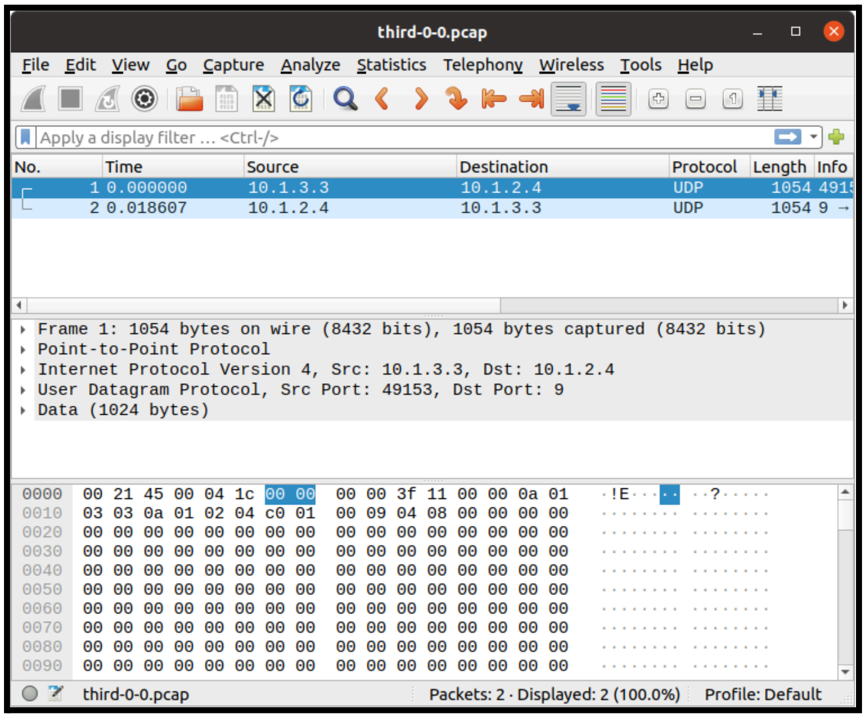
> $ sudo ./waf --run "scratch/third.cc --tracing=true"



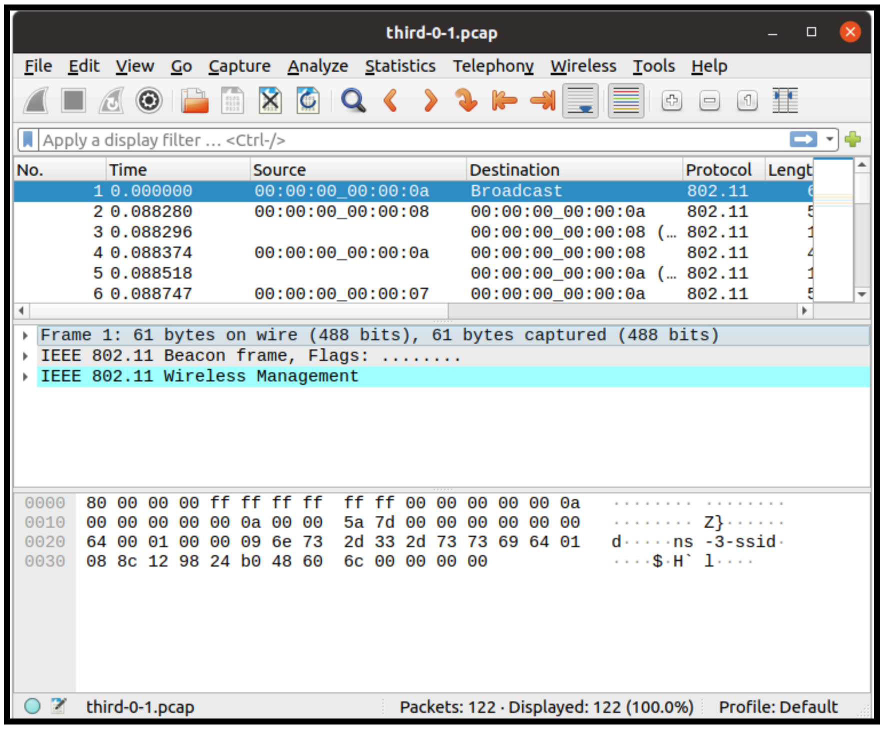
```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scrat
ch/third.cc --tracing=true"
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.923s)
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.01799s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01799s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03371s client received 1024 bytes from 10.1.2.4 port 9
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ 
```

> After executing the above command, a **.pcap file is created** for Wireshark analysis. To open each .pcap file, simply **double-click** on it.



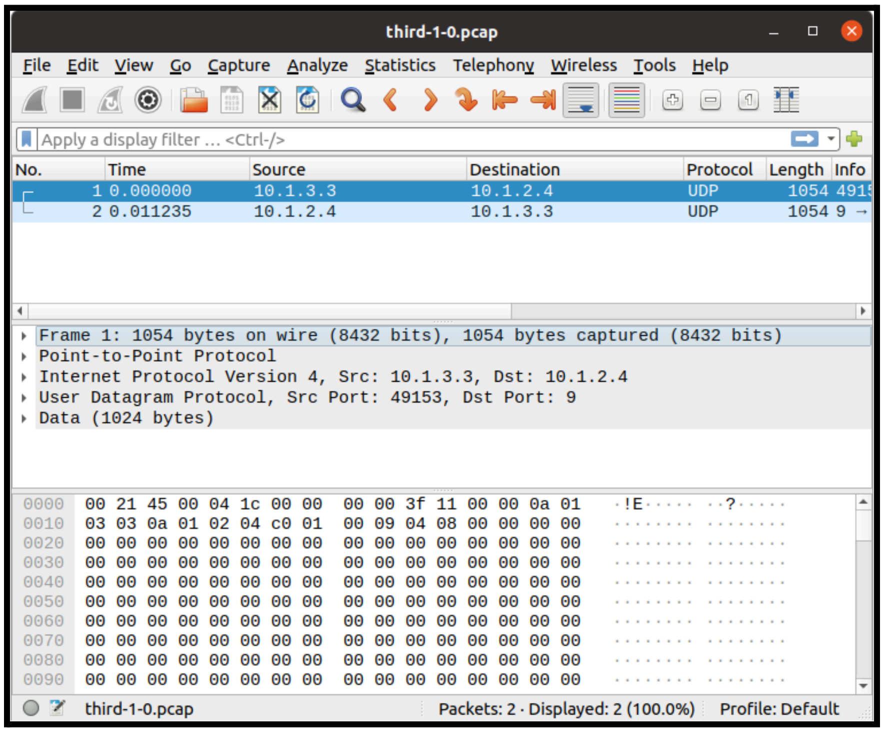

third-0-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.  Time       Source      Destination  Protocol  Length  Info
   1 0.000000   10.1.3.3    10.1.2.4     UDP       1054    491
   2 0.018607   10.1.2.4    10.1.3.3     UDP       1054    9 →

Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.3.3, Dst: 10.1.2.4
User Datagram Protocol, Src Port: 49153, Dst Port: 9
Data (1024 bytes)

0000  00 21 45 00 04 1c 00 00  00 00 3f 11 00 00 0a 01   ·!E····· ··?·····
0010  03 03 0a 01 02 04 c0 01  00 09 04 08 00 00 00 00   ········ ········
0020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········

third-0-0.pcap    Packets: 2 · Displayed: 2 (100.0%)    Profile: Default
```


third-0-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

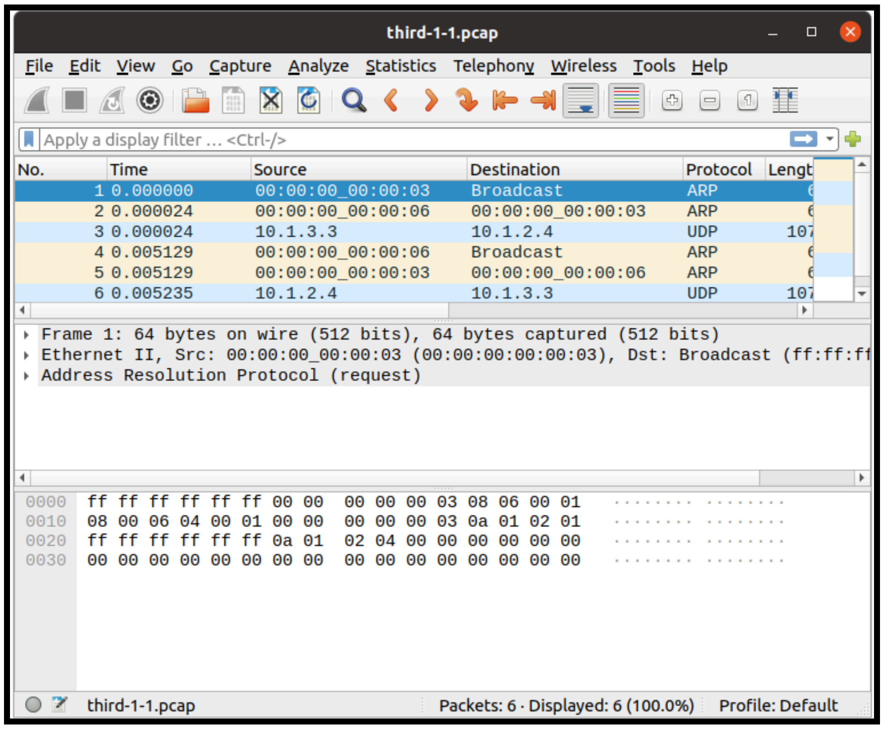
```
No.  Time      Source              Destination            Protocol  Lengt
   1 0.000000  00:00:00_00:00:0a   Broadcast              802.11
   2 0.088280  00:00:00_00:00:08   00:00:00_00:00:0a      802.11
   3 0.088296                      00:00:00_00:00:08 (…   802.11
   4 0.088374  00:00:00_00:00:0a   00:00:00_00:00:08      802.11
   5 0.088518                      00:00:00_00:00:0a (…   802.11
   6 0.088747  00:00:00_00:00:07   00:00:00_00:00:0a      802.11

Frame 1: 61 bytes on wire (488 bits), 61 bytes captured (488 bits)
IEEE 802.11 Beacon frame, Flags: ........
IEEE 802.11 Wireless Management

0000  80 00 00 00 ff ff ff ff  ff ff 00 00 00 00 00 0a   ········ ········
0010  00 00 00 00 00 0a 00 00  5a 7d 00 00 00 00 00 00   ········ Z}······
0020  64 00 01 00 00 09 6e 73  2d 33 2d 73 73 69 64 01   d·····ns -3-ssid·
0030  08 8c 12 98 24 b0 48 60  6c 00 00 00 00            ····$·H` l····
```

third-0-1.pcap Packets: 122 · Displayed: 122 (100.0%) Profile: Default


third-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time      Source     Destination  Protocol  Length  Info
   1 0.000000  10.1.3.3   10.1.2.4     UDP         1054  4915
   2 0.011235  10.1.2.4   10.1.3.3     UDP         1054  9 →

Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.3.3, Dst: 10.1.2.4
User Datagram Protocol, Src Port: 49153, Dst Port: 9
Data (1024 bytes)

0000  00 21 45 00 04 1c 00 00  00 00 3f 11 00 00 0a 01   ·!E····· ··?·····
0010  03 03 0a 01 02 04 c0 01  00 09 04 08 00 00 00 00   ········ ········
0020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
```

third-1-0.pcap Packets: 2 · Displayed: 2 (100.0%) Profile: Default


third-1-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.     Time        Source               Destination          Protocol  Lengt
      1 0.000000    00:00:00_00:00:03    Broadcast            ARP       6
      2 0.000024    00:00:00_00:00:06    00:00:00_00:00:03    ARP       6
      3 0.000024    10.1.3.3             10.1.2.4             UDP       107
      4 0.005129    00:00:00_00:00:06    Broadcast            ARP       6
      5 0.005129    00:00:00_00:00:03    00:00:00_00:00:06    ARP       6
      6 0.005235    10.1.2.4             10.1.3.3             UDP       107
```

```
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
Ethernet II, Src: 00:00:00_00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff
Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 00 00  00 00 00 03 08 06 00 01   ........ ........
0010  08 00 06 04 00 01 00 00  00 00 00 03 0a 01 02 01   ........ ........
0020  ff ff ff ff ff ff 0a 01  02 04 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

Conclusion: Wireshark successfully captured and revealed IP address details, facilitating comprehensive network analysis.

6. Create a Trace file (.tr file) and then retrieve information using the TraceMetric Tool/tcpdump.

Aim: To generate a trace file (.tr file) and analyze network traffic using the TraceMetric Tool/tcpdump for performance evaluation and debugging purposes

Theory:

TraceMetric Tool

TraceMetric tools are software utilities designed to analyze and extract specific metrics or information from trace files. These tools offer functionalities to parse trace files, filter data based on criteria, calculate various network metrics (e.g., throughput, packet loss, delay), and present results in a readable format.

Depending on the tool, it may have a graphical user interface (GUI) or operate through a command-line interface (CLI).

tcpdump

tcpdump is a widely-used command-line packet analyzer for Unix-like systems. It captures packets traversing a network interface and saves them to a file or displays them in real-time.

tcpdump allows users to apply filters based on protocol, source/destination IP addresses, port numbers, etc., to capture specific network traffic. By analyzing the captured packets, users can gain insights into network behavior, diagnose problems, and troubleshoot network issues.

Code:

> third.cc

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

// Default Network Topology
//
//   Wifi 10.1.3.0
//                  AP
//  *      *        *        *
//  |      |        |        |          10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                  point-to-point  |          |          |          |
//                                  ================
//                                  LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
  uint32_t nPackets = 1;
  bool tracing = false;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
  cmd.AddValue ("nPackets", "Number of packets to echo", nPackets);

  cmd.Parse (argc,argv);

  // The underlying restriction of 18 is due to the grid position
  // allocator's configuration; the grid layout will exceed the
  // bounding box if more than 18 nodes are provided.
  if (nWifi > 18)
        {
        std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding
box" << std::endl;
        return 1;
        }

  if (verbose)
        {
        LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
        }
```

```
NodeContainer p2pNodes;
p2pNodes.Create (2);

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

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

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

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());

WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
        "Ssid", SsidValue (ssid),
        "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac",
        "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);
```

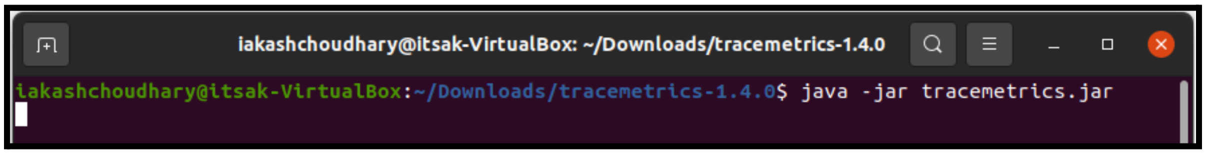
```
MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
                               "GridWidth", UintegerValue (3),
                               "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

InternetStackHelper stack;
stack.Install (csmaNodes);
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

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);

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (nPackets));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
```

```
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

  ApplicationContainer clientApps =
          echoClient.Install (wifiStaNodes.Get (nWifi - 1));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  Simulator::Stop (Seconds (10.0));

  // Creating pcap files for Wireshark-pcap: packet capture information
  if (tracing == true)
          {
          pointToPoint.EnablePcapAll ("third");
          phy.EnablePcap ("third", apDevices.Get (0));
          csma.EnablePcap ("third", csmaDevices.Get (0), true);
          }

  // NetAnimation
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(p2pNodes);
  mobility.Install(csmaNodes);

  AnimationInterface anim("third.xml");
  AnimationInterface::SetConstantPosition(p2pNodes.Get(0), 5, 15);
  AnimationInterface::SetConstantPosition(p2pNodes.Get(1), 10, 15);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(1), 15, 10);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(2), 20, 15);
  AnimationInterface::SetConstantPosition(csmaNodes.Get(3), 15, 20);
  anim.EnablePacketMetadata(true);

  // Create the ASCII trace file stream
  AsciiTraceHelper ascii;
  Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream("third.tr");

  // Install the trace sinks for all interfaces
  pointToPoint.EnableAsciiAll (stream);
  csma.EnableAsciiAll (stream);

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

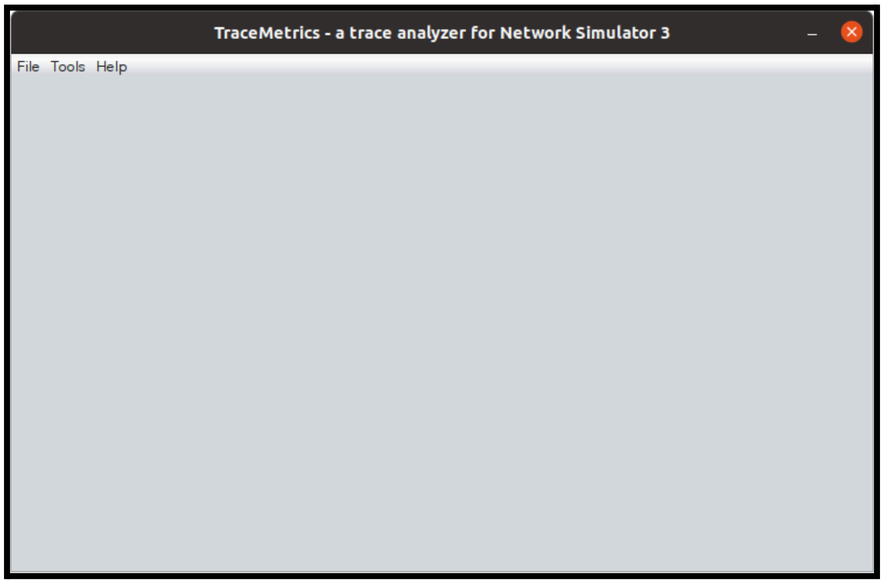
Command & Screenshot:

Step 1: To run the TraceMetrics Tool, use the following command.

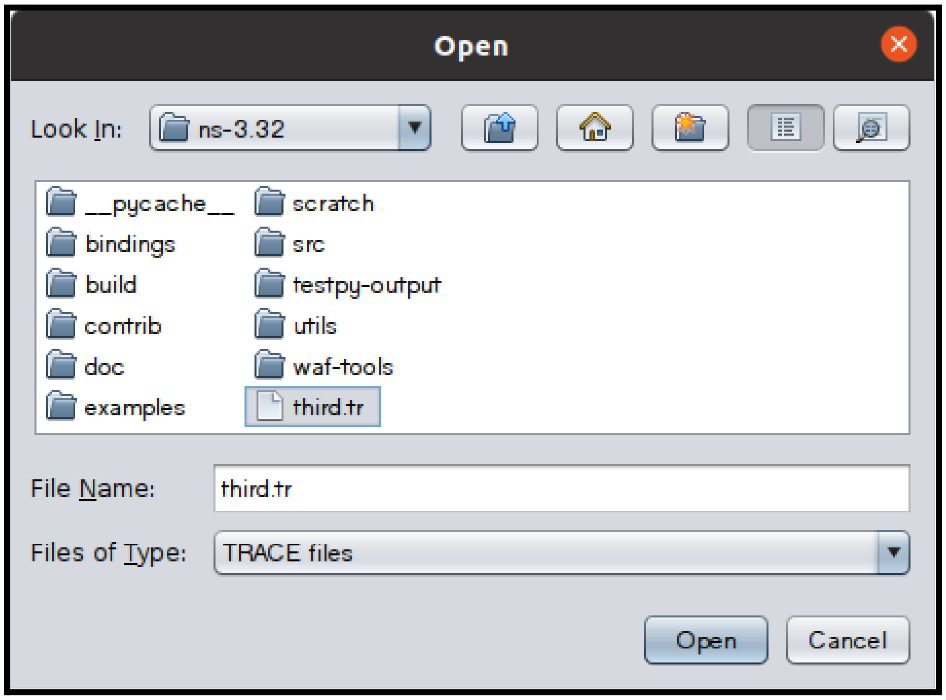
> **$ java -jar tracemetrics.jar**



```
iakashchoudhary@itsak-VirtualBox:~/Downloads/tracemetrics-1.4.0$ java -jar tracemetrics.jar
```



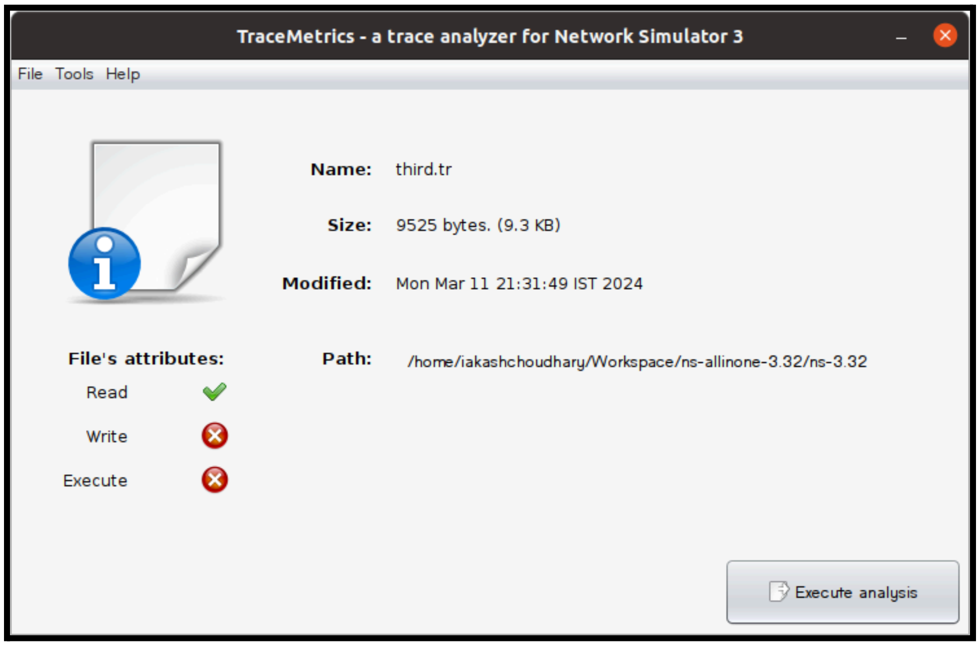
Step 2: Click on File. > Choose Open File. > Browse to the file location. > Click Open.



File Name: third.tr

Files of Type: TRACE files

Step 3: Now, click on the Execute analysis.



Name: third.tr

Size: 9525 bytes. (9.3 KB)

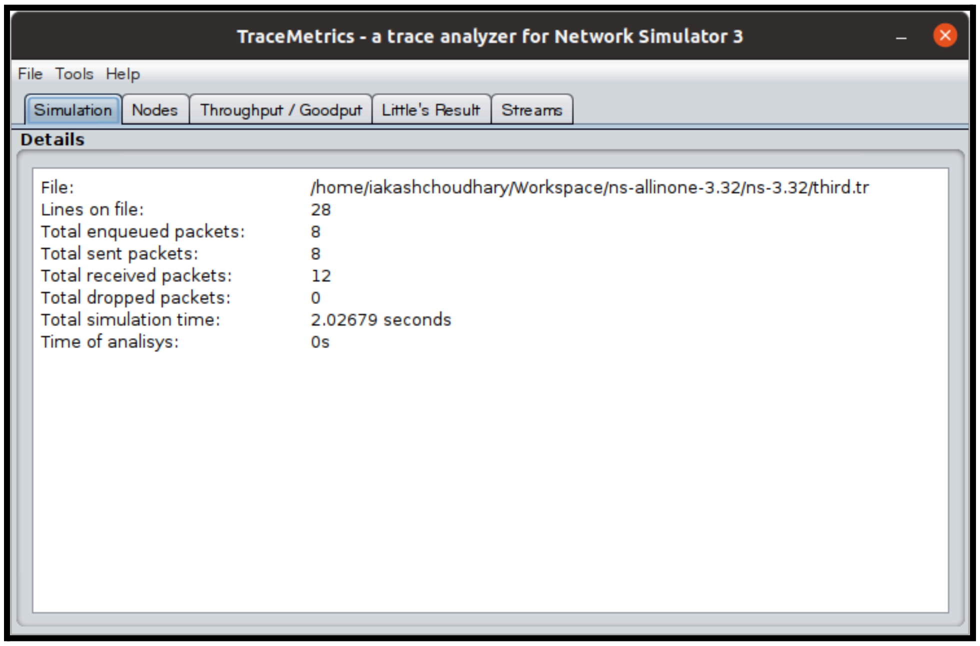
Modified: Mon Mar 11 21:31:49 IST 2024

Path: /home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32

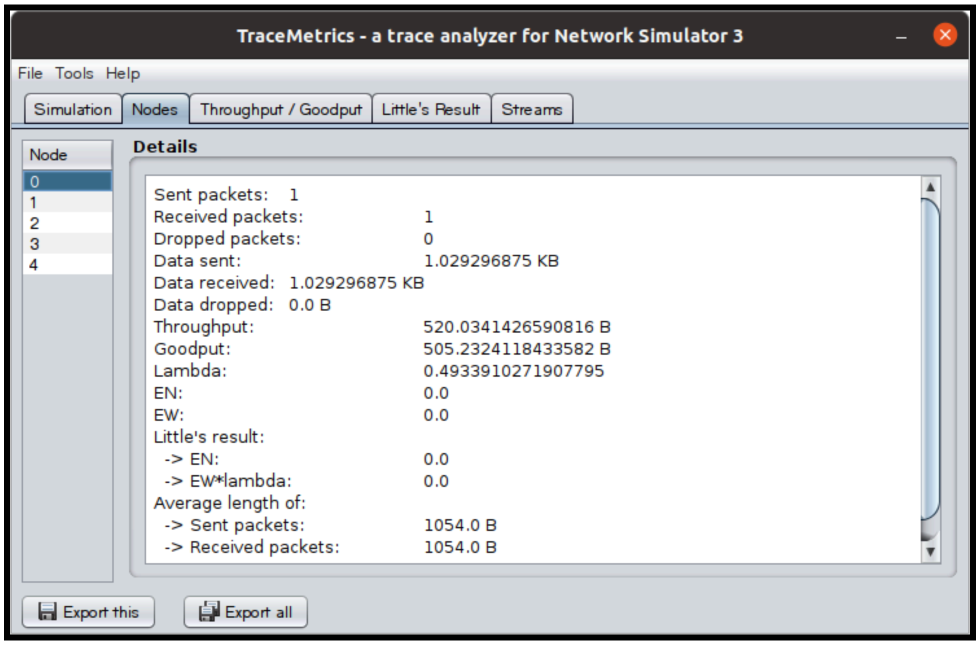
File's attributes:

Read	✓
Write	✗
Execute	✗

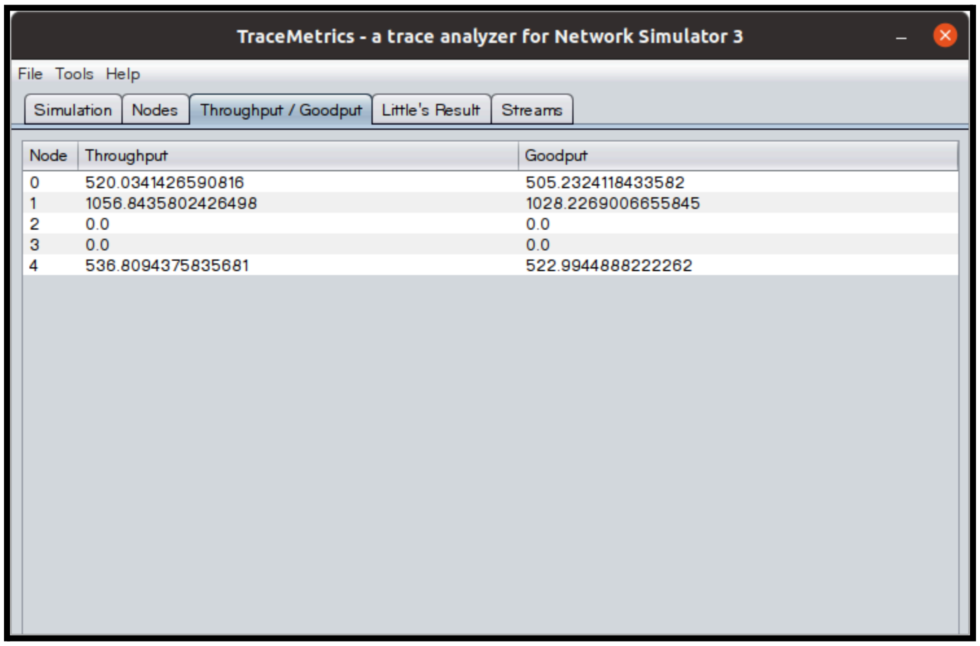
Step 4: Finally, you can check the trace metrics of the hybrid topology.



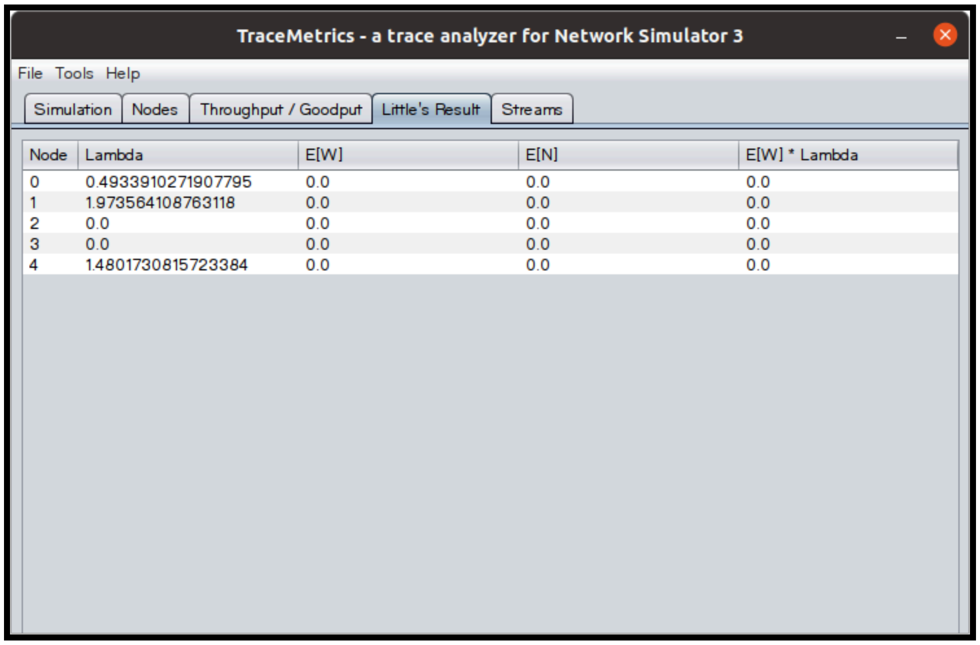
```
File:                      /home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/third.tr
Lines on file:             28
Total enqueued packets:    8
Total sent packets:        8
Total received packets:    12
Total dropped packets:     0
Total simulation time:     2.02679 seconds
Time of analisys:          0s
```



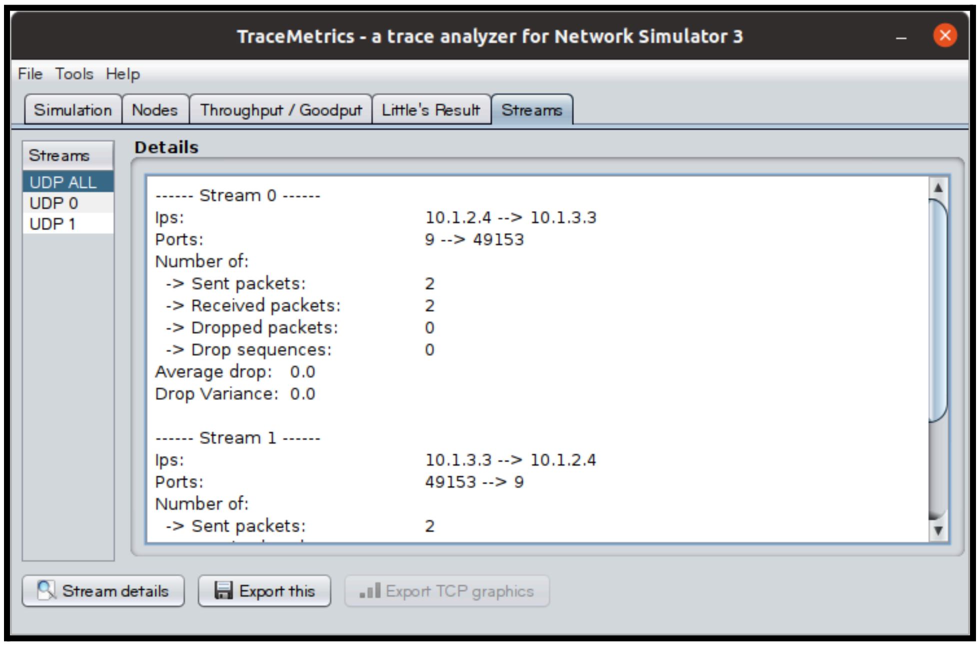
```
Sent packets:    1
Received packets:              1
Dropped packets:               0
Data sent:                     1.029296875 KB
Data received:   1.029296875 KB
Data dropped:    0.0 B
Throughput:                    520.0341426590816 B
Goodput:                       505.2324118433582 B
Lambda:                        0.4933910271907795
EN:                            0.0
EW:                            0.0
Little's result:
  -> EN:                       0.0
  -> EW*lambda:                0.0
Average length of:
  -> Sent packets:             1054.0 B
  -> Received packets:         1054.0 B
```



Node	Throughput	Goodput
0	520.0341426590816	505.2324118433582
1	1056.8435802426498	1028.2269006655845
2	0.0	0.0
3	0.0	0.0
4	536.8094375835681	522.9944888222262



Node	Lambda	E[W]	E[N]	E[W] * Lambda
0	0.4933910271907795	0.0	0.0	0.0
1	1.973564108763118	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	1.4801730815723384	0.0	0.0	0.0



```
------ Stream 0 ------
Ips:                        10.1.2.4 --> 10.1.3.3
Ports:                      9 --> 49153
Number of:
  -> Sent packets:          2
  -> Received packets:      2
  -> Dropped packets:       0
  -> Drop sequences:        0
Average drop:   0.0
Drop Variance:  0.0

------ Stream 1 ------
Ips:                        10.1.3.3 --> 10.1.2.4
Ports:                      49153 --> 9
Number of:
  -> Sent packets:          2
```

Conclusion: Creating a Trace file (.tr file) and analyzing it using TraceMetric Tool/tcpdump provides insights into network traffic patterns and allows for detailed examination of network communication.