1

INTRODUCTION TO NETWORK SIMULATION AND SNIFFING SOFTWARE

Unit Structure

1.0 Objectives

- Understand the use of network Simulation.
- Understand the use of sniffing software.
- You can install NS3 on Linux platform.
- You can install NetAnim.
- You can install Wireshark.
- You can run basic commands of Linux.

1.1 Introduction

Network Simulation is the software which tells the behaviour of computer networks. It is common useful method to calculate different network topologies exclusive of real-world implementation. Mean Network Simulation is method in computer network who analyse the performance of network entities, relation between network entities like nodes, Nswitched, Routers, Nodes, Access Points. In Simulator computer network can be molded with the help of links, devices, and applications. These are available using network with technologies Internet of things (IOT),5G, WLANs, Adhoc network of mobile, WSNs, LTE, Adhoc network of Vehicles etc.

Network Snipping Software is integral part of network analysis of capabilities in Network Performance Monitor (NPM). It is designed to provide inside into top metrics, allow admin to set dynamic alert and troubleshoot.

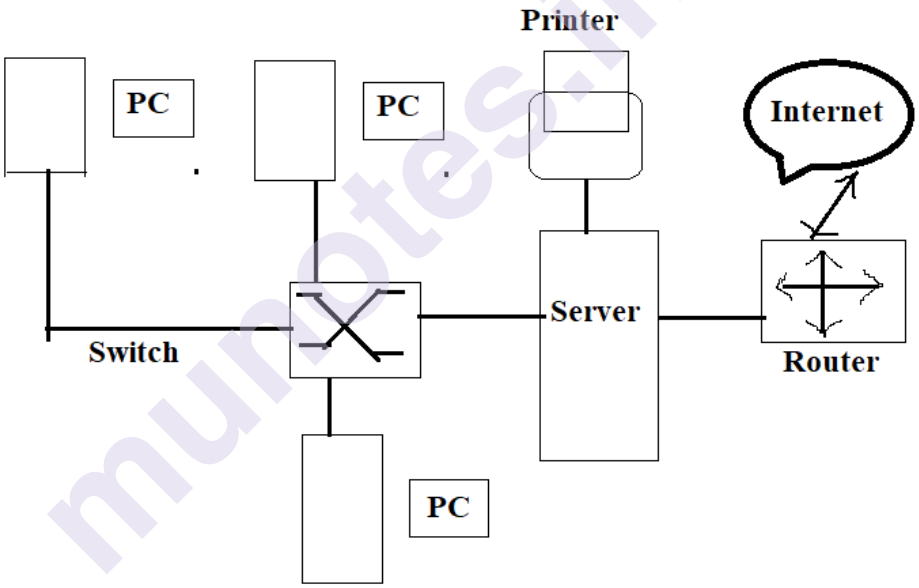
1.2 Network Simulation and sniffing software:

Network Simulation is method in computer network who analyse the performance of network entities, relation between network entities like nodes, N Switched, Routers, Nodes, Access Points.

By using Sniffing software, you can check the network traffic that is applicable for coming and going to that network. Kali Linux having different tools for sniffing.

1.2.1 NS3(Network Simulator):

Network Simulator uses mathematical formulas to create a theoretical and entirely virtual model of a network.



Network Simulation

There are different network simulator tools are available some are open source, and some are commercial.

- Network simulator Version (NS-2) (Open Source)
- NS-3(Open Source)
- Netkit (Open Source)
- Marionet (Open Source)
- JSIM (Java Based Simulation) (Open Source)
- OPNET (Commercial)
- QUALNET (Commercial)

1.2.2 Difference between NS2 and NS3

NS3 is a discrete event simulator which is like ns2.Difference between NS2 and Ns3 are language used by NS2(OTCL with C++) and NS3(C++ with optional Python).One more difference is that NS2 have default animator that is Nam where NS3 does not default animator but support NetAnim.

1.2.3 NS3 Installation

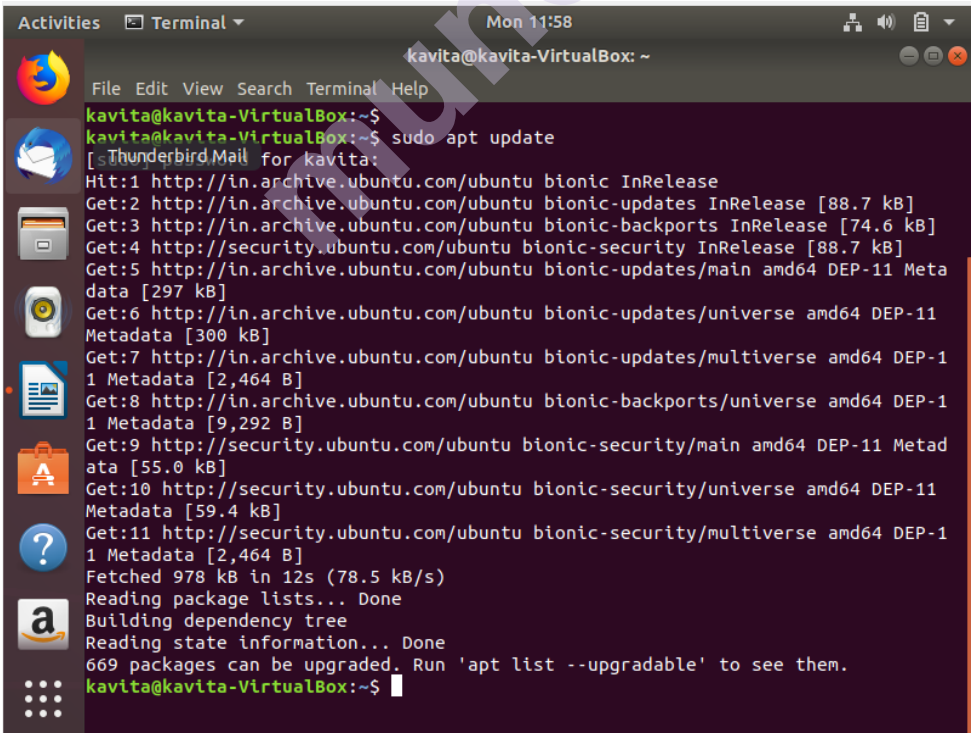
For NS3 installation on Ubuntu 1804 version.

We have to executve following commands

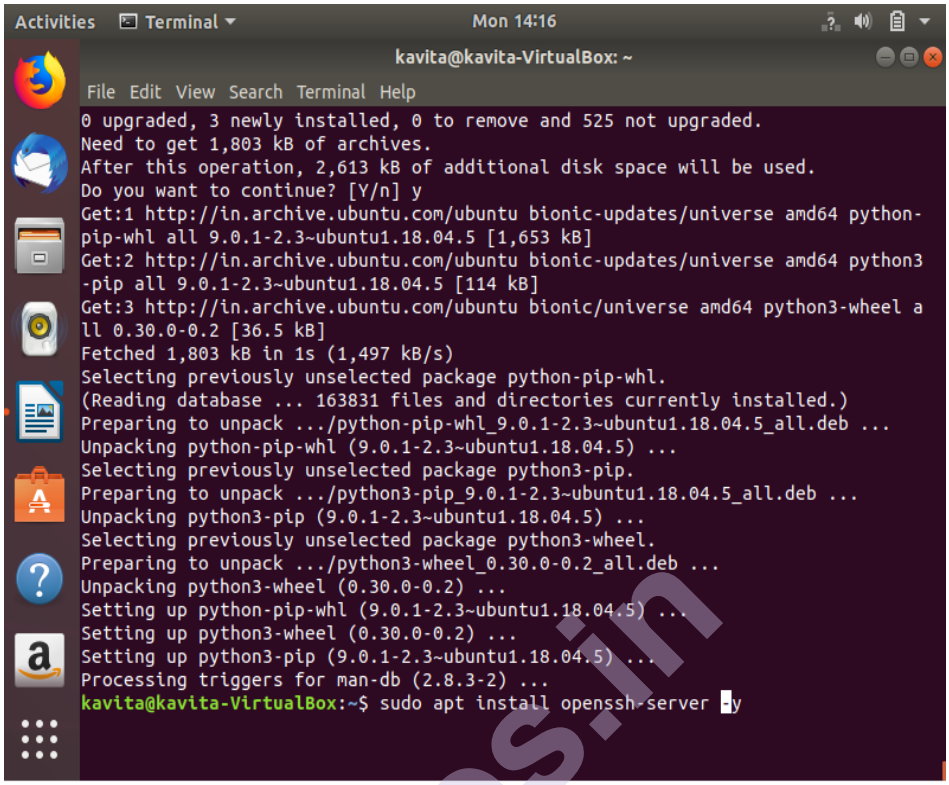
Step1: sudo apt update

Step2: sudo apt-get install gcc g++ python python3 python3-dev qt5-default python3-setuptools git mercurial gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython openmpi-bin openmpi-common openmpi-doc libopenmpi-dev autoconf cvs bzr unrar gdb valgrind uncrustify doxygen graphviz imagemagick python3-sphinx dia gsl-bin libgsl-dev libgsl23 libgslcblas0 tcpdump libxml2 libxml2-dev cmake libc6-dev libc6-dev-i386 libclang-6.0-dev llvm-6.0-dev automake python3-pip vtun lxc uml-utilities libboost-signals-dev libboost-filesystem-dev -y

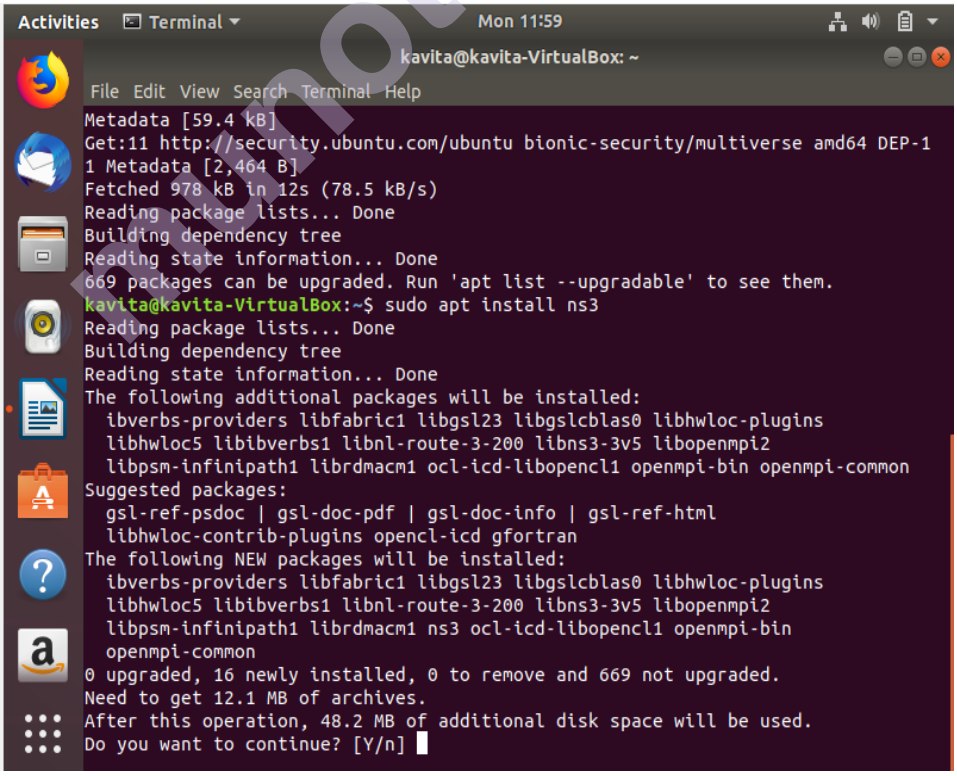
Step3: pip3 install --user cxxfilt

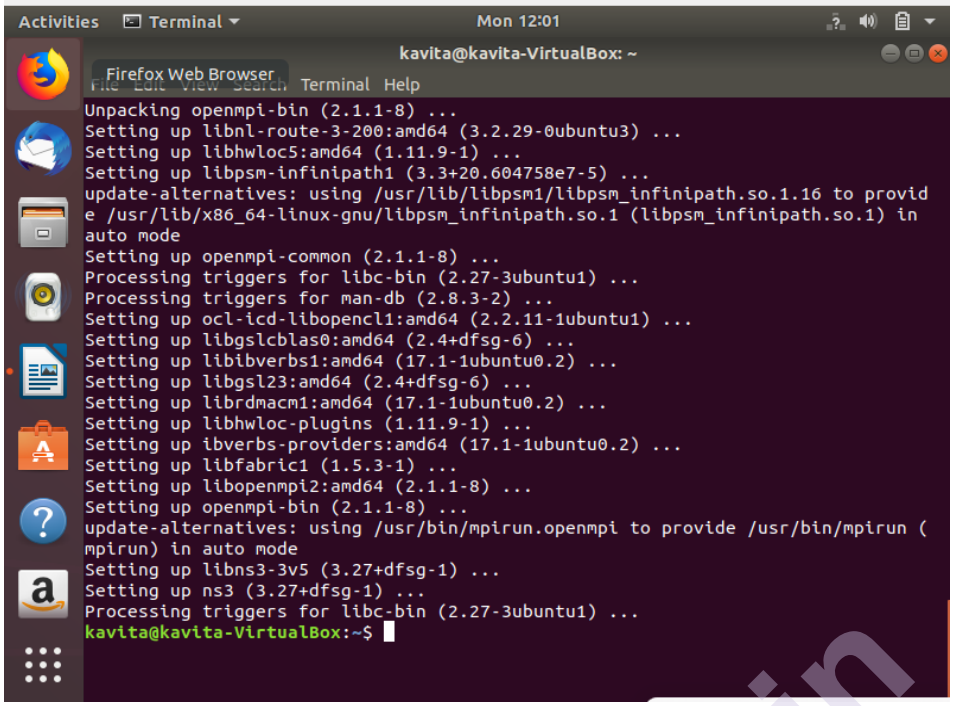


Step4: Enable ssh service in Ubuntu, you need to install openssh-server



Step5: Install NS3

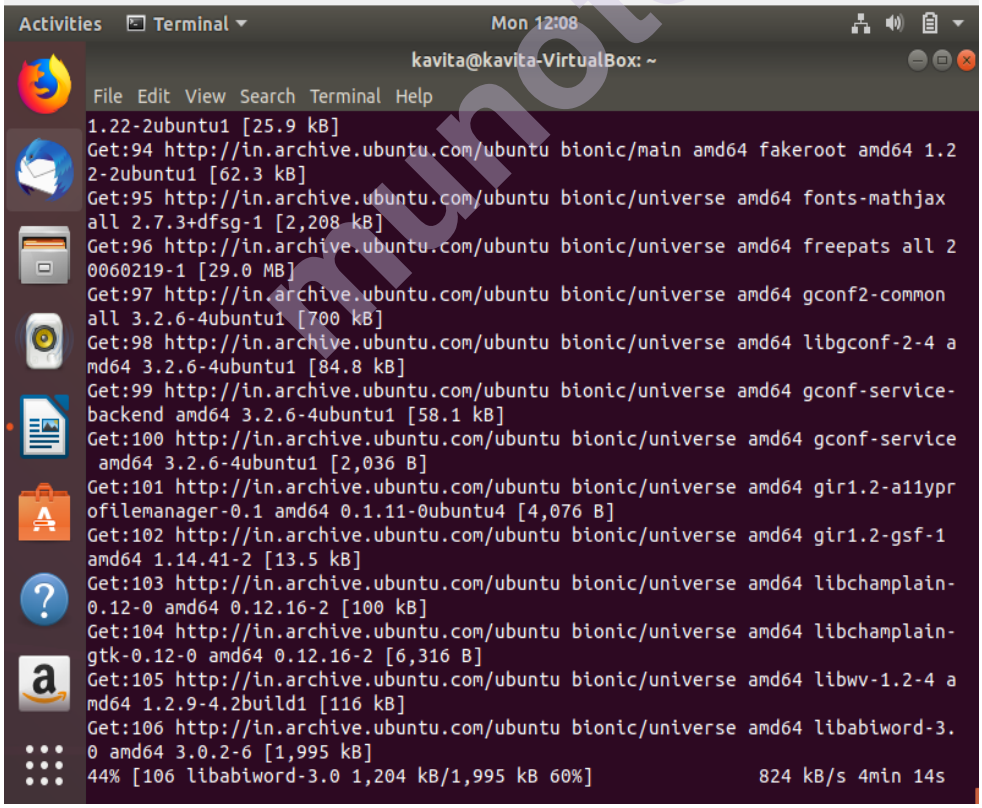


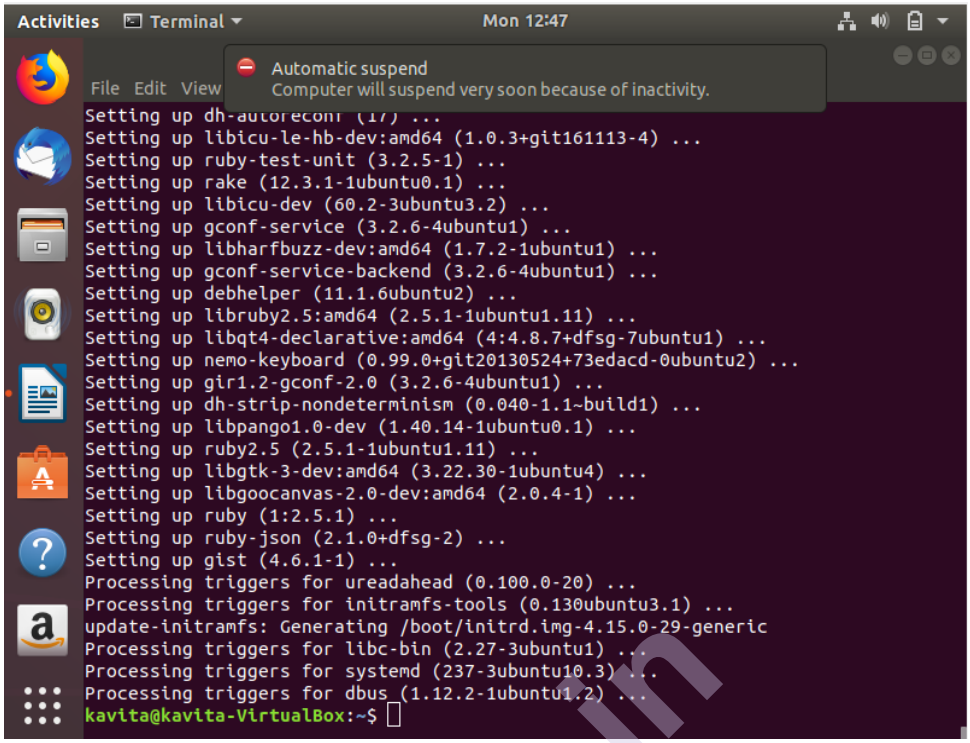


Step6: Download NS3 from https://www.nsnam.org and extract the folder.

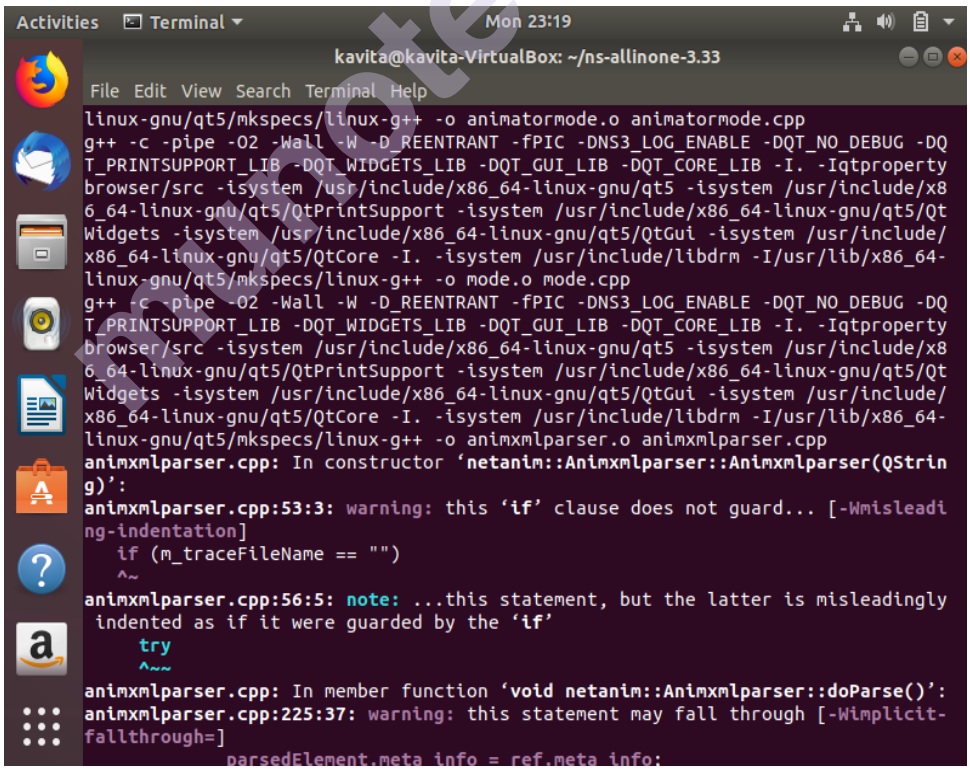
$ cd ns-allinone-3.30.1/

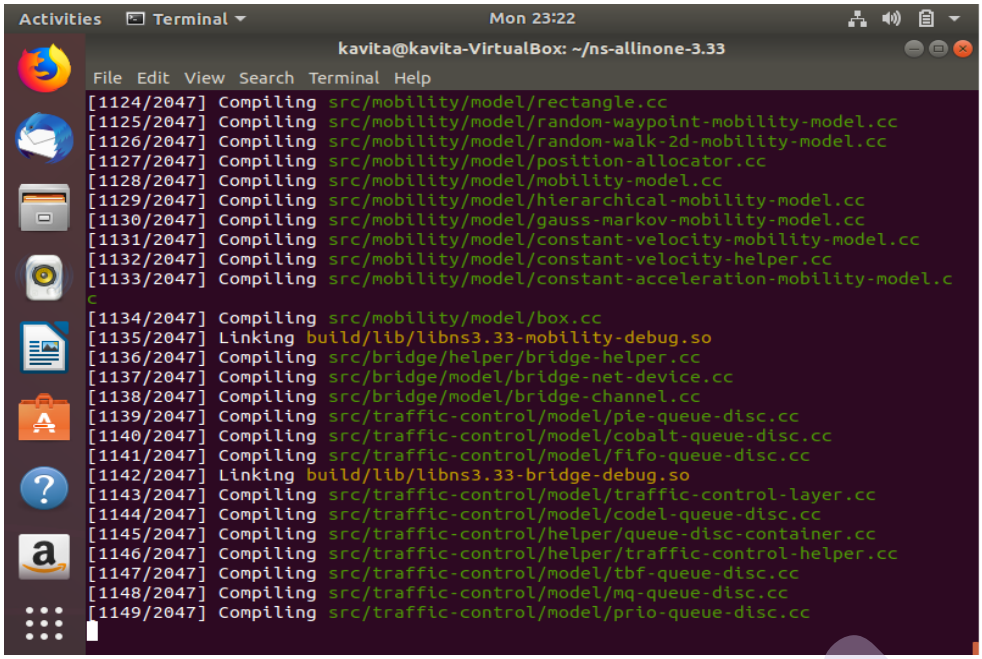
ns-allinone-3.30.1$ sudo ./build.py





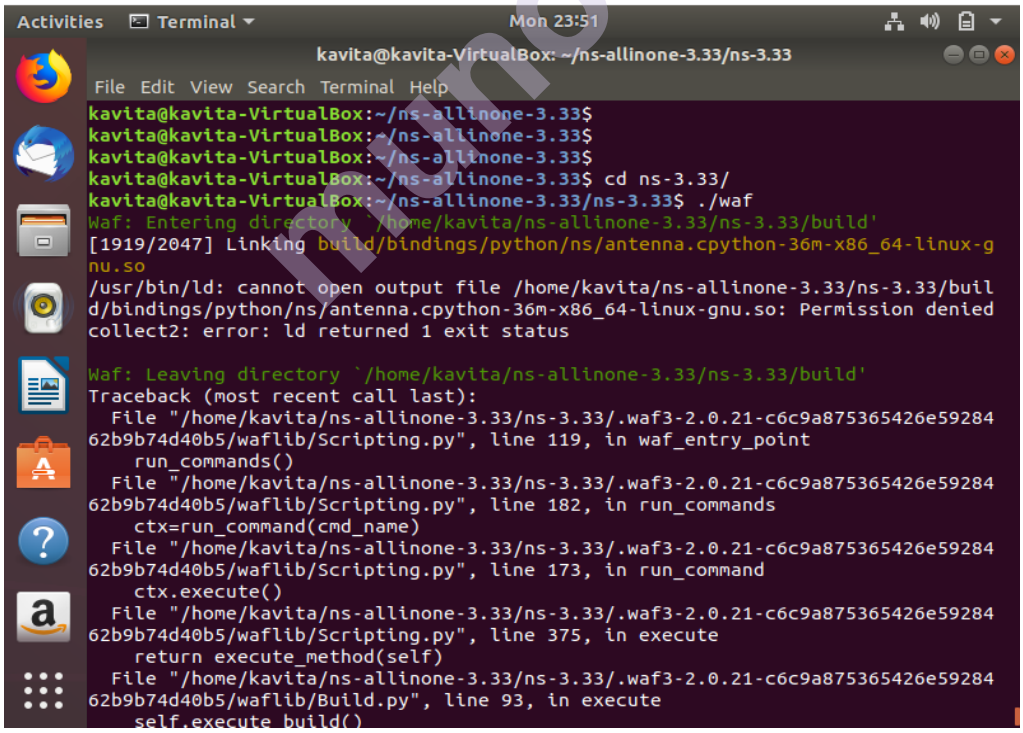
Step 7: Compiling messages and Compiling Process for this its take apex 30min





Step 8: Configure the NS3

ns-allinone-3.30.1$ cd ns-3.30.1/
ns-allinone-3.30.1/ns-3.30.1$ ./waf --build-profile=debug --enable-examples --enable-tests configure
ns-allinone-3.30.1/ns-3.30.1$ ./waf
ns-allinone-3.30.1/ns-3.30.1$ ./test.py -c core



Step9: ns-allinone-3.30.1/ns-3.30.1$ ./WAF

If we get **Hello simulator** then we installed NS3 successfully.

1.2.4 NetAnim

NetAnim is the network Animator that comes with NS3.During compilation of process of NS3 NetAnim may not be compiled .so we need to compile NetAnim.

It is an offline network animator tool that now comes with NS3 that ns-allinone3. All versions. By using NetAnim we can animate NS3 simulator, using the xml file trace the output in the simulation.

NetAnim is the software which execute xml file to generate graphical output on NS3 simulator.

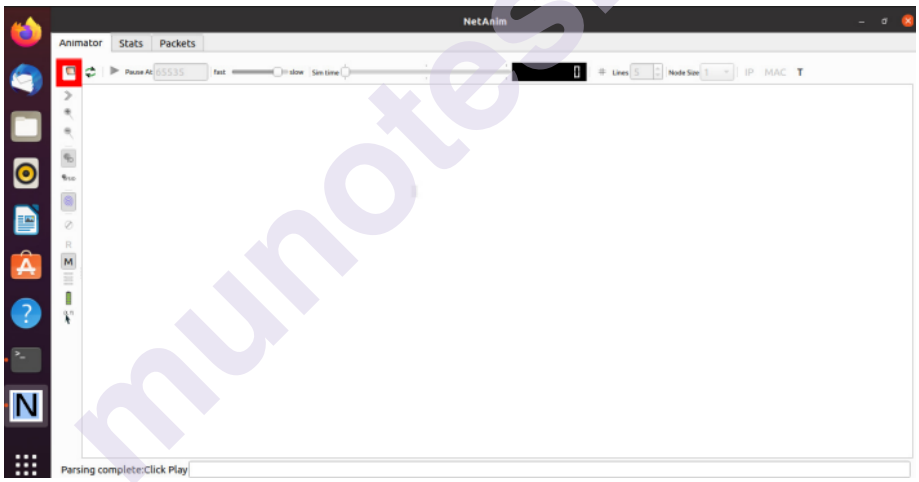
1.2.5 Installation of NetAnim

You can directly install NetAnim

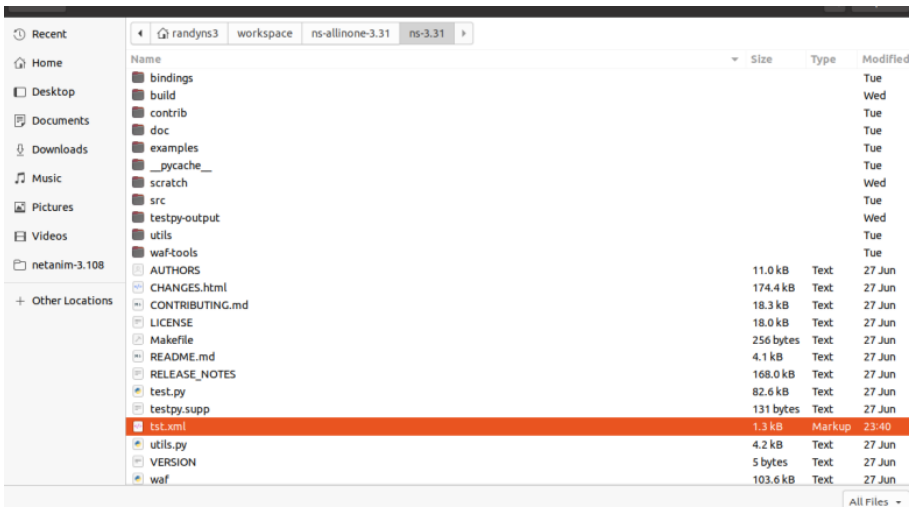
Otherwise, you have to execute some commands but for this we need NS3 installed or compiled.

Step1: sudo apt-get install NetAnim

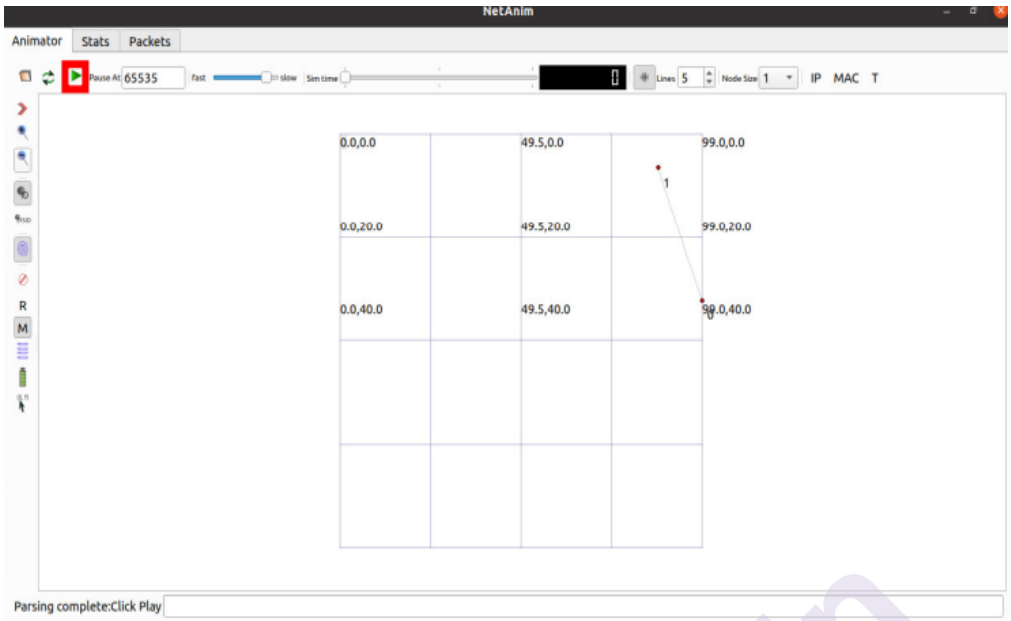
Step2: NetAnim file.xml



Step3: Select Xml File



Step4: Run the simulation by clicking



1.2 6 Wireshark:

It is the network protocol for analayzing freely available packages.Wireshark is network packet analyzer.It is used to check incoming and outgoing packets in the network and save it offline analysis.It works on Windows,linux,macOS,FreeBSD etc.

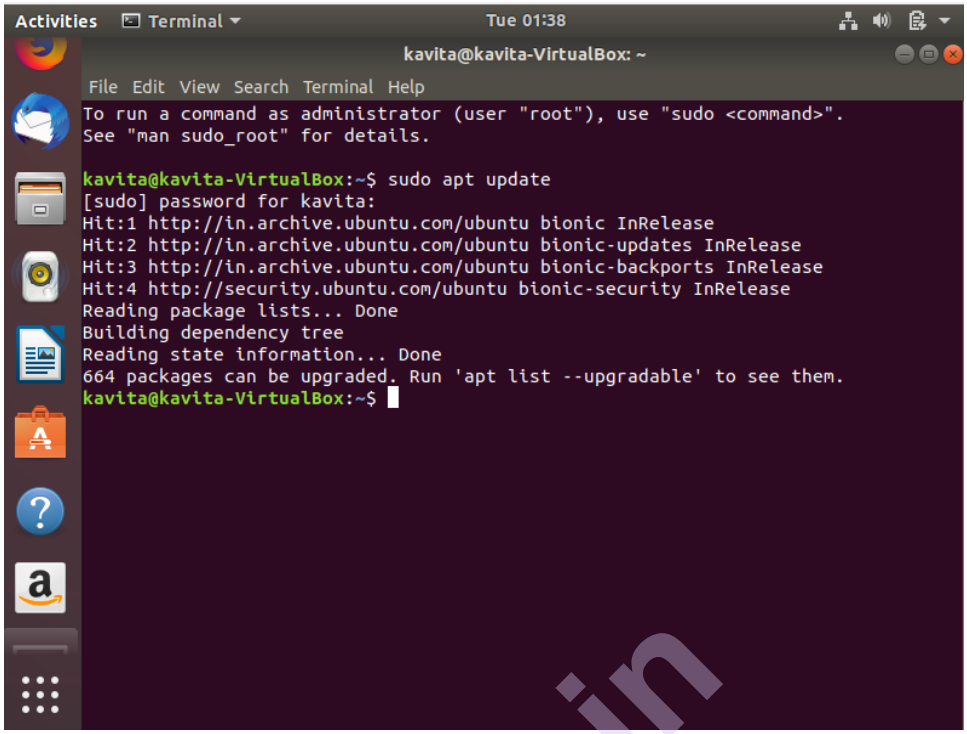
It is open source packet analyzer.

1.2.7 Features of Wreshark:

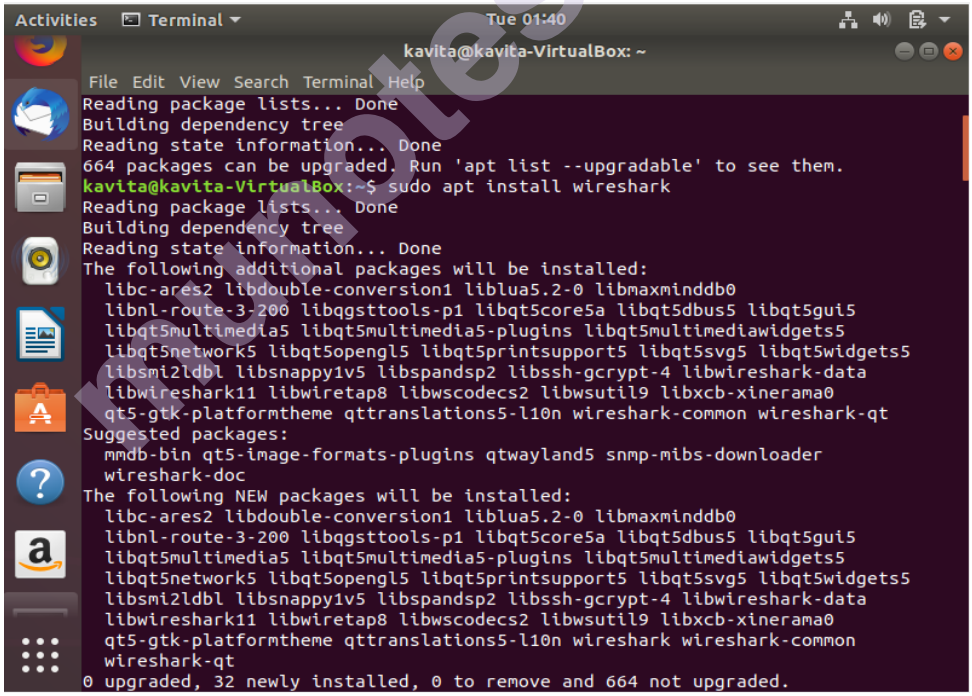
1) Live capture or offline analyze the packets.
2) It runs on multiplatform like Windows,Linux,MacOS,FreeBSD etc.
3) It's used in industry and education.
4) Many different Read/Write capture file format.
5) Colouring the packets for fast analysis.
6) Insection of hundreads of different protocols.
7) Analyze VOIP protocol.
8) Result can be saved in XML,CSV,Postscript and Plain Text document.
9) Three way handshake
10) It Performance troubleshooting dropped packets and problems.

1.2.8. Installation of Wireshark:

Step 1: Update the system by using sudo apt update

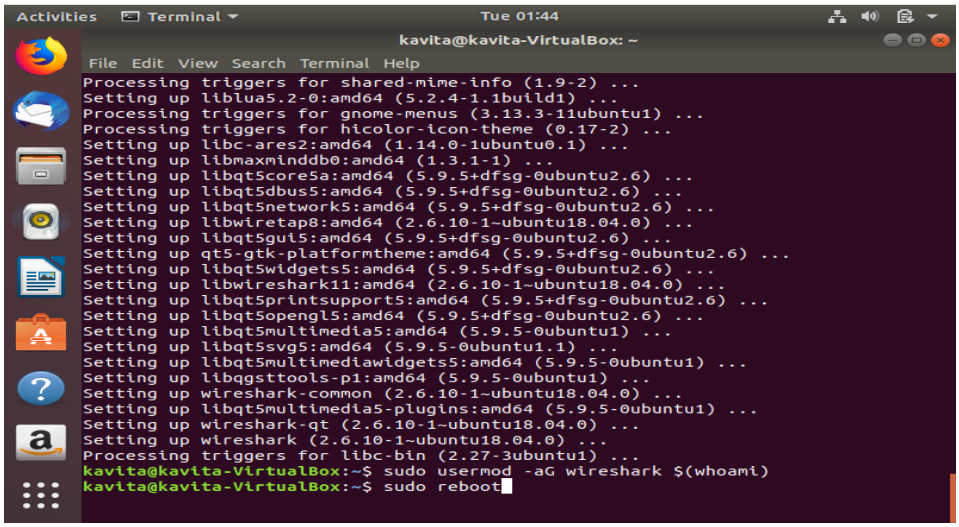


Step2: Install wireshark using sudo apt install wireshark

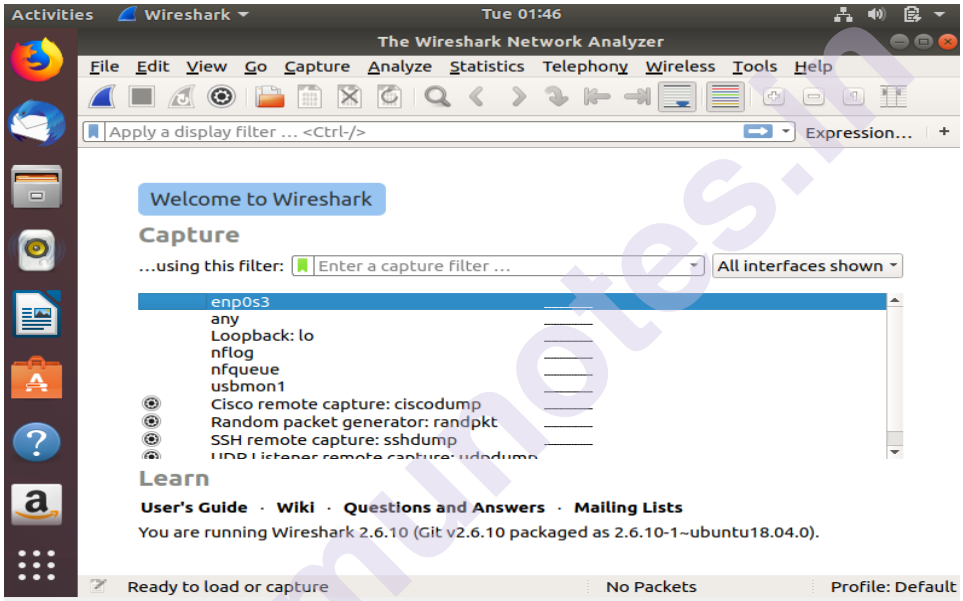


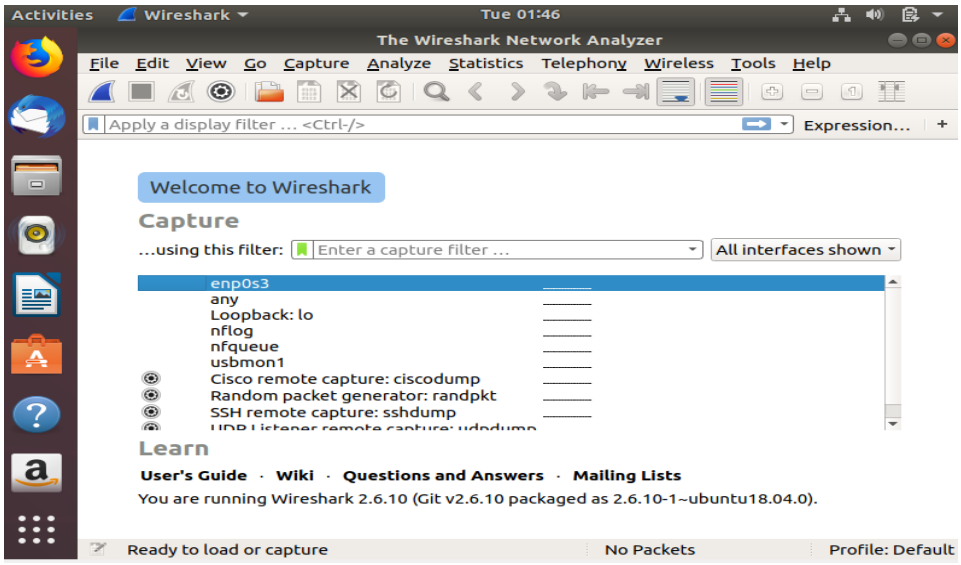
Step3: Add user in wireshark by using usermod -aG wireshark $(whoami)

Reboot the system-sudo reboot



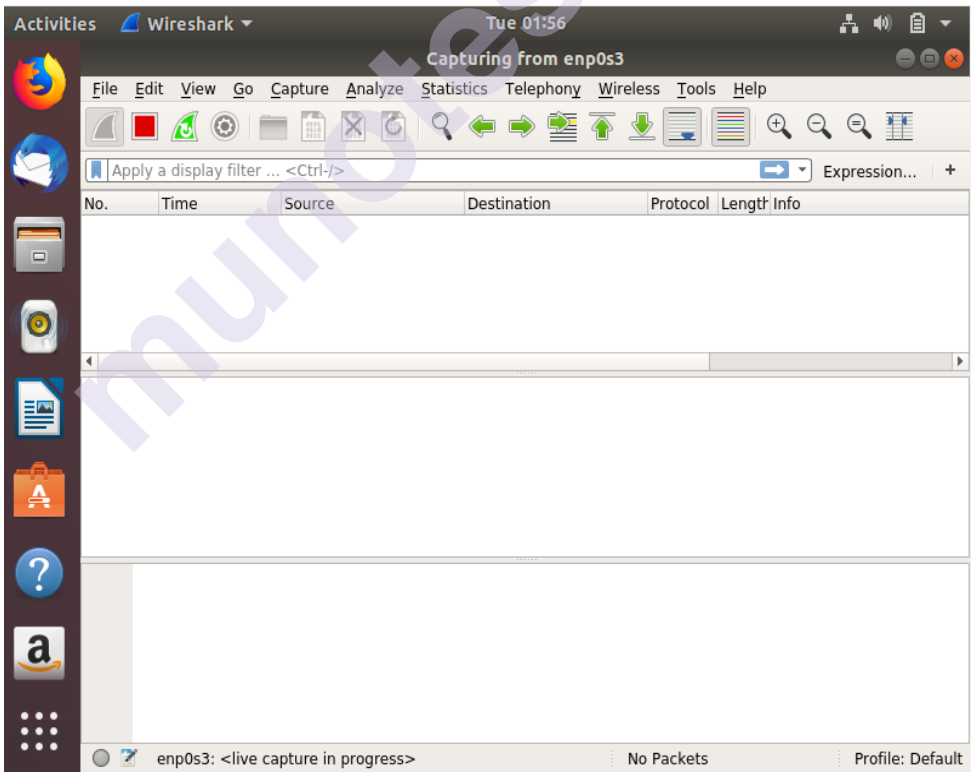
Step4: To run Wireshark type the command (CI) on terminal Wireshark



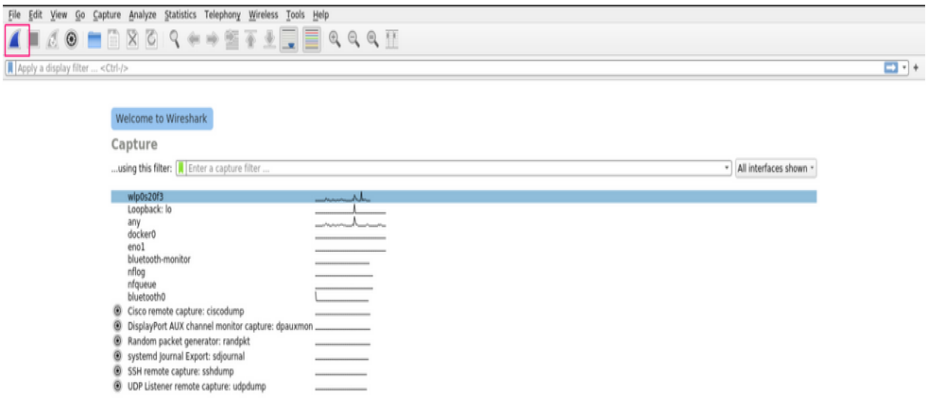


Step5: To run Wireshark using GUI go to application and then Wireshark

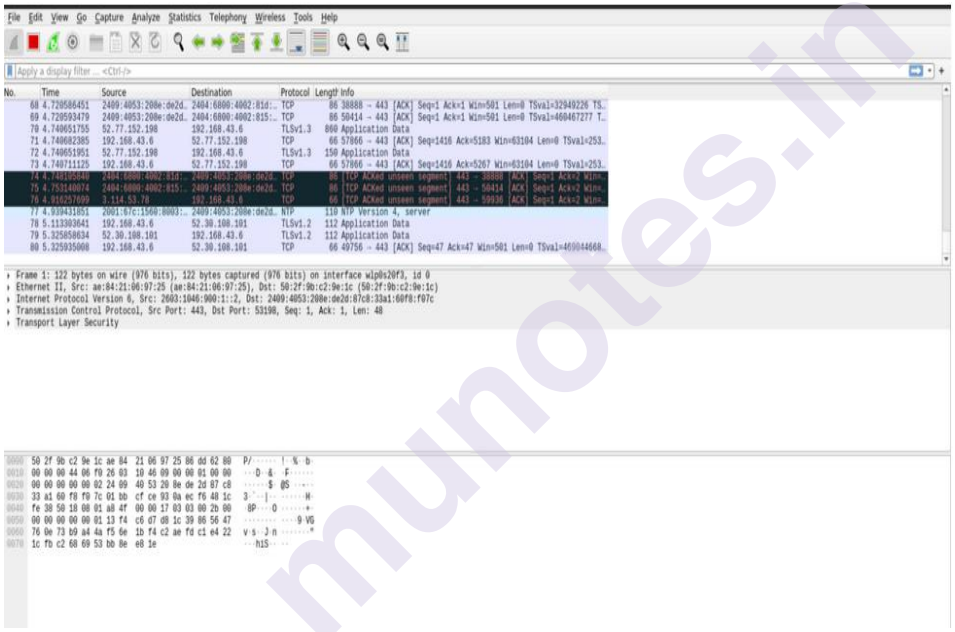
Step 6: Capture packets for analyze from menu File-Capture



Step7: Select any interface



Step 8: Captured Packet List



1.3 Summary:

In this chapter, we learned the NS3 Simulator. There were versions of NS2 and NS3 having differentiated between them regarding the languages used. NS3 is an open-source simulator which is used check performance of the network. We must follow some steps while installing NS3 Ubuntu18.0. 4version .NetAnim is the network Animator that comes with NS3. It is an offline network animator tool that now comes with NS3 that ns-allinone3. NetAnim is the software which execute xml file to generate graphical output on NS3 simulator. We must follow some steps while installing NetAnim in Ubuntu18.0. 4version. Wireshark is open-source packet. It is the network protocol for analyzing freely available packages. Wireshark is a network packet analyzer. While installing Wireshark we have to install Wireshark. You can run by using command Interface (CI) or graphical interface (GUI).

1.4 Reference for further reading

1) manual.pdf (nsnam.org)
2) https://linuxhint.com/install_configure_wireshark_ubuntu/
3) Download | ns-3 (nsnam.org)
4) Wireshark · Download
5) https://www.nsnam.com/2014/08/installing-netanim-software-for-ns3-in.html
6) https://linuxhint.com/install_wireshark_ubuntu/
7) Java Network Programming, Third Edition, by Elliotte Rusty Harold.Oreily Pub

Self-Learning Topics:

Linux Operating System Commands for installation

Question:

1) What is the use of NS3?
2) Installation of NS-3 in Linux
3) What is the use NetAnim?
4) Installation of NetAnim in Linux.
5) What is mean by Wireshark. Explain the features Wireshark. Installation of Wireshark in Linux.

❄❄❄❄❄❄❄

2

CLIENT SERVER NETWORK TOPOLOGY USING NS-3

Unit Structure

2.0 Objective

After going through this unit, you will be able to:

- Design and Construct various (simple, different and complex) network topologies using Network Simulation tool (NS 3)
- Simulate socket programming in JAVA for client-server networks with different protocols such as TCP,UDP,FTP.

2.1 Introduction

A network simulator is a tool that allows you to simulate a real-world network on a single computer using C++ or Python scripts.

Normally, if we want to see how our network functions with various parameters, we would conduct tests.

We lack the necessary number of computers and routers to create various topologies. Even if we had these resources, constructing a network for experiment purposes is prohibitively expensive.

To get around these limitations, we employed NS3, a discrete event network emulator for the Internet.

NS3 allows us to establish various virtual nodes (i.e., computers in real life) and install devices, internet stacks, applications, and other items to our nodes using various Helper classes.

We can use NS3 to construct Point-to-Point, Wireless, CSMA, and other types of connections between nodes.

A point-to-point link is the same as a local area network (LAN) connection between two computers. Wireless connections between PCs and routers are the same as WiFi connections. The topology of a CSMA connection is the same as that of a computer bus. After establishing connections, we attempt to install a network interface card (NIC) on each node to enable network communication.

When network cards are activated in devices, we add data rate, packet size, and other parameters to the channels (i.e., the real-world way used to transport data).

We are now employing programmes to generate traffic and transfer packets.

2.2 An Overview of NS-3

- Available for Linux, OS X, and Windows "with Cygwin tools" as a free and open-source discrete event network emulator.
- Written in C++, with a Python interface for visualisation and scripting available as an option.
- Helpers built-in to make "scripting" in C++ simple.

Ns3 provides us with unique characteristics that can be applied to real-world integrations.

The following are some of these characteristics:

- **Node tracing:** NS3 allows us to follow the paths of the nodes, which allows us to see how much data is sent and received. To keep track of these operations, trace files are created.
- **NetAnim:** It stands for Network Animator. It's an animated representation of how the network would look in real life, as well as how data will be transported from one node to the next.
- **Pcap file:** NS3 assists in the creation of a pcap file that may be used to obtain all packet information (e.g., Sequence number, Source IP,

destination IP, etc). Wireshark, a software programme, can be used to view these pcaps.

- **Gnuplot:** GnuPlot is a programme that plots graphs from data obtained from the NS3 trace file. In comparison to other graphing tools, Gnuplot produces better accurate graphs and is also less difficult.

This is a quick overview of NS3.

2.2.1 What is a system?

The ns-3 simulation has the same conceptual structure as a real network.

Simulation Structure :

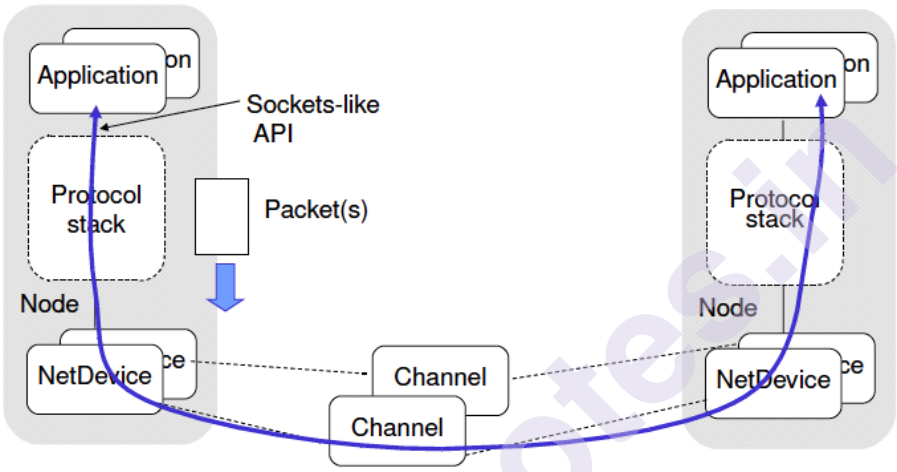


Figure 1 : Structure of NS-3 simulation process

- Comprehensive documentation, including a handbook, tutorial, and numerous examples.
- Supports different network layers:
  - Applications: On/Off, Bulk transfer, HTTP, etc.
  - Transport: TCP, UDP.
  - Network: IPv4, IPv6, routing.
  - Physical: Ethernet, PPP, 802.11, LTE, mobility models.

Steps in Simulation:

a) Define what you want to simulate;

b) Define your simulation topology (nodes, channels, network interfaces, network stack, and applications);

c) Use a text editor to create the simulation script;

d) Run your script with WAF ($./waf –run ScriptName);

e) Examine the output traces.

2.2.2 Features of the application:

- Program Helper Name: the sort of app to use (BulkSendHelper, OnOffHelper, DashClientHelper, DashServerHelper, and so on).
- Transport protocol : ns3::TcpSocket / Udp...Helper.
- Remote: the receiver app's address.
- App. Start: the time at which the app should begin.
- App. Stop: the time when the app should be closed.
- SendSize: specifies the size of the programme segment to send.
- MaxBytes: the total number of bytes that the App will generate.

2.2.3 TCP Socket attributes:

- TcpL4Protocol:: SocketType: type of TCP congestion control (TcpWave, TcpReno, TcpNewReno, TcpWestwood, TcpTahoe).
- TcpSocket::SndBufSize / TcpSocket::RcvBufSize sender/receiver buffer size (bytes).
- InitialCwnd: the number of segments to send in the first window.
- Parameters for ACKs: specify the options for ACKs (i.e. delayed ACK count, SACK, etc.).
- MSS: maximum TCP segment size allowed.

Some TCP variants (such as TCPWave) have extra features (i.e. Beta, TxTimer)

2.2.4 IP layer attributes:

- SetBase: network base address to start a range for IP addressing.
- NewNetwork: start addressing a new network.
- Assign: assign an IP address to a network interface.
- Populate Routing Tables: enable a global routing between different networks.

2.2.5 Attributes of the NetDevice and the Channel:

- DataRate: the network device's transmission rate (bandwidth).
- Delay: the communication route between two nodes experiences a physical speed-of-light delay.
- DropTailQueue: A FIFO packet queue that drops tail-end packets when the queue reaches capacity.
- ErrorModel: used to signal that the receiver network device should consider a packet/bit to be errored based on a certain error model.

2.3 Program to create simple topology

The arrangement in which computer systems or network devices are connected is referred to as a network topology. Both the physical and

conceptual aspects of a network can be defined by topologies. In the same network, the logical and physical topologies could be the same or distinct.

2.3.1 Write a program to simulate traffic between two nodes

Point-to-point networks have exactly two hosts, such as computers, switches, routers, and servers, which are connected back to back via a single cable. Frequently, one host's receiving end is connected to the other's sending end, and vice versa.

Decide the requirement for simple network - number of nodes. Type of channel, few protocols(set of rules)

Default Network Topology

192.168.1.0
n0 ---------- n1
point-to-point
Program : Made changes in ns-3.31/examples/tutorials/first.cc file

```
// Set of libraries
#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"
//Define a namespace (due to chances of same name so after declaring the function call not //required eveytime)

using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int
main (int argc, char *argv[])
{
  CommandLine cmd (__FILE__);
  cmd.Parse (argc, argv);

//Take logs (getting information about previous communication of same network)
  Time::SetResolution (Time::NS); //Total time application is going to consumed
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

// Take n number of computers
  NodeContainer nodes;//Object from NodeContainer class with object name nodes
```

```
  nodes.Create (2); //simple point to point communication, requried only 2
nodes

//Choose the way of communication
  PointToPointHelper pointToPoint;// class with type of channel
//Setting up device parameters
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
//Setting up channel parameters (There are multiple parameters)
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

//Install the way of communication on nodes
  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

//Asking to follow rules (Install internet rules on nodes)
  InternetStackHelper stack;
  stack.Install (nodes);

//Assign IP address to communicate
  Ipv4AddressHelper address;   // can be used IPv6 too
  address.SetBase ("192.168.1.0", "255.255.255.0"); //(IP address, subnet
mask)
// Assign addresses whatever nodes present in network
 Ipv4InterfaceContainer interfaces = address.Assign (devices);

// create a x type of server on port x
  UdpEchoServerHelper echoServer (15);  //object=echoserver with port 9

// Install a node as server
  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0)); //start server
  serverApps.Stop (Seconds (10.0)); //stop server

// create a x type of client and set its attributes
// client communicate with port number 9
  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

// Install another node as client
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0)); //start client
  clientApps.Stop (Seconds (10.0)); //stop client
```

```
//Run the simulation
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

Command to run a program :

user@user-Lenovo-G50-80:~/Desktop/ns-allinone-3.31/ns-3.31$ ./waf --run scratch/first

Output :

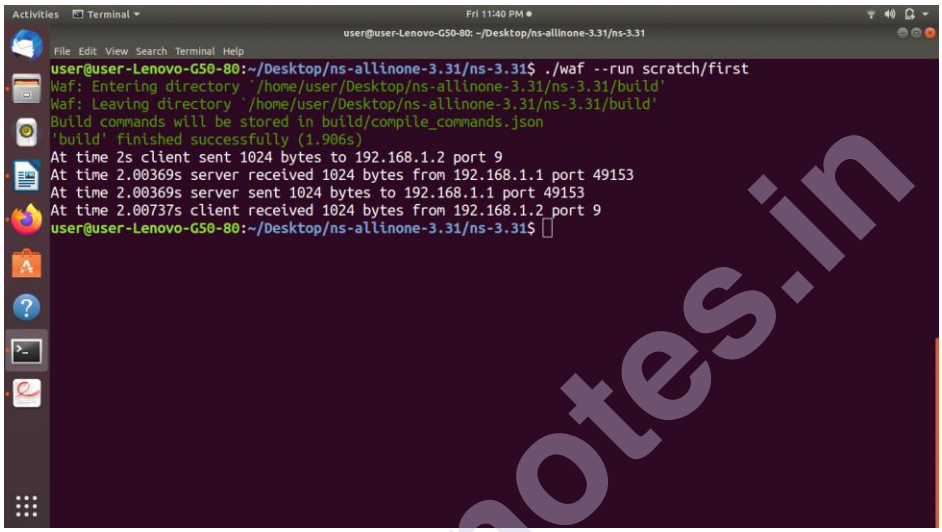


Figure 2 : Output 1 with port 9

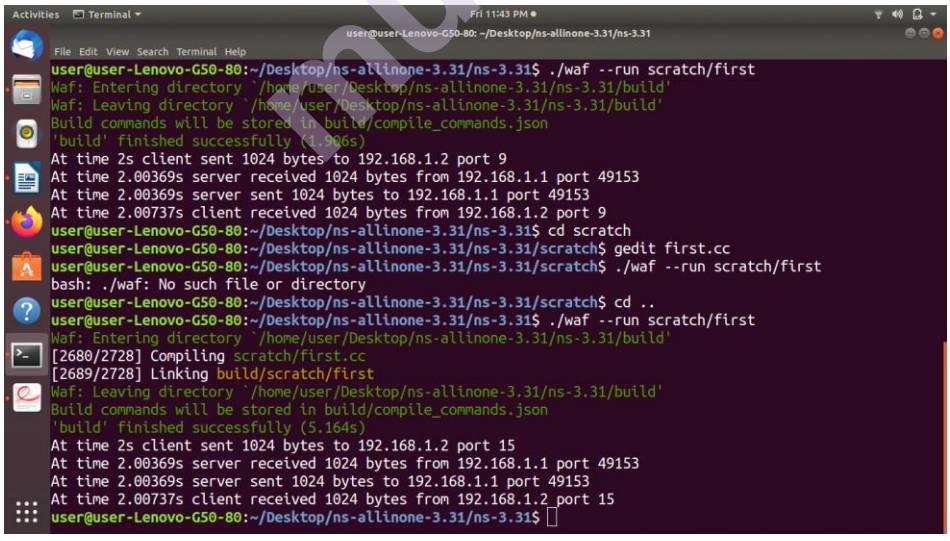


Figure 3-Output2 with port 15

Result : It shows communication has been established between two nodes. Client sending 1024 bytes to server on port 9, as shown figure 2. We changed port number to 15 and again shown as figure 3.

2.4 Programs to different types of topologies

2.4.1 Write a program to simulate star topology.

In a star topology, all the devices are connected to a central device. This device will then control all the data traffic flow within the entire network. A good example of such a topology is a home wireless network, where all the desktops, laptops, tablets, printers, and smartphones are connected to a single wireless router.

Star Network topology :

```
      n2 n3 n4
         \ | /
          \|/
   n1--- n0---n5
          /|\
         / | \
       n8 n7 n6
```

Program : It is from star.cc[2]

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/netanim-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-layout-module.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("Star");
int
main (int argc, char *argv[])
{

// Set up some default values for the simulation.
Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue
(137));

// try and stick 15kb/s into the data rate
  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue
("14kb/s"));

// Default number of nodes in the star.  Overridable by command line
argument.
uint32_t nSpokes = 8;
```

```
 CommandLine cmd (__FILE__);
 cmd.AddValue ("nSpokes", "Number of nodes to place in the star",
nSpokes);
cmd.Parse (argc, argv);

NS_LOG_INFO ("Build star topology.");
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
PointToPointStarHelper star (nSpokes, pointToPoint);

NS_LOG_INFO ("Install internet stack on all nodes.");
InternetStackHelper internet;
star.InstallStack (internet);

NS_LOG_INFO ("Assign IP Addresses.");
star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0",
"255.255.255.0"));

NS_LOG_INFO ("Create applications.");
// Create a packet sink on the star "hub" to receive packets.
uint16_t port = 50000;

Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (),
port));
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
hubLocalAddress);
ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub
());
hubApp.Start (Seconds (1.0));
hubApp.Stop (Seconds (10.0));

// Create OnOff applications to send TCP to the hub, one on each spoke
node.
OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
onOffHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));

ApplicationContainer spokeApps;
for (uint32_t i = 0; i < star.SpokeCount (); ++i)
{
AddressValue remoteAddress (InetSocketAddress
(star.GetHubIpv4Address (i), port));
onOffHelper.SetAttribute ("Remote", remoteAddress);
spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
```

```
}
spokeApps.Start (Seconds (1.0));
spokeApps.Stop (Seconds (10.0));

NS_LOG_INFO ("Enable static global routing.");
// Turn on global static routing so we can actually be routed across the
star.
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

NS_LOG_INFO ("Enable pcap tracing.");
// Do pcap tracing on all point-to-point devices on all nodes.
pointToPoint.EnablePcapAll ("star");

NS_LOG_INFO ("Run Simulation.");
Simulator::Run ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");
return 0;
}
```

Output by following commands :

i) ~/Desktop/ns-allinone-3.31/ns-3.31$ **./waf --run scratch/star**
Build finished successfully

ii) ~/Desktop/ns-allinone-3.31/ns-3.31$ **ls**

iii) ~/Desktop/ns-allinone-3.31/ns-3.31$**tcpdump -nn -tt -r tcp-star-server-8-1.pcap**

tcpdump is a most powerful and widely used command-line packets sniffer or package analyzer tool which is used to capture or filter TCP/IP packets that are received or transferred over a network on a specific interface.

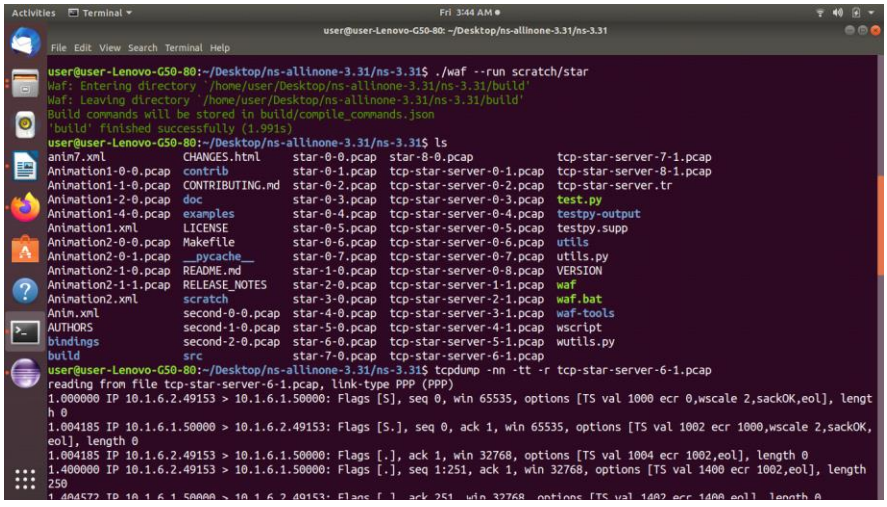


Figure 4 : Output of Star topology as packet sniffer

Result : The output shown as in figure 4, in the form of packet transfer from node 6 to node 1.

2.4.2 Write a program to simulate bus topology

All devices in a Bus topology share a single communication line or cable. When numerous hosts are sending data at the same time, the bus topology may cause problems.
As a result, Bus topology either employs CSMA/CD technology or assigns one host the role of Bus Master.

Bus Network Topology -

194.15.1.0
n0 ---------- n1 n2 n3 n4
point-to-point | | | |
==========
LAN(BUS) 194.15.2.0

Program : Made changes in 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"
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 ("194.15.1.0", "255.255.255.0");
 Ipv4InterfaceContainer p2pInterfaces;
 p2pInterfaces = address.Assign (p2pDevices);

 address.SetBase ("194.15.2.0", "255.255.255.0");
 Ipv4InterfaceContainer csmaInterfaces;
 csmaInterfaces = address.Assign (csmaDevices);

 UdpEchoServerHelper echoServer (15);
```

```
  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get
(nCsma));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma),
15);
  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);

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

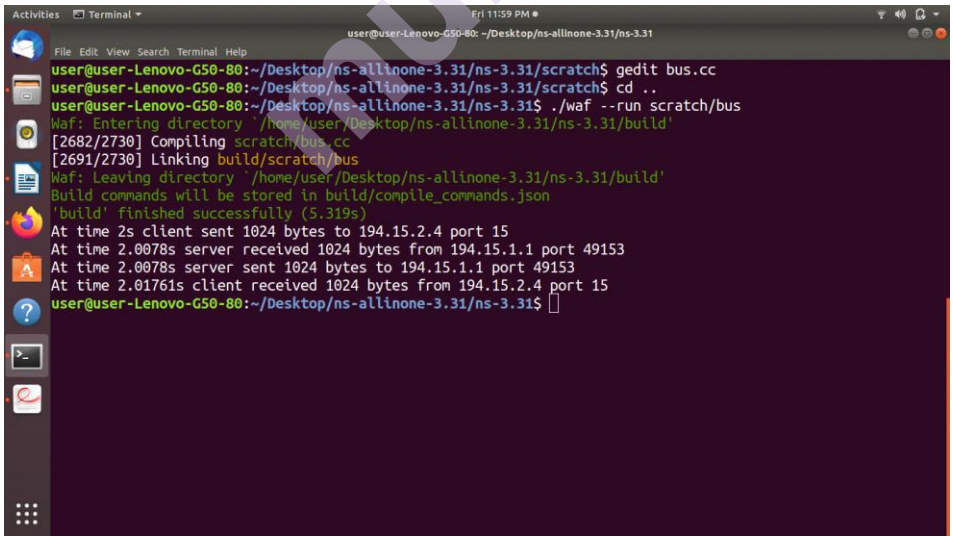


Figure 5 : Output of Bus topology

Result : The output of Bus topology shown in figure 5.

2.5 Program for complex topologies

Hybrid topology refers to a network structure with many topologies in its design. The advantages and disadvantages of each incorporating topology are passed down to the hybrid topology. Aspects of Star, Ring, Bus, and Daisy-chain topologies may be present in the merging topologies.

2.5.1 Write the program to simulate hybrid topology

The majority of WANs are connected using a Dual-Ring topology, and the networks that connect to them are typically Star topologies. The Internet is the most well-known example of a hybrid topology.

Hybrid Network Topology:
Wifi 10.1.3.0
Access Points
*** * * ***
| | | | 10.1.1.0
n5 n6 n7 n0 ---------- n1 n2 n3 n4
point-to-point | | | |
==========
LAN 10.1.2.0

Program : (same as 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"

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));

  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;
}
```

Output :

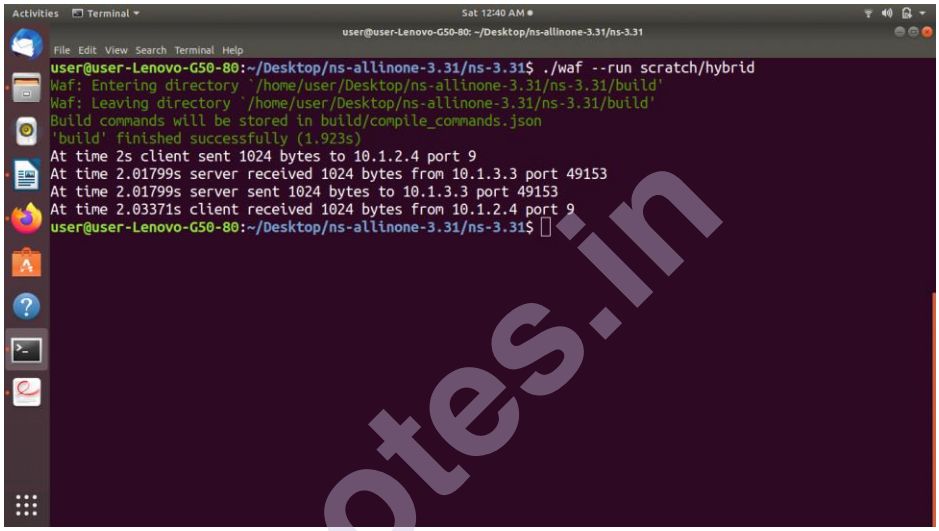


Figure 6 : Output of Hybrid topology

Result : The output of Hybrid topology shown in figure 6. Hybrid topologies, often known as hybrid networks, are networks that mix two or more topologies in such a way that the resulting network does not conform to any of the traditional topologies (e.g., bus, star, ring, etc.)

2.6 Program for client server networks

Software : Eclipse IDE for Java Developers

You can check the output in console window of Eclipse IDE or can be compile and run a java program in command prompt.

Steps to compile and run a java program in command prompt:

- Save your program in a file with a .java extension
- open the command prompt and go to the directory where your file is saved.
- Run the command – javac filename.java
- After the compilation run the command – java filename
- Make sure the Java path is set correctly.

2.6.1 Program to simulate UDP server client

Theory :

1) **DatagramPacket :**

Data is contained in a unit called a datagram when using UDP. A datagram is a self-contained, autonomous message delivered across the network whose delivery, arrival time, and content cannot be guaranteed. A datagram is represented by DatagramPacket in Java.

One of the following constructors can be used to build a DatagramPacket object:

PacketDatagram (byte[] buf, int length)

PacketDatagram (byte[] buf, int length, InetAddress address, int port)

The first function Object() { [native code] } creates a DatagramPacket that will be received.

The second function Object() { [native code] } builds a DatagramPacket to be delivered, therefore you must supply the target host's address and port number.

The length option defines the amount of data in the byte array to be used, which is normally the array's length (buf.length).

2) **DatagramSocket** :

It is the second type of datagram socket. DatagramPackets are sent and received using DatagramSocket. A UDP connection between two machines on a network is represented by a DatagramSocket.Both the client and the server in Java use DatagramSocket.

Like TCP sockets, there are no separate classes for client and server.

So, using one of the constructors below, you create a DatagramSocket object to establish a UDP connection for sending and receiving datagrams:

DatagramSocket()

DatagramSocket is a datagram socket (int port)

DatagramSocket is a datagram socket (int port, InetAddress laddr)

To create a client that binds to an arbitrary port number, use the no-arg function Object() { [native code] }.

The second function Object() { [native code] } creates a server that connects to a given port number, allowing clients to connect to it.

The third function Object() { [native code] } establishes a connection between the server and the provided IP address (in case the computer has multiple IP addresses).

If the socket could not be opened or bind to the supplied port or address, these constructors can throw SocketException. As a result, you must either catch or rethrow this checked exception.

Procedure :

i. Create a new project = "UDPserverclient"
ii. Create Package = "UDP"
iii. Create two files under package as "UDPserver.java" and "UDPclient.java"
iv. Write the programs in respective file to send data from server to client.
v. For transfer of a file, create Datagram socket in both with same number. (Socket numbers can be given from 1024 to 65532.
vi. Also define read from beginning to end of file in both programs.
    ii. First run "UDPserver.java" then run "UDPclient.java".
    iii. Output will be displayed in console window.

Algorithm:

1. Create a server socket.
2. Then create a client socket.
3. DatagramInputStream and DataOutputStream is to abstract different ways to input and output the stream is a file.
4. Specify port number for socket.
5. Close all streams.
6. Close server and client socket.
7. stop

Program for UDP Server :

```
package UDP;
import java.io.*;
import java.net.*;
public class UDPserver {

    public static void main(String[] args) throws IOException {

        DatagramSocket server = new DatagramSocket(3500);
        byte[] buf = new byte[256];
        DatagramPacket packet = new
DatagramPacket(buf,buf.length);
        server.receive(packet);
        String response = new String(packet.getData());
        System.out.println("Response Data : "+response);
        server.close();
}
}
```

Program for UDP Client :

```
package UDP;

import java.io.*;
import java.net.*;

public class UDPclient {

        public static void main(String[] args) throws IOException {

                DatagramSocket client = new DatagramSocket();
                InetAddress add = InetAddress.getByName("localhost");
                String str = "Hello everyone";
                byte[] buf = str.getBytes();
                DatagramPacket p = new
DatagramPacket(buf,buf.length,add,3500);
                client.send(p);
               }
}
```

Simulated output :

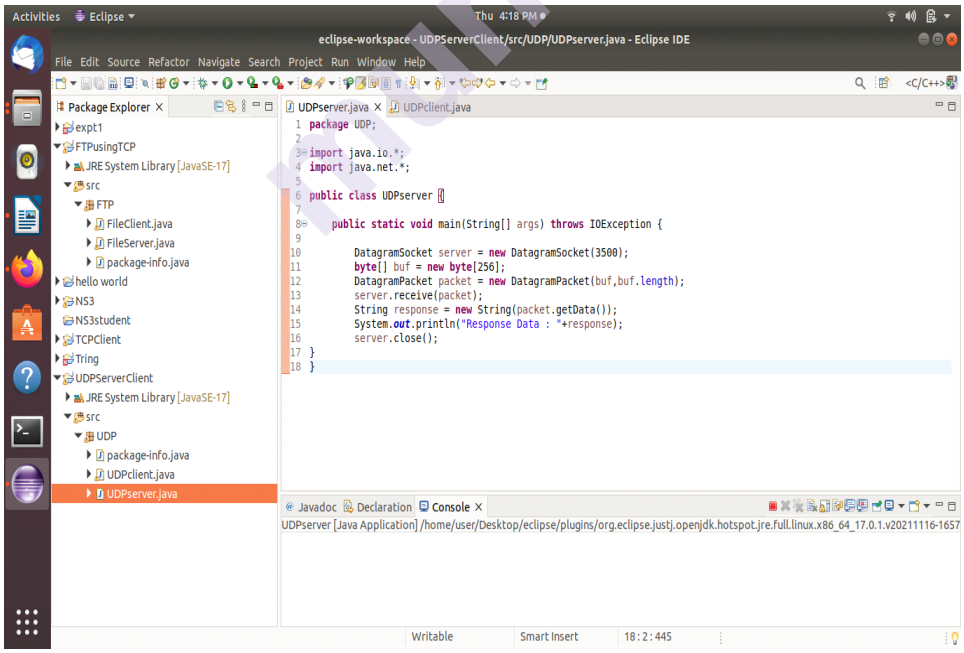


Figure 7 : Run file of UDP server.java

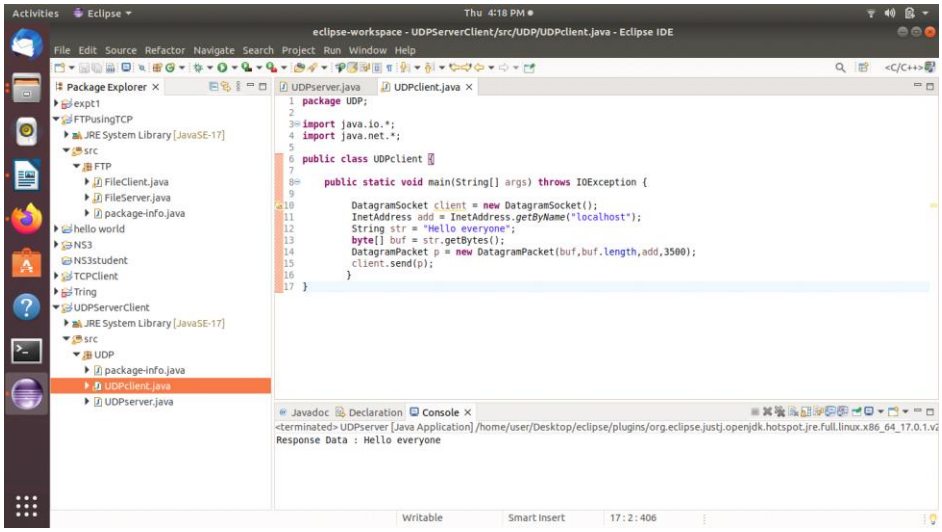


Figure 8: Run file of UDP client.java and Output shown in console window

Result : UDP client – server connected and send message successfully as shown in figure 7 and 8.

2.6.2 Program to simulate FTP using TCP protocol

The File Transfer Protocol (FTP) is a standard network protocol for transferring computer files between hosts on a TCP-based network, such as the Internet.

FTP is based on a client-server design, with the client and server using separate control and data connections. FTP users can connect anonymously if the server is set to allow it, or they can authenticate themselves using a clear-text sign-in protocol, usually in the form of a username and password. FTP is frequently secured using SSL/TLS for secure transmission that protects the username and password as well as encrypts the content (FTPS). In other cases, SSH File Transfer Protocol (SFTP) is utilised instead, however it is technologically different.

Procedure :

i. Create a new project = “FTPusingTCP”

ii. Create Package = “FTP”

iii. Create two files under package as “FileServer.java” and “FileClient.java”

iv. Write the programs in respective file to transfer a file from server to client.

v. Give the path which file content need to transfer with new file name as destination.

Example = “Test.txt” to “cubic.txt”

vi. For transfer of a file, create TCP socket in both with same number. (Socket numbers can be given from 1024 to 65532.

vii. Also define path, size of the file, read from beginning to end of file in both programs.
Vii. First run "FileServer.java" then run "FileClient.java".
Viii. Hereafter you can verify the file is transfered with new name succesfully by FTP using TCP protocol.

Algorithm:

1. Create a server socket.
2. Then create a client socket.
3. FileInputStream and FIleOutputStream is to abstract different ways to input and output the stream is a file.
4. Specify the IP address and port number for socket.
5. Close all streams.
6. Close server and client socket.
7. stop

Program for TCP Server :

```
package FTP;
import java.io.*;
import java.net.*;
import java.util.Arrays;
public class FileServer {
        public static void main(String[] args) throws Exception {

                ServerSocket s = new ServerSocket (4002);
                Socket sr = s.accept();   //for accepting socket
                FileInputStream fr = new
FileInputStream("/home/user/Desktop/Test.txt");     //finds file location
         byte b[]= new byte[2500];     //shows file size with any random size
number
         fr.read(b,0,b.length);  //start reading a file from 0th line to length of
file
 OutputStream os = sr.getOutputStream ();   //Converting file to stream to
send to client
 os.write(b, 0, b.length);
        }

}
```

Program for Client :

```
package FTP;
import java.io.*;
import java.net.*;
import java.util.Arrays;
public class FileClient {
```

```
        public static void main(String[] args) throws Exception {

                byte []b= new byte[25004];
                Socket sr = new Socket("localhost", 4002);
                InputStream  is=sr.getInputStream();
        FileOutputStream fr=new
fileOutputStream("/media/user/4668C01C49C8F823/cubic.txt");
                is.read(b,0,b.length);
                fr.write(b,0,b.length);


        }
}
```

Simulated output :

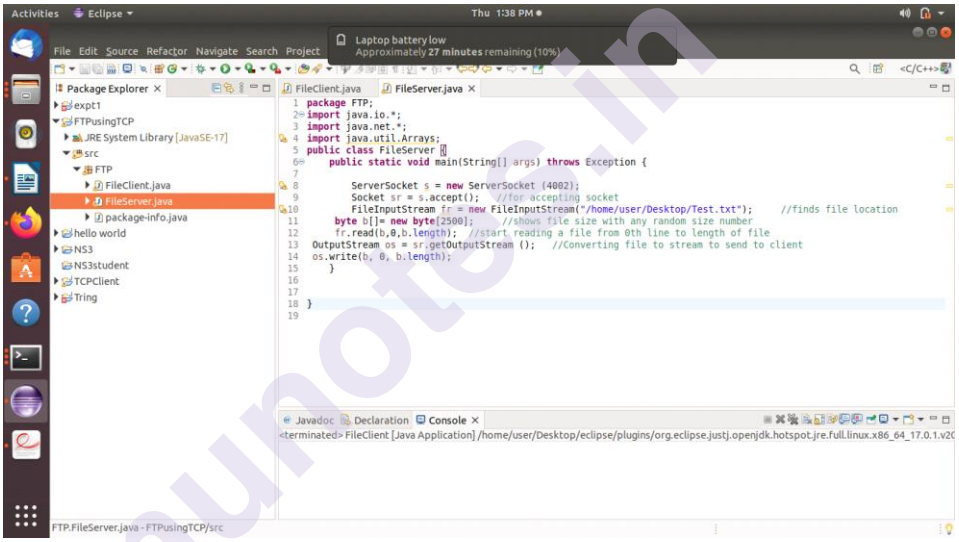


Figure 9 : Run file of TCP server.java

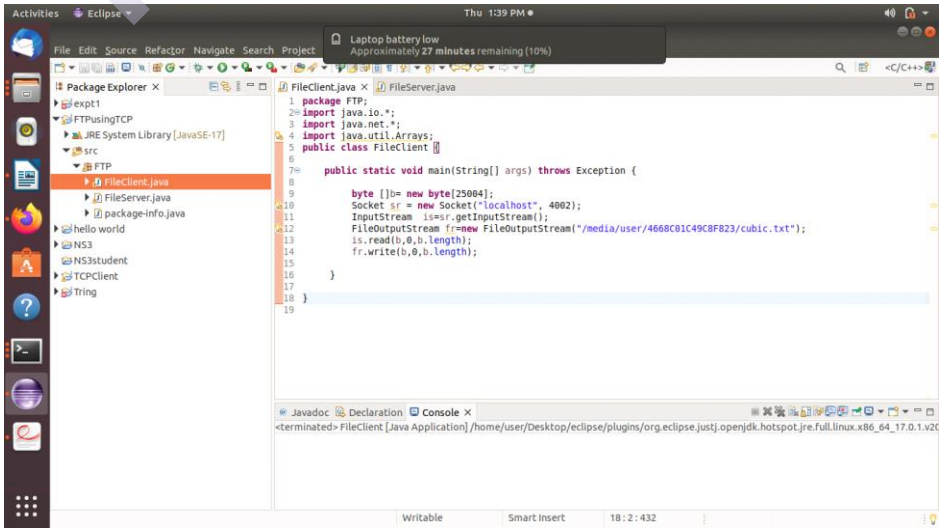


Figure 10 : Run file of TCP client.java

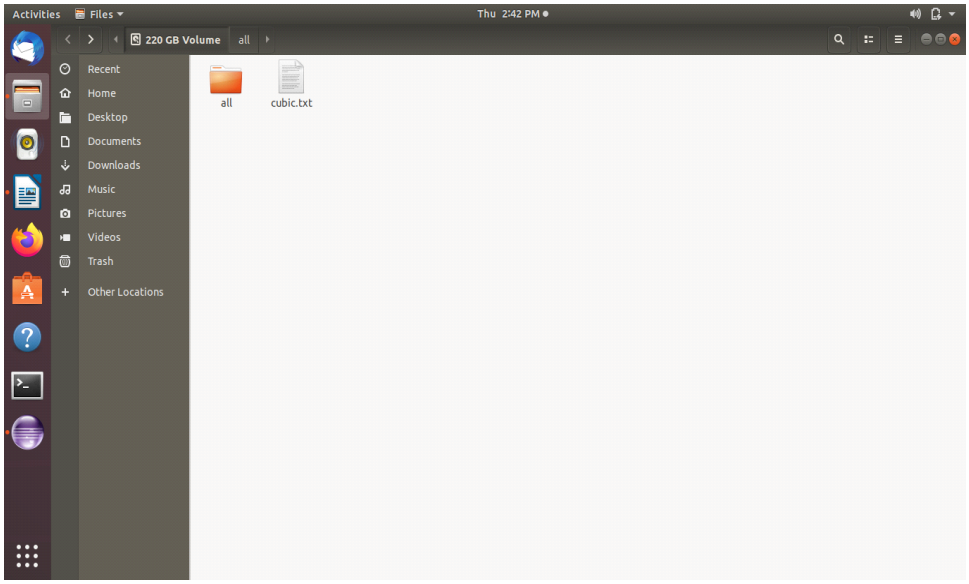


Figure 11 : Transferd file on required location with specified name

Result : The file contents are successfully transferred on new location with another name as shown in Figures 9,10,11 by using FTP over TCP connection.

2.7 Determination of System requirements

- Operating System :-Linux (Ubantu)
- Application Software : NS – 3, Eclipse IDE for JAVA developers

2.8 Summary

We can summarize that the different types of topologies simulated C++ code on NS3 platfom successfully. Also written socket programming for client-server communication for UDP as well as TCP.

2.9 References

1] https://www.nsnam.org/doxygen/index.html

2] ns-3: src/netanim/examples/star-animation.cc Source File (unicamp.br)

3] Learning Network Programming with Java by Richard M.

4] TCP/IP Sockets in Java, Second Edition: Practical Guide for Programmers (The Practical Guides) 2nd Edition by Kenneth L. Calvert , Michael J.

2.10 Unit End Exercises

1. What is network topology?
2. What is the difference between Star and Bus topology?
3. What is role of socket address in networking?
4. Why server is started before the client in a NS3 program?
5. What is the difference between Star and Bus topology?
6. What is hub and how it is different from router?
7. Define Socket.
8. What is socket programming?
9. What is the syntax for connecting Client and Server?
10. What is the command to assign port number to client and server?
11. How to build File Input Stream object with byte array as a parameter.
12. How to read file in byte array with File Input Stream.

❄❄❄❄❄❄❄

3

ANIMATING THE NETWORK

Unit Structure

3.1 OBJECTIVES

After going through this unit, you will be able to:

- Understood the installation and configuration of Network simulator and animation software
- Able to
- construct network topologies using NetAnim software
- Analyze and visualize network packet transfer using network animation software

3.2 Introduction to NetAnim

NetAnim [2] is Graphical user interface (GUI) based network simulator used for NS (Network Simulator)-3. IP address and MAC address of nodes can also be checked in it. We need to follow following steps before and after installation of NetAnim by following steps.

Open the terminal

Step 1 : Update

$sudo apt update

$sudo apt upgrade

Step 2 : Installation and opening of NetAnim in NS3 commands-

The website: http://www.nsnam.org/wiki/index.php/NetAnim

1) Install Mercurial:
   $sudo apt−get/dnf install mercurial
2) Install QT4 development package:
   $sudo apt install qt4-default qt4-qmake

3) $ qmake NetAnim.pro
4) $make
   netAnim is installed
5) ./NetAnim
   opens NetAnim window

3.3 Animation a network using NetAnim

3.3.1 Animate a simple network using NetAnim in Network Simulator

- **Simulated Wired Network**

  Let's discuss the wired network with example Animation1.cc (made changes in tutorial/examples/second.cc) by copying this to scratch folder.

  $ cd ns-3-allinone-3.31

  $ cd ns-3.31

  $ cp scratch/Animation1.cc

  Add the header file to the script Animation1.cc

  #include “ns3/netanim-module.h”

- Add the following statement before Simulation::Run() as shown in Figure 3.

AnimationInterface anim ("Animation1.xml");

AnimationInterface is responsible for creation of XML file and uses the tracing infrastructure to track packets flow between nodes.

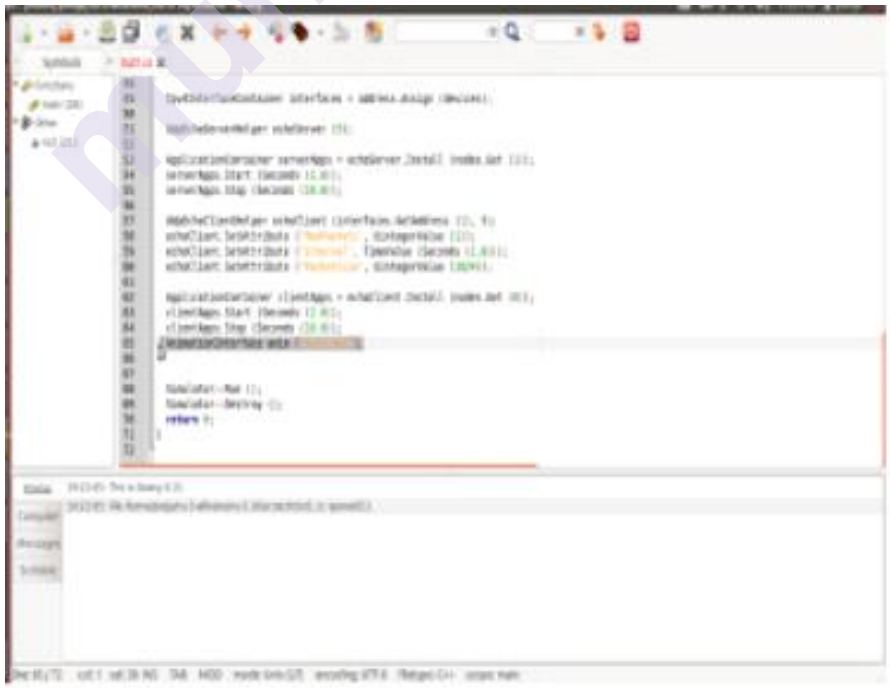


Figure 1: Adding required statement to the script

"SetConstantPosition" set the x-y coordinates of a node which is stationary as shown in program 1,

```
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);
```

NetAnim uses Metadata to provide better statistics and filter, and some brief information about the packets such as TCP sequence number or source and destination IP address during packet animation. Add following statement into Animation1. cc anim. Enable Packet Metadata (true);

Note : we shouldn't enable this feature when using Wimax links.

Save and run the script : Run the file without extension as in Figure 1.

```
$ cd ns-allinone-3.31
$ cd ns-3.31
$ ./waf --run scratch/Animation1
```

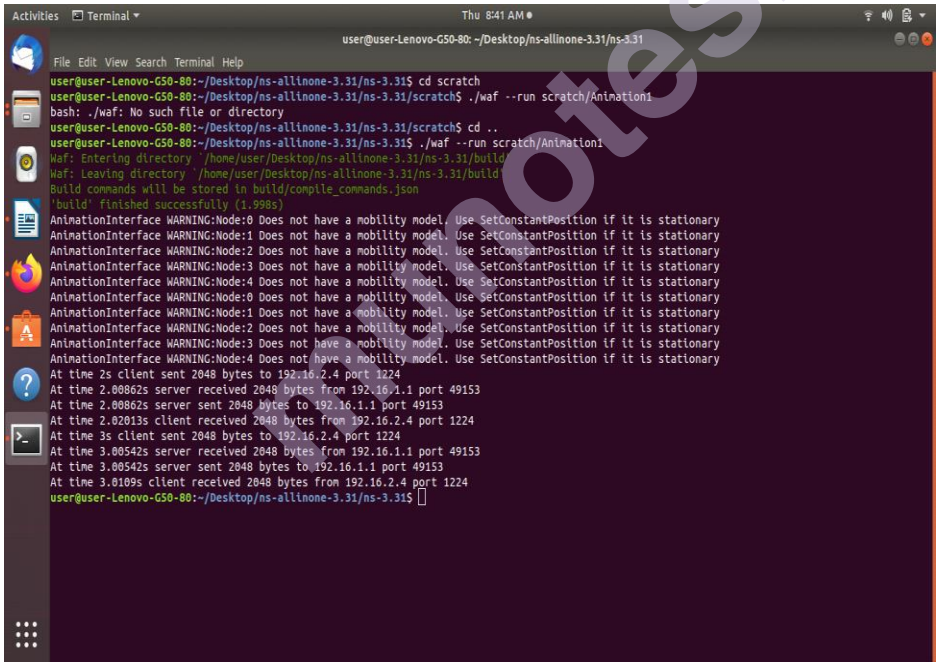


Figure 2. Run the file by waf

To check Animation1.xml file is generated

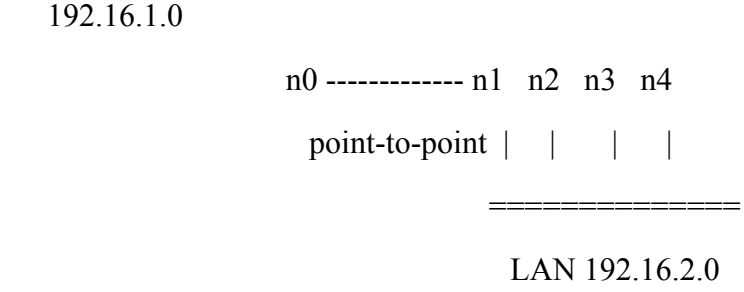
```
/ns-allinone-3.31/ns-3.31$ ls
ns-allinone-3.31/ns-3.31$ cd ..
/ns-allinone-3.31$ cd netanim-3.108
```

Open XML trace file, go to

```
/ns-allinone-3.31/netanim-3.108$./NetAnim
```

NetAnim window will open on screen.
Then goto Open menu -select ns-3.31 -Animation1.xml.

Network Topology :



Program :

```
#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"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("AnimationExample");
int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
//unsigned integer32 bits
  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);
// n0 referred as p2pnodes.Get(0)
// n1 can be referred as csmaNodes.Get(0) or p2pnodes.Get(1)

 NodeContainer csmaNodes;
  csmaNodes.Add (p2pNodes.Get (1)); // Make the first node
  csmaNodes.Create (nCsma);    //nCsma=3 already declared

  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));
// No need to install this -stack.Install(p2pNodes.Get(1));
  stack.Install (csmaNodes);
//p2pNodes.Get(1) = csmaNodes.Get(0)
  Ipv4AddressHelper address;
  address.SetBase ("192.16.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces;
  p2pInterfaces = address.Assign (p2pDevices);

  address.SetBase ("192.16.2.0", "255.255.255.0");
  Ipv4InterfaceContainer csmaInterfaces;
  csmaInterfaces = address.Assign (csmaDevices);

  UdpEchoServerHelper echoServer (1224);

  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get
(nCsma));
```

```
//Server is csmaNodes.Get(3)-n4
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma),
1224);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (2));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (2048));

  ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get
(0));

//n0 is client

  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  pointToPoint.EnablePcapAll ("Animation1");

//pcap will be opened using either wireshark or tcpdump

csma.EnablePcap ("Animation1", csmaDevices.Get (1), true);
csma.EnablePcap ("Animation1", csmaDevices.Get (3), 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;
}
```

Outputs :

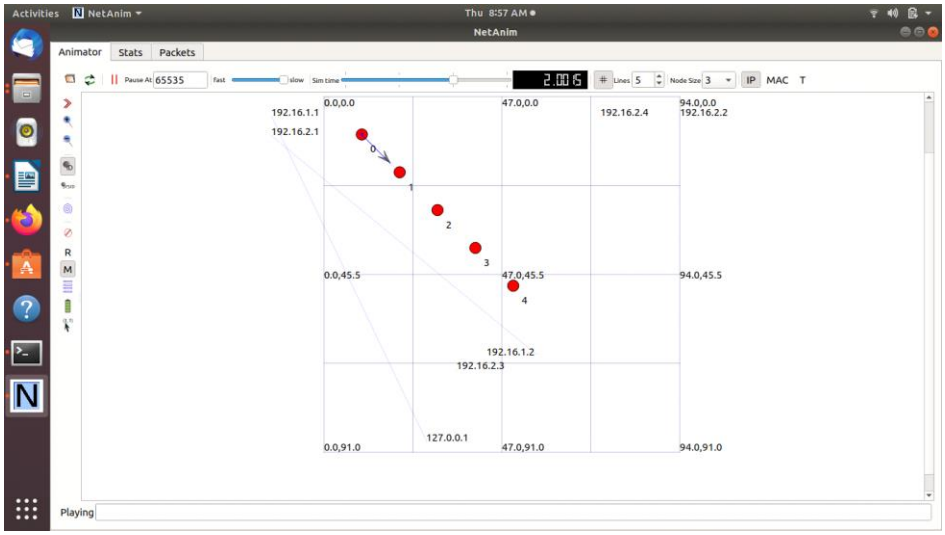


Figure 3: Running a Network using GUI

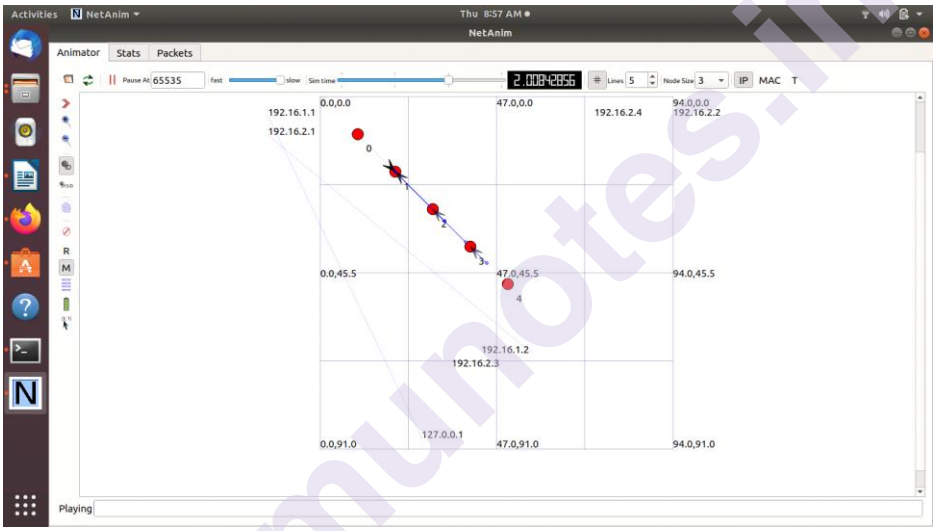


Figure 4: Running a Network using GUI

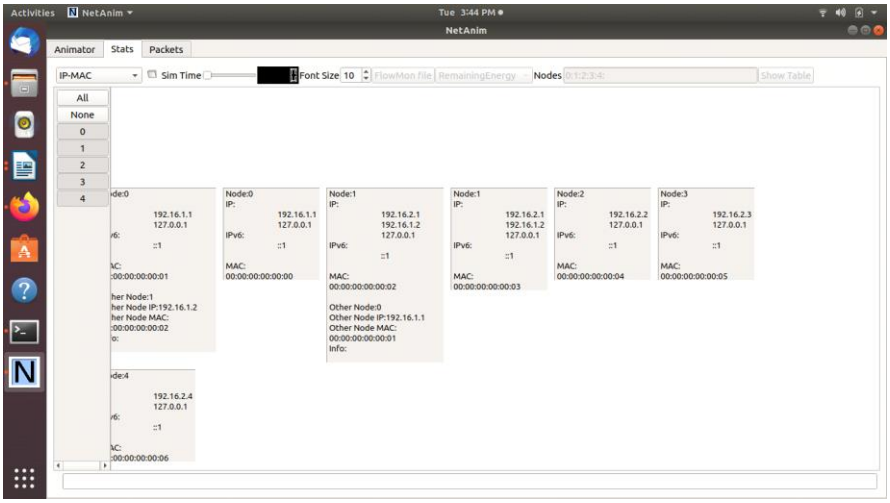


Figure 5: Statistic of a Network

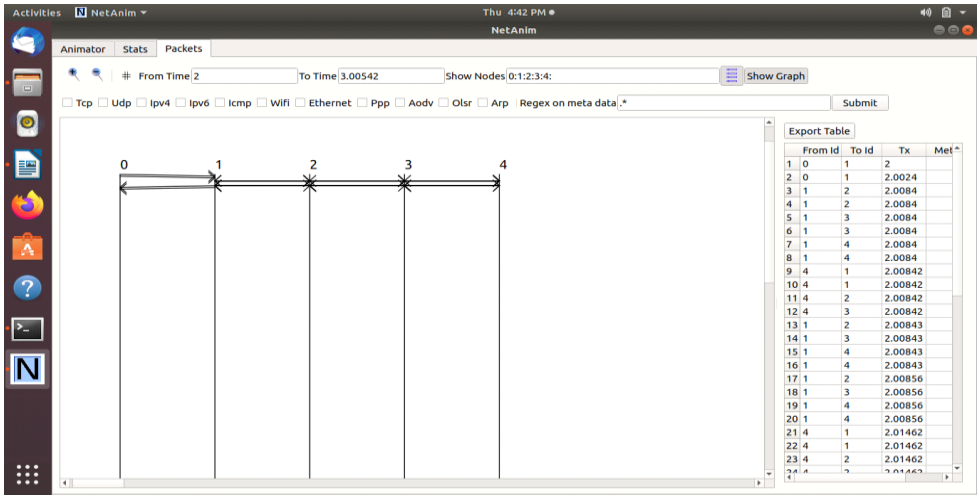


Figure 6: Meta information during simulation

3.3.2 Animate Three way handshake for TCP connection using NetAnim

TCP stands for Transmission Control Protocol, which means it does something to ensure that data is transmitted in a secure manner. Positive Acknowledgement with Re-transmission is a feature of TCP that ensures stable communication (PAR). The transport layer's Protocol Data Unit (PDU) is referred to as a segment. Now, until it receives an acknowledgement, a device implementing PAR will transmit the data unit. The receiver discards the segment if the data unit received at the receiver's end is damaged (it validates the data with the checksum functionality of the transport layer that is used for Error Detection). As a result, the sender must resend the data unit for which there is no affirmative acknowledgement. As you can see from the above process, three segments are exchanged between the sender (client) and receiver (server) in order to establish a reliable TCP connection as shown in Figure 7.

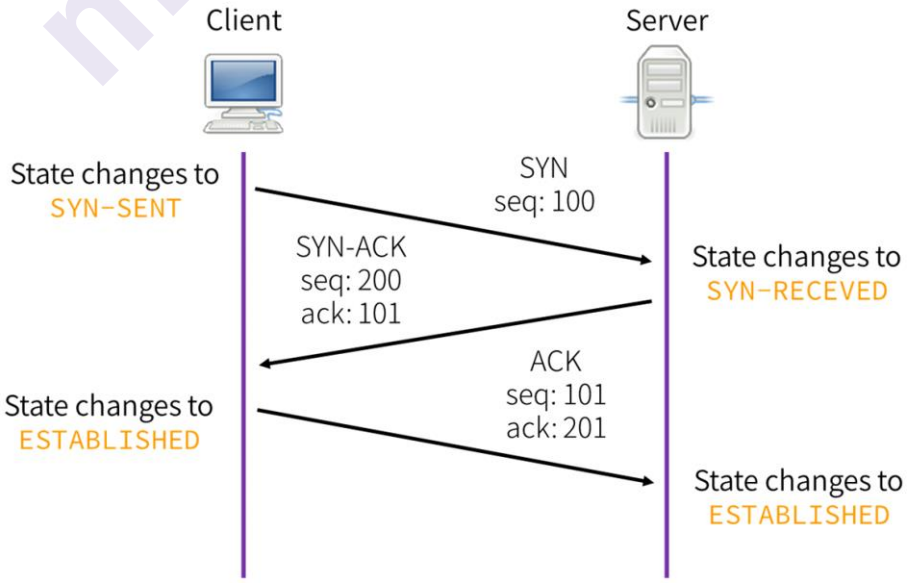


Figure 7 : Process of Three way handshake for TCP connection

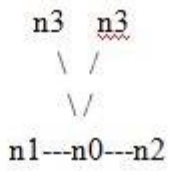
Step 1 (SYN): The client wants to establish a connection with a server, therefore it sends a segment with SYN (Synchronize Sequence Number), which informs the server that the client is likely to begin communication and with what sequence number it will begin segments.

Step 2 (ACK + SYN): With the SYN-ACK signal bits set, the server answers to the client request.

The answer of the segment it received is denoted by ACK, and SYN denotes the sequence number with which it is likely to begin the segments.

Step 3 (acknowledgement): Finally, the client confirms the server's response, and the two create a secure connection via which they will begin the real data transfer.

Network topology : 5 nodes in a star



Program [3]:

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

#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-global-routing-helper.h"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TcpServer");

int
main (int argc, char *argv[])
  {
 // Users may find it convenient to turn on explicit debugging
  // for selected modules; the below lines suggest how to do this

//LogComponentEnable ("TcpServer", LOG_LEVEL_INFO);
```

```
//LogComponentEnable ("TcpL4Protocol", LOG_LEVEL_ALL);
 //LogComponentEnable ("TcpSocketImpl", LOG_LEVEL_ALL);
//LogComponentEnable ("PacketSink", LOG_LEVEL_ALL);

// Set up some default values for the simulation.
  Config::SetDefault ("ns3::OnOffApplication::PacketSize",
UintegerValue (250));
 Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue
("5kb/s"));
  uint32_t N = 5; //number of nodes in the star
  // Allow the user to override any of the defaults and the above
 // Config::SetDefault()s at run-time, via command-line arguments
 CommandLine cmd;
cmd.AddValue ("nNodes", "Number of nodes to place in the star", N);
  cmd.Parse (argc, argv);

  // Here, we will create N nodes in a star.
   NS_LOG_INFO ("Create nodes.");
NodeContainer serverNode;
   NodeContainer clientNodes;
serverNode.Create (1);
 clientNodes.Create (N-1);
NodeContainer allNodes = NodeContainer (serverNode, clientNodes);

  // Install network stacks on the nodes
  InternetStackHelper internet;
  internet.Install (allNodes);

 //Collect an adjacency list of nodes for the p2p topology
  std::vector<NodeContainer> nodeAdjacencyList (N-1);
  for(uint32_t i=0; i<nodeAdjacencyList.size (); ++i)
    {
     nodeAdjacencyList[i] = NodeContainer (serverNode, clientNodes.Get
(i));
    }

 // We create the channels first without any IP addressing information
 NS_LOG_INFO ("Create channels.");
  PointToPointHelper p2p;
  p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));
  std::vector<NetDeviceContainer> deviceAdjacencyList (N-1);
   for(uint32_t i=0; i<deviceAdjacencyList.size (); ++i)
    {
```

```
    deviceAdjacencyList[i] = p2p.Install (nodeAdjacencyList[i]);
   }

  // Later, we add IP addresses.
   NS_LOG_INFO ("Assign IP Addresses.");
  Ipv4AddressHelper ipv4;
  std::vector<Ipv4InterfaceContainer> interfaceAdjacencyList (N-1);
   for(uint32_t i=0; i<interfaceAdjacencyList.size (); ++i)
    {
     std::ostringstream subnet;
     subnet<<"10.1."<<i+1<<".0";
      ipv4.SetBase (subnet.str ().c_str (), "255.255.255.0");
     interfaceAdjacencyList[i] = ipv4.Assign (deviceAdjacencyList[i]);
    }

   //Turn on global static routing
   Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

   // Create a packet sink on the star "hub" to receive these packets
    uint16_t port = 50000;
    Address sinkLocalAddress (InetSocketAddress (Ipv4Address::GetAny
(), port));
    PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory",
sinkLocalAddress);
   ApplicationContainer sinkApp = sinkHelper.Install (serverNode);
    sinkApp.Start (Seconds (1.0));
   sinkApp.Stop (Seconds (10.0));

  // Create the OnOff applications to send TCP to the server
  OnOffHelper clientHelper ("ns3::TcpSocketFactory", Address ());
   clientHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  clientHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));

  //normally wouldn't need a loop here but the server IP address is different
  //on each p2p subnet
  ApplicationContainer clientApps;
```

```
for(uint32_t i=0; i<clientNodes.GetN (); ++i)
{
 AddressValue remoteAddress
  (InetSocketAddress (interfaceAdjacencyList[i].GetAddress (0), port));
 clientHelper.SetAttribute ("Remote", remoteAddress);
  clientApps.Add (clientHelper.Install (clientNodes.Get (i)));
 }
clientApps.Start (Seconds (1.0));
 clientApps.Stop (Seconds (10.0));

 //configure tracing
  AsciiTraceHelper ascii;
  p2p.EnableAsciiAll (ascii.CreateFileStream ("tcp-star-server.tr"));
 p2p.EnablePcapAll ("tcp-star-server");

 NS_LOG_INFO ("Run Simulation.");
AnimationInterface anim ("Anim.xml");
anim.EnablePacketMetadata (true);

 Simulator::Run ();
 Simulator::Destroy ();
 NS_LOG_INFO ("Done.");
  return 0;
  }
```

Outputs :

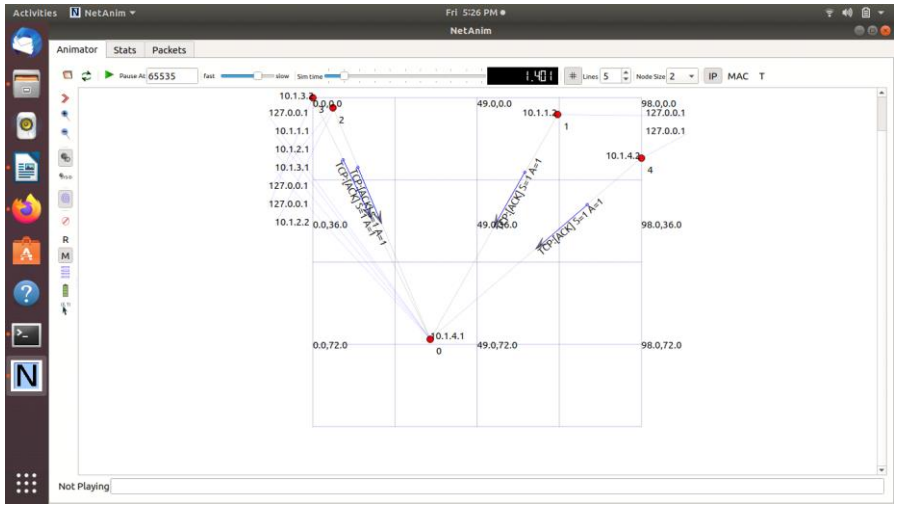


Figure 8: Running a TCP 3 ways handshake using GUI

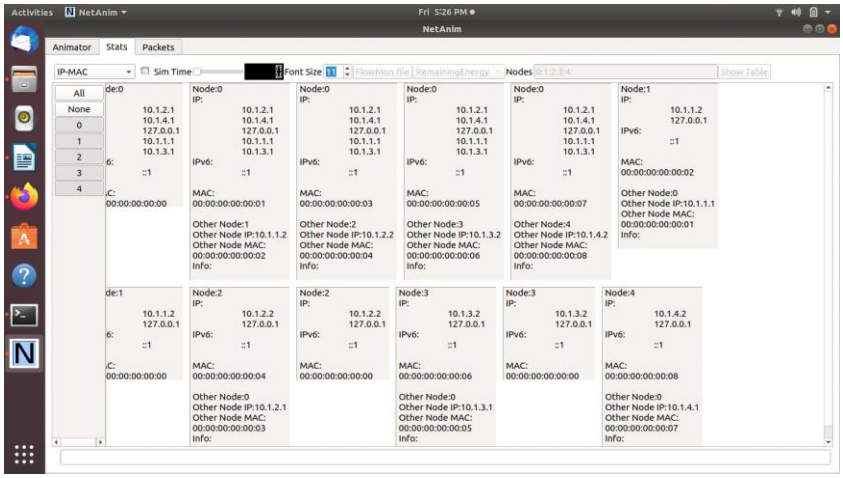


Figure 9: Statistic of a Network

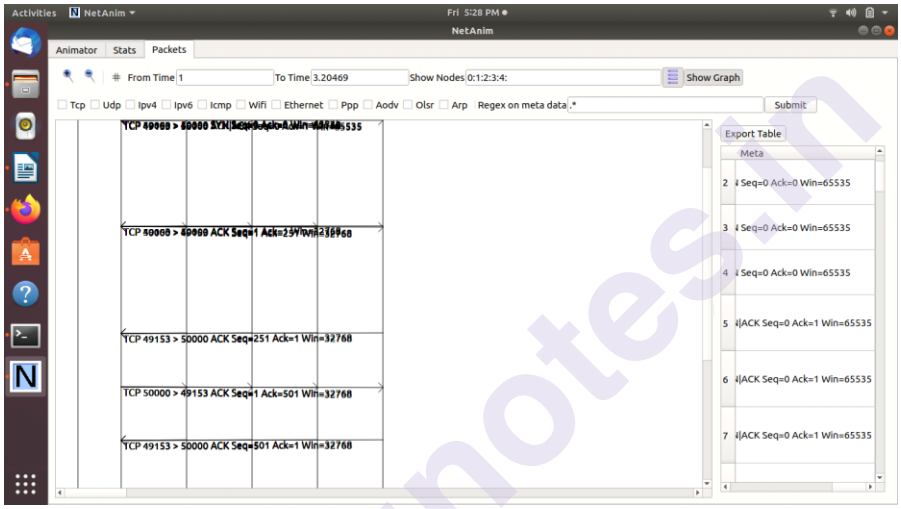


Figure 10: Meta information during simulation

rom	Id	To	Id	Tx	Meta
1	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
2	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
3	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
4	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
5	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
6	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
7	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
8	0	1			TCP 49153 > 50000 SYN Seq=0 Ack=0 Win=65535
0	1	1.00209			TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1 Win=65535
0	2	1.00209			TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1 Win=65535
0	3	1.00209			TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1 Win=65535
0	4	1.00209			TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1 Win=65535

```
0       5       1.00209         TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1
Win=65535
0       6       1.00209         TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1
Win=65535
0       7       1.00209         TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1
Win=65535
0       8       1.00209         TCP 50000 > 49153 SYN|ACK Seq=0 Ack=1
Win=65535
1       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
2       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
3       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
4       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
5       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
6       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
7       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
8       0       1.00419         TCP 49153 > 50000 ACK Seq=1 Ack=1
Win=32768
1       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
2       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
3       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
4       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
5       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
6       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
7       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
8       0       1.4     TCP 49153 > 50000 ACK Seq=1 Ack=1 Win=32768
0       1       1.40249         TCP 50000 > 49153 ACK Seq=1 Ack=251
Win=32768
0       2       1.40249         TCP 50000 > 49153 ACK Seq=1 Ack=251
Win=32768
0       3       1.40249         TCP 50000 > 49153 ACK Seq=1 Ack=251
Win=32768
0       4       1.40249         TCP 50000 > 49153 ACK Seq=1 Ack=251
Win=32768
0       5       1.40249         TCP 50000 > 49153 ACK Seq=1 Ack=251
Win=32768
0       6       1.40249         TCP 50000 > 49153 ACK Seq=1 Ack=251
Win=32768
0       7       1.40249         TCP 50000 > 49153 ACK Seq=1 Ack=251
Win=32768
```

0	8	1.40249	TCP 50000 > 49153 ACK Seq=1 Ack=251 Win=32768
1	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
2	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
3	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
4	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
5	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
6	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
7	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
8	0	1.8	TCP 49153 > 50000 ACK Seq=251 Ack=1 Win=32768
0	1	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768
0	2	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768
0	3	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768
0	4	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768
0	5	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768
0	6	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768
0	7	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768
0	8	2.00249	TCP 50000 > 49153 ACK Seq=1 Ack=501 Win=32768

Table 1 : Complete Export table from Figure 9 after simulation of TCP 3 ways handshake

3.4 Summary

We can summarize that the NetAnim (GUI-based network simulator) explains how to run a network using a GUI.

TCP 3-way handshake, also known as a three-way handshake or TCP 3-way handshake, is a protocol for establishing a connection between a server and a client in a TCP/IP network.

A client must initiate the conversation by requesting a communication session with the server via the TCP handshake mechanism. The client creates a connection with the server in the first phase. The server responds to the client request with the SYN-ACK signal set in the second step. The client acknowledges the Server's response in this final stage.

The connection between two independent endpoints is automatically terminated via TCP.

3.5 References

1] https://www.nsnam.org/doxygen/index.html
2] https://www.ijcseonline.org/pub_paper/10-IJCSE-01425%20.pdf
3] https://www.nsnam.org/docs/release/3.18/doxygen/tcp-star-server_8cc_source.html
4] Learning Network Programming with Java by Richard M.
5] TCP/IP Sockets in Java, Second Edition: Practical Guide for Programmers (The Practical Guides) 2nd Edition by Kenneth L. Calvert , Michael J.

3.6 Unit End Exercises

1. What is the difference between IP address and physical address?
2. What is the role of NetAnim in NS3 simulator?
3. What is the role of scratch folder in NS3 program simulation?
4. Give the difference between TCP and UDP protocol.
5. What is PORT?
6. Give the significance of TCP acknowledgments.
7. Explain the process of three-way handshaking protocol.
8. What is retransmission?

❄❄❄❄❄❄❄

4

ANALYZING NETWORK TRAFFIC USING WIRE SHARK

Unit Structure

EXPERIMENTS TO BE EXECUTED

1 Study the steps for installing Wireshark, the packet-sniffing tool for performing Network analysis
2 Study of working with captured packets
3 Study of advanced Wireshark features
4 Study of security packet analysis
5 Study of Operating System Fingerprinting

4.0 WIRESHARK

Wireshark is the world's foremost and widely-used network protocol analyzer. It lets us to know what's happening on our network at a microscopic level and is the de facto standard across many commercial and non-profit enterprises, government agencies, and educational institutions.

Wireshark has a rich feature set and includes the following:

- Deep inspection of hundreds of protocols
- Live capture and offline analysis
- Standard three-pane packet browser
- Multi-platform
- Captured network data can be browsed via a GUI, TTY-mode TShark utility
- Powerful display filters
- Rich VoIP analysis
- Read/write many different capture file formats
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Cloning rules can be applied to the packet list for quick intuitive analysis
- Output can be exported to XML, PostScript, CSV or plain text

Wireshark is a packet sniffer and analysis tool. It captures network traffic on the local network and stores that data for offline analysis. Wireshark captures network traffic from Ethernet, Bluetooth, Wireless (IEEE.802.11), Token Ring, Frame Relay connections, and more.

Ed. Note: A "packet" is a single message from any network protocol (i.e., TCP, DNS, etc.)

Ed. Note 2: LAN traffic is in broadcast mode, meaning a single computer with Wireshark can see traffic between two other computers. If you want to see traffic to an external site, you need to capture the packets on the local computer.

Wireshark allows you to filter the log either before the capture starts or during analysis, so you can narrow down and zero into what you are looking for in the network trace. For example, you can set a filter to see TCP traffic between two IP addresses. You can set it only to show you the packets sent from one computer.

Download Wireshark

Check the official Wireshark Download page for the operating system you need.

4.1 WIRESHARK FOR WINDOWS

Wireshark comes in two flavors for Windows, 32 bit and 64 bit and the installation is simple.

4.2 WIRESHARK FOR MAC

Wireshark is available on Mac as Homebrew install.
Run this command at your terminal prompt:
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
To install Wireshark run this command from the Terminal:
brew install wireshark

4.3 WIRESHARK FOR LINUX

Installing Wireshark on Linux can be a little different depending on the Linux distribution. If you aren't running one of the following distros, please double-check the commands.

Ubuntu

From a terminal prompt, run these commands:

1. sudo apt-get install wireshark
2. sudo dpkg-reconfigure wireshark-common
3. sudo adduser $USER wireshark

Those commands download the package, update the package, and add user privileges to run Wireshark.

Red Hat Fedora

From a terminal prompt, run these commands:

1. sudo dnf install wireshark-qt
2. sudo usermod -a -G wireshark username

The first command installs the GUI and CLI version of Wireshark, and the second adds permissions to use Wireshark.

Kali Linux

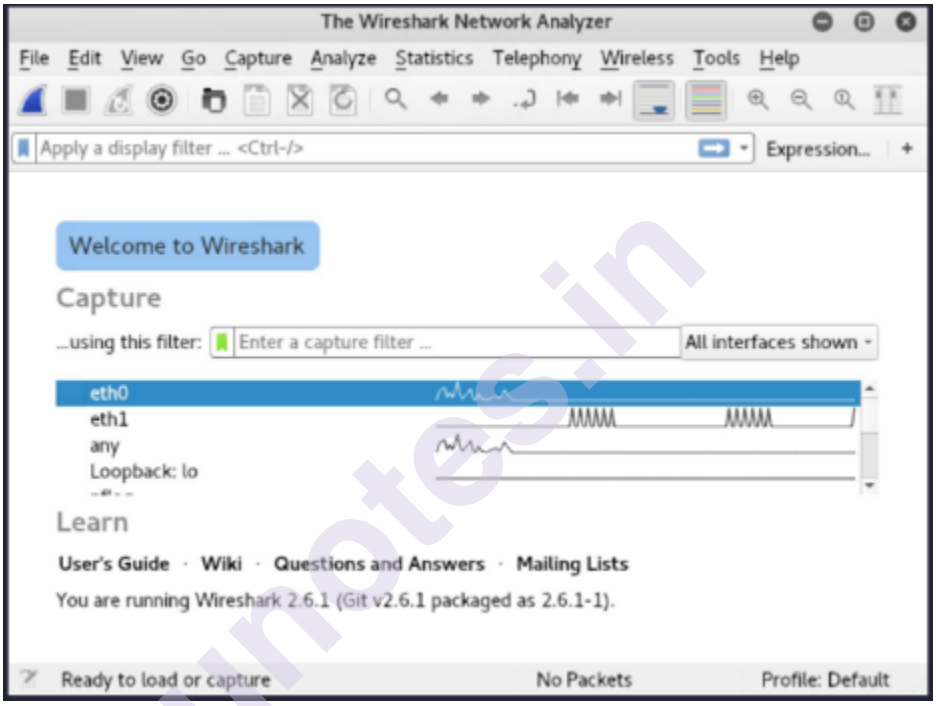
Wireshark is probably already installed! It's part of the basic package. Check your menu to verify. It's under the menu option "Sniffing & Spoofing."

4.4 DATA PACKETS ON WIRESHARK

Now that we have Wireshark installed let's go over how to enable the Wireshark packet sniffer and then analyze the network traffic.

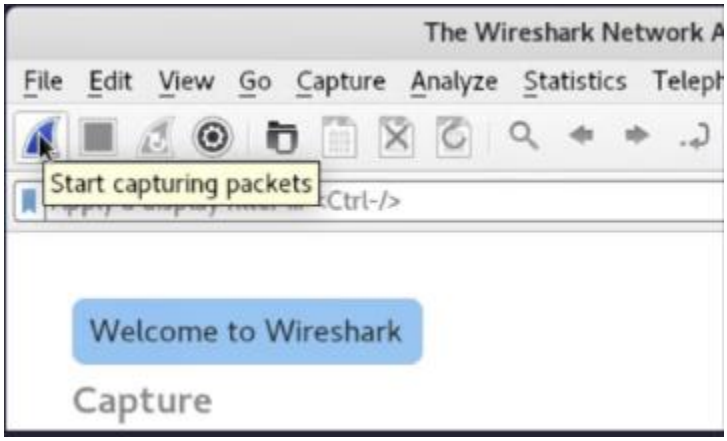
4.5 CAPTURING DATA PACKETS ON WIRESHARK

When you open Wireshark, you see a screen that shows you a list of all of the network connections you can monitor. You also have a capture filter field, so you only capture the network traffic you want to see.

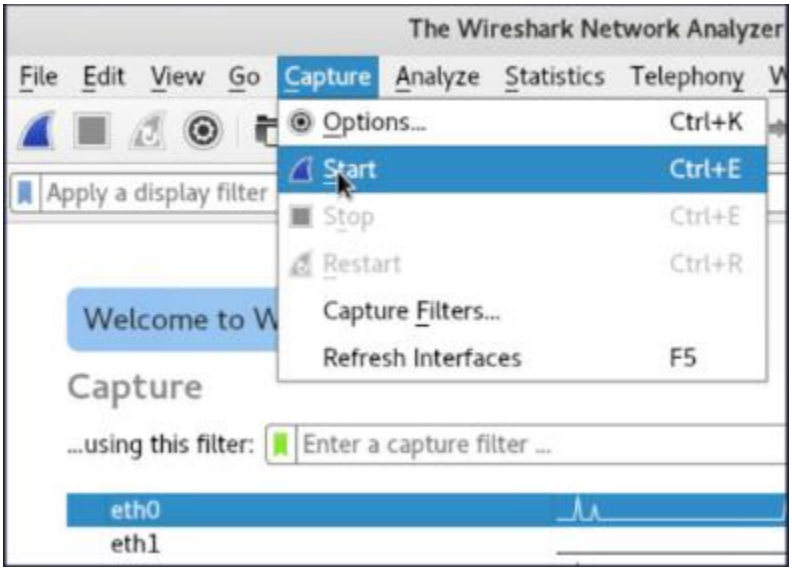


You can select one or more of the network interfaces using "shift left-click." Once you have the network interface selected, you can start the capture, and there are several ways to do that.

Click the first button on the toolbar, titled "Start Capturing Packets."

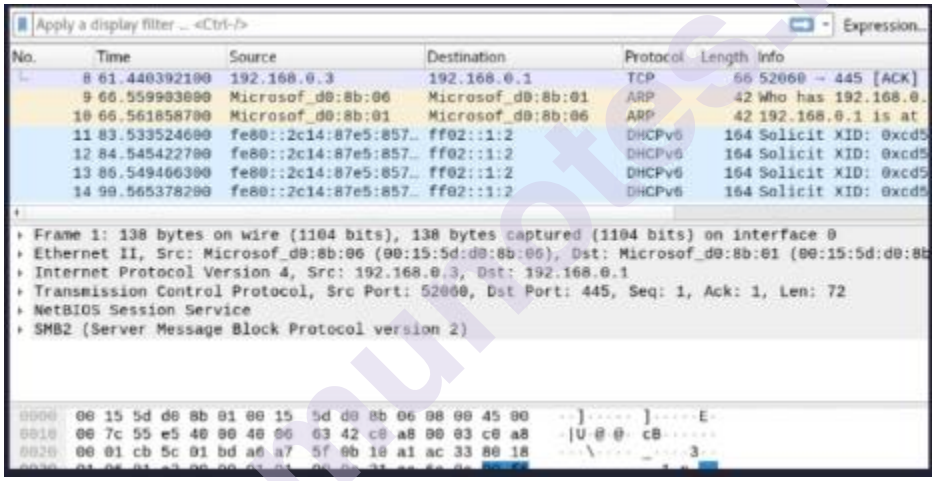


You can select the menu item Capture -> Start.



Or you could use the keystroke Control – E.

During the capture, Wireshark will show you the packets that it captures in real-time.



Once you have captured all the packets you need, you use the same buttons or menu options to stop the capture.

Best practice says that you should stop Wireshark packet capture before you do analysis.

4.6 ANALYZING DATA PACKETS ON WIRESHARK

Wireshark shows three different panes for inspecting packet data. The Packet List, the top pane, is a list of all the packets in the capture. When you click on a packet, the other two panes change to show you the details about the selected packet. Details about each column in the top pane:

- **No.**: This is the number order of the packet that got captured. Bracket indicates that this packet is part of a conversation.

- **Time**: This column shows you how long after you started the capture that this packet got captured.
- **Source**: Address of the system that sent the packet.
- **Destination**: Address of the destination of that packet.
- **Protocol**: Type of packet (TCP, DNS, DHCPv6, or ARP).
- **Length**: Length of the packet in bytes.
- **Info**: Information about the packet contents.

Packet Details, the middle pane, shows you as much readable information about the packet as possible.

The bottom pane, Packet Bytes, displays the packet exactly as it got captured in hexadecimal.

4.7 WIRESHARK FILTERS

One of the best features of Wireshark is the Wireshark Capture Filters and Wireshark Display Filters. Filters allow you to view the capture the way you need to see it so you can troubleshoot the issues at hand.

4.8 WIRESHARK CAPTURE FILTERS

Capture filters limit the captured packets by the filter. Meaning if the packets don't match the filter, Wireshark won't save them. Here are some examples of capture filters:

host IP-*address*: this filter limits the capture to traffic to and from the IP address

net 192.168.0.0/24: this filter captures all traffic on the subnet.

dst host IP-*address*: capture packets sent to the specified host.

port 53: capture traffic on port 53 only.

port not 53 and not arp: capture all traffic except DNS and ARP traffic

4.9 WIRESHARK DISPLAY FILTERS

Wireshark Display Filters change the view of the capture during analysis. After you have stopped the packet capture, you use display filters to narrow down the packets in the Packet List so you can troubleshoot your issue.

The most useful (in my experience) display filter is:

ip.src==*IP-address* and ip.dst==*IP-address*

This filter shows you packets from one computer (ip.src) to another (ip.dst). You can also use ip.addr to show you packets to and from that IP. Here are some others:

tcp.port eq 25: This filter will show you all traffic on port 25, which is usually SMTP traffic.

icmp: This filter will show you only ICMP traffic in the capture, most likely they are pings.

ip.addr != *IP_address*: This filter shows you all traffic except the traffic to or from the specified computer.

Analysts even build filters to detect specific attacks, like this filter to detect the Sasser worm:

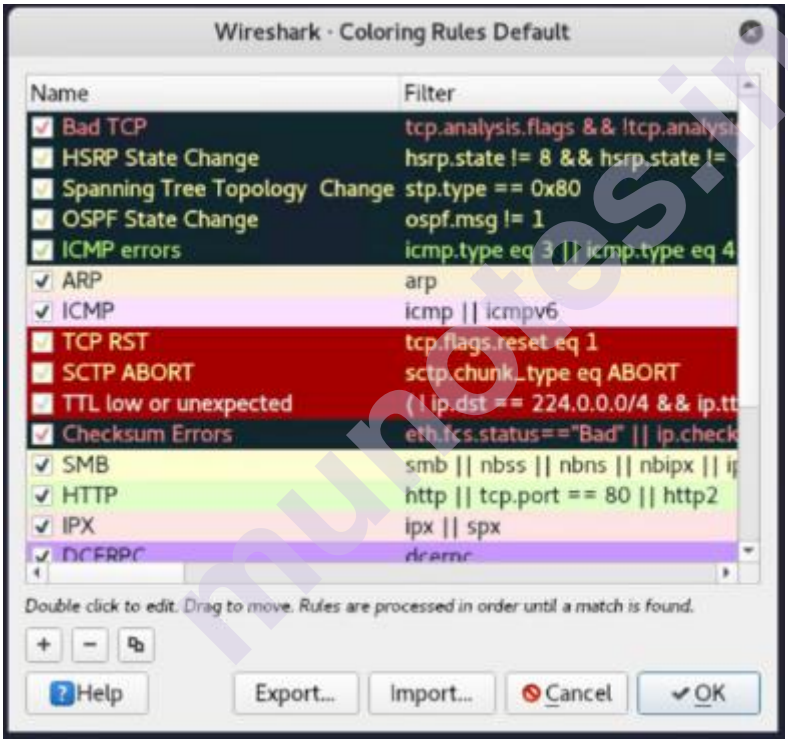
ls_ads.opnum==0x09

Additional Wireshark Features

Beyond the capture and filtering, there are several other features in Wireshark that can make your life better.

4.10 WIRESHARK COLORIZATION OPTIONS

We can setup Wireshark so it colors your packets in the Packet List according to the display filter, which allows you to emphasize the packets you want to highlight. Check out some examples here.



Wireshark Promiscuous Mode

Wireshark only captures packets going to and from the computer where it runs. By checking the box to run Wireshark in Promiscuous Mode in the Capture Settings, you can capture most of the traffic on the LAN.

4.11 WIRESHARK COMMAND LINE

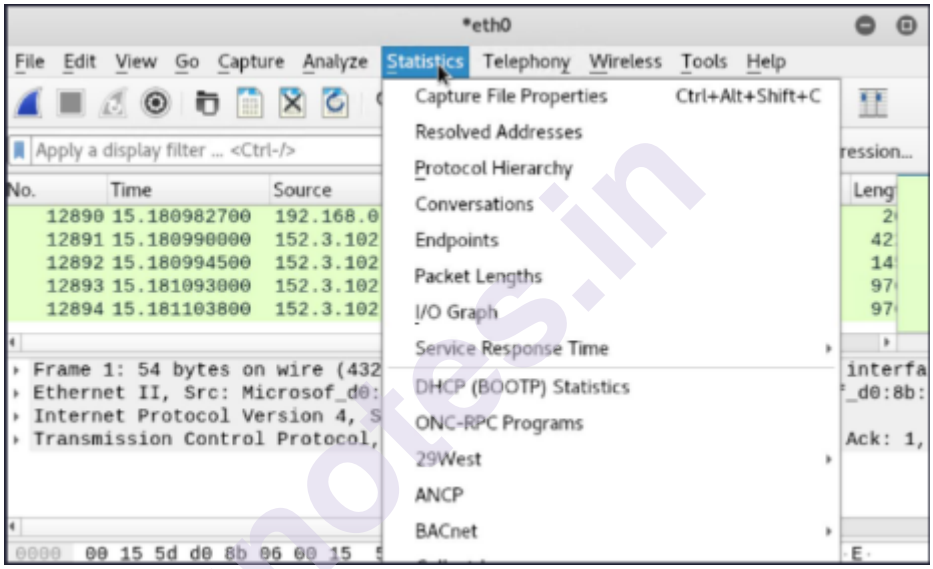
Wireshark does provide a Command Line Interface (CLI) if you operate a system without a GUI. Best practice would be to use the CLI to capture and save a log so you can review the log with the GUI.

Wireshark Commands

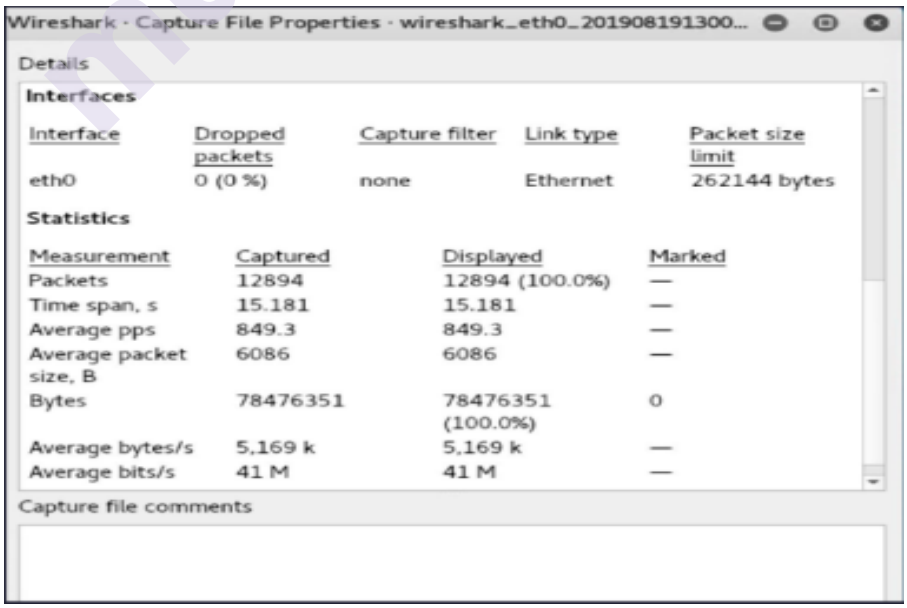
- wireshark : run Wireshark in GUI mode
- wireshark –h : show available command line parameters for Wireshark
- wireshark –a duration:300 –i eth1 –w wireshark. : capture traffic on the Ethernet interface 1 for 5 minutes. –a means automatically stop the capture, -i specifics which interface to capture

Metrics and Statistics

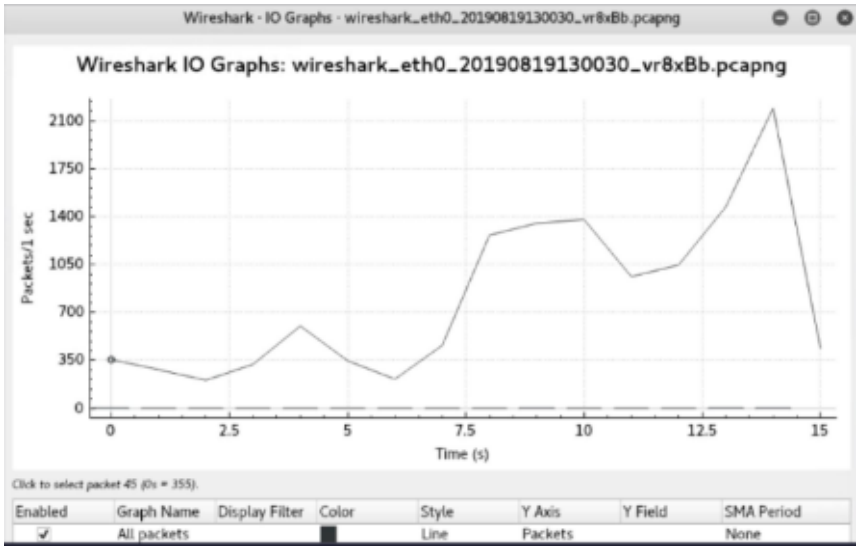
Under the Statistics menu item, you will find a plethora of options to show details about your capture.



Capture File Properties:



Wireshark I/O Graph:



4.12 NETWORK TRAFFIC ANALYSIS USING WIRESHARK

Network traffic analysis is the routine task of various job roles, such as network administrator, network defenders, incident responders and others.

Capture filters with protocol header values

Wireshark comes with several capture and display filters. But a user can create display filters using protocol header values as well. Use this technique to analyze traffic efficiently.

proto[offset:size(optional)]=value

Following the above syntax, it is easy to create a dynamic capture filter, where:

- proto = desired protocol
- offset = header value
- size = data length
- value = data you want to find

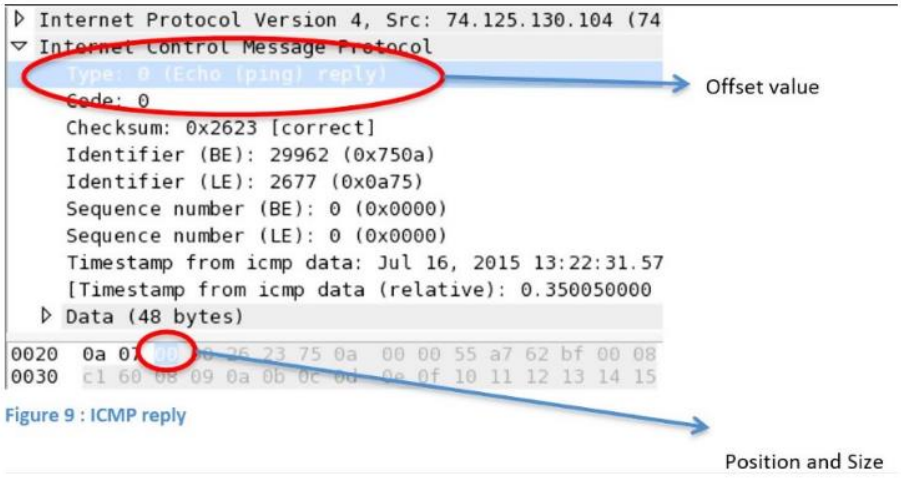


Figure 9 : ICMP reply

Figure 1 Source: resources.infosecinstitute.com

Some instances are in the following table:

Filter	Description
icmp[0] = 0	ICMP request packets
icmp[0:1] = 8	ICMP reply packets
tcp[13] = 18	TCP SYN/ACK flag packets only
tcp[13] = 32	TCP URG flag set packets only

Analyzing endpoints

This feature comes in handy to determine the endpoint generating the highest volume or abnormal traffic in the network. To analyze the endpoints between two communication devices, do the following:

Capture traffic and select the packet whose endpoint you wish to check. -> Click Statistics menu -> Select Endpoints.

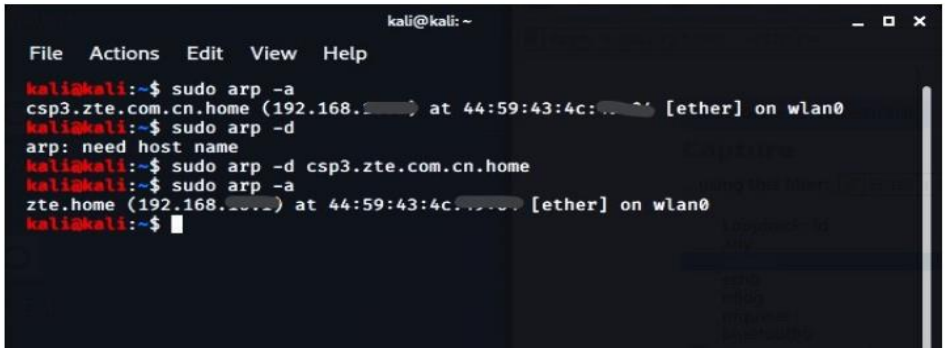
The most traffic-intensive endpoint, as seen in the picture below, is 192.168.10.4.

ARP traffic analysis

Address resolution protocol (ARP) generally uses to find the MAC address of the target machine. Let's try capturing and analyzing ARP traffic.

First things first, know the target machine IP. In our case, it's going to be the default gateway address.

Find existing ARP cache -> Delete the existing one to understand the demo -> Check ARP cache for verification.



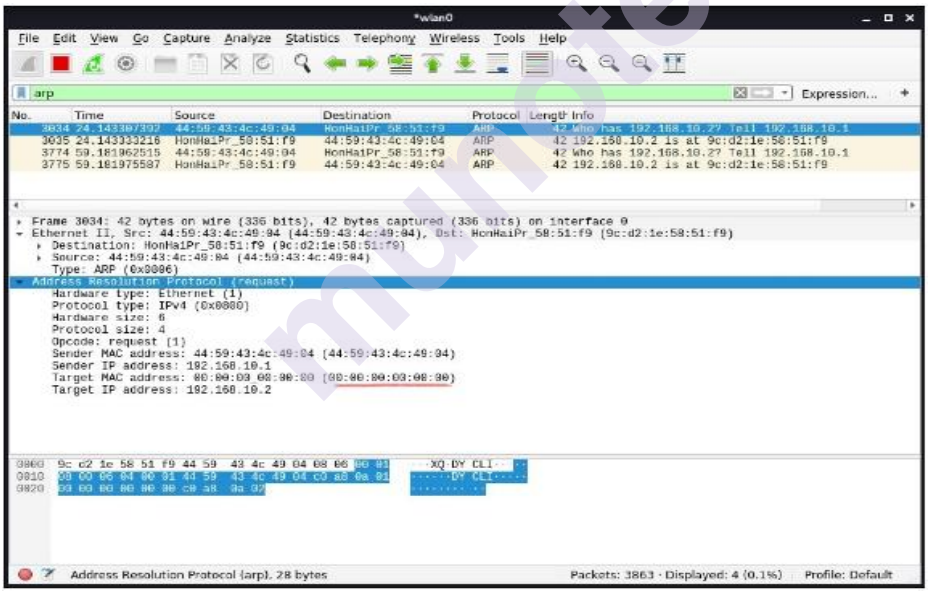
Start Wireshark data capturing, and ping the default gateway address -> Now, let's analyze what happens after removing the ARP entry and pinging a new IP address in the meantime.

Analyze an ARP Request

Using the *'arp'* filter, analyze the captured traffic in Wireshark.

Observe the packet request details from Ethernet and ARP; observe the source and destination IP and sender MAC and IP address.

Monitor the victim's MAC address. Since the destination MAC address is unavailable at the request packet stage, the victim's MAC address is zero, and the destination IP is the local system IP address.

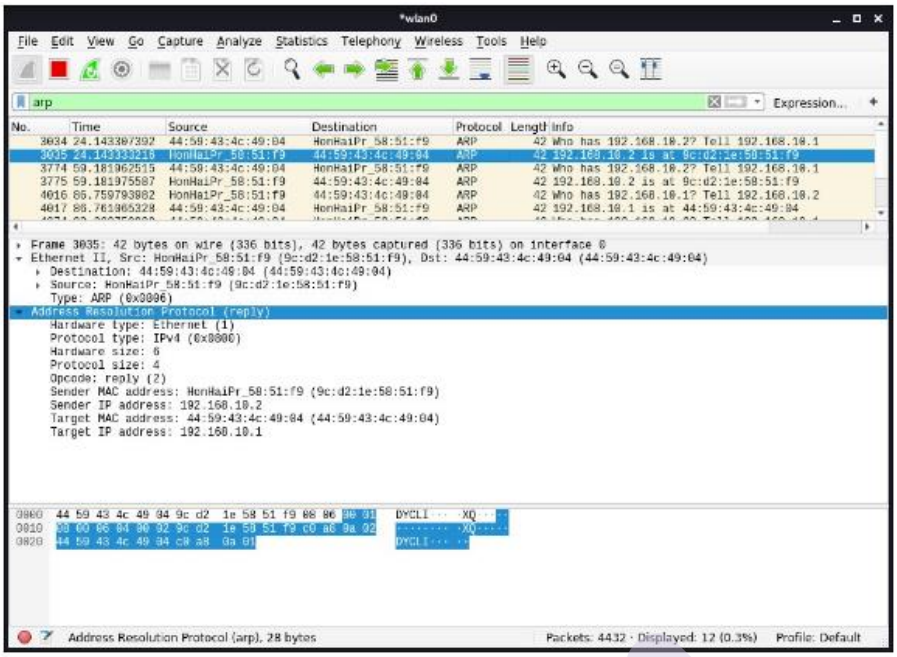


Analyze an ARP Response

Observe the packet replay details from Ethernet and ARP; observe the change in source and destination IP and MAC addresses.

The destination and source MAC address are switched in the response packet.

Everything is similar as before, except the target MAC address, which was all zeroes before. Now, that has turned into your MAC address.

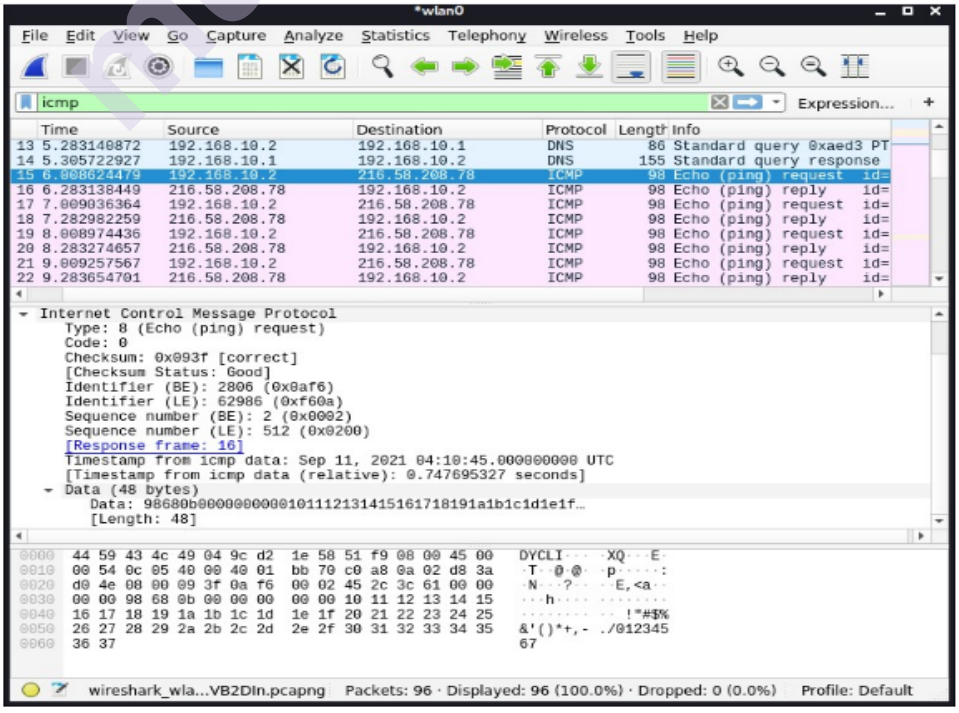


4.13 ICMP TRAFFIC ANALYSIS

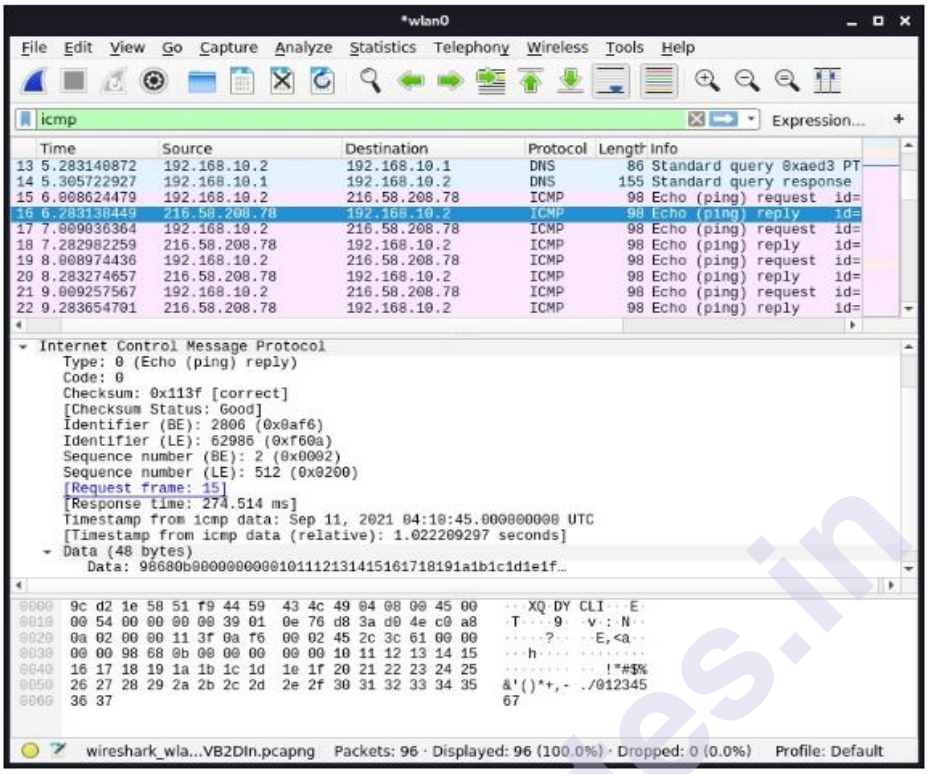
ICMP is used for error alerting and monitoring to verify whether data arrives in a timely basis at its desired destination.

To capture ICMP traffic, ping Google.com. Use the *'ICMP'* filter to see ICMP traffic. Click the ICMP echo-request packet from the Wireshark capture window and start observing the information.

In the ***request packet,*** the source IP is your (requestor) IP address. Whereas the destination IP is that of Google. You can also analyze the ICMP details like Checksum, Identifier Number, Sequence Number, etc.



In the ***response packet***, observe the swapping of IPs between source and destination. You can also compare both request and response details, as they are similar.

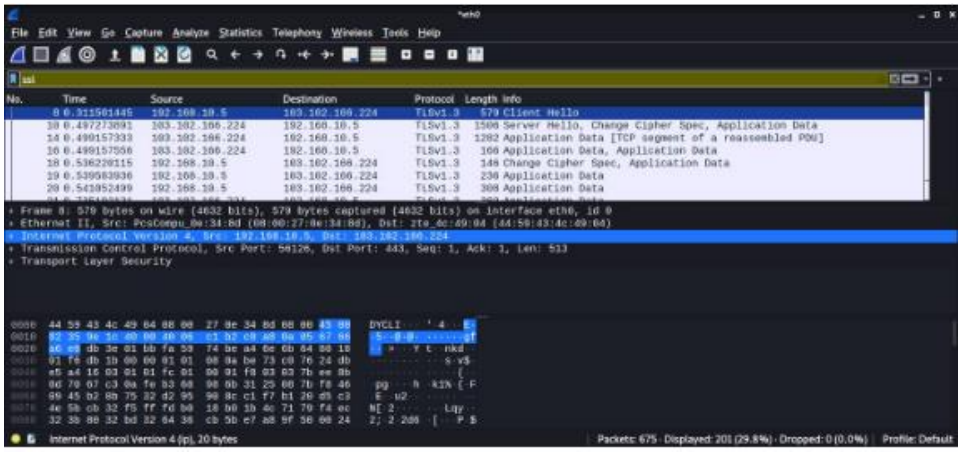


4.14 HTTPS TRAFFIC ANALYSIS

The Hypertext Transfer Application Layer Protocol (HTTP) utilizes the internet to establish protocols whenever the HTTP client/server transmits/receives HTTP requests.

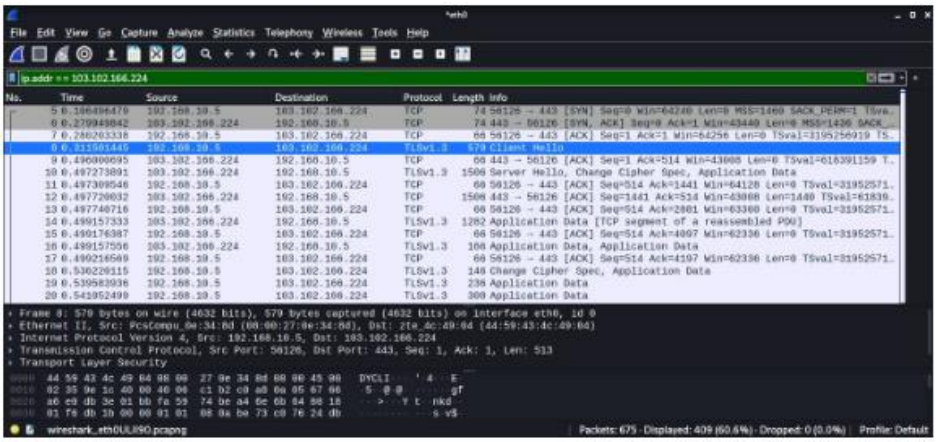
Start a Wireshark capture -> Open a web browser -> Navigate to any HTTPS-based website -> Stop the Wireshark capture.

Input ' ssl' in the filter box to monitor only HTTPS traffic -> Observe the first TLS packet -> The destination IP would be the target IP (server).



To see more traffic of the target IP (destination IP), input the following filter

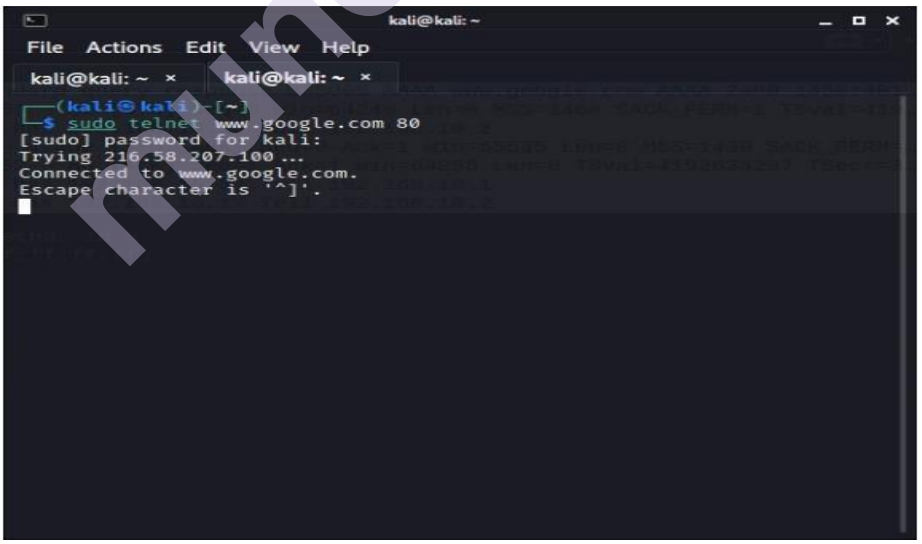
ip.addr ==



4.15 TCP TRAFFIC ANALYSIS

A standard port scan takes advantage of the TCP three-way handshake. The attacker sends the SYN packet to the target port. The port is considered open when he gets SYN+ACK as a response, whereas the arrival of RST shows the port is closed. After receiving SYN+ACK, the hacker would send an ACK packet to establish a TCP connection.

Let's analyze a TCP network traffic using telnet on Google port 80. Capture the Wireshark traffic while entering the telnet command.



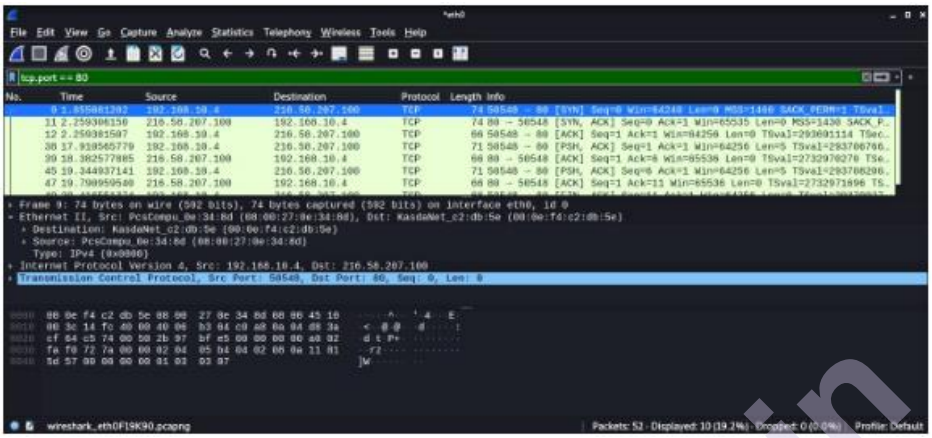
Analyze TCP SYN traffic

Input *'tcp.port == 80'* to see only TCP traffic connected to the webserver connection.

Observe the TCP [SYN] packet. Expand Ethernet and observe the destination address that is the default gateway address; whereas, the source is your own MAC address.

To check the IP details, observe Internet Protocol Version 4; in our case, the destination IP is Googles' web server IP, and the source IP is the local IP address.

To view TCP details, observe Transmission Control Protocol, like port numbers. Monitor the flag values. SYN, which is enabled, shows the initial section of the TCP three-way handshake.

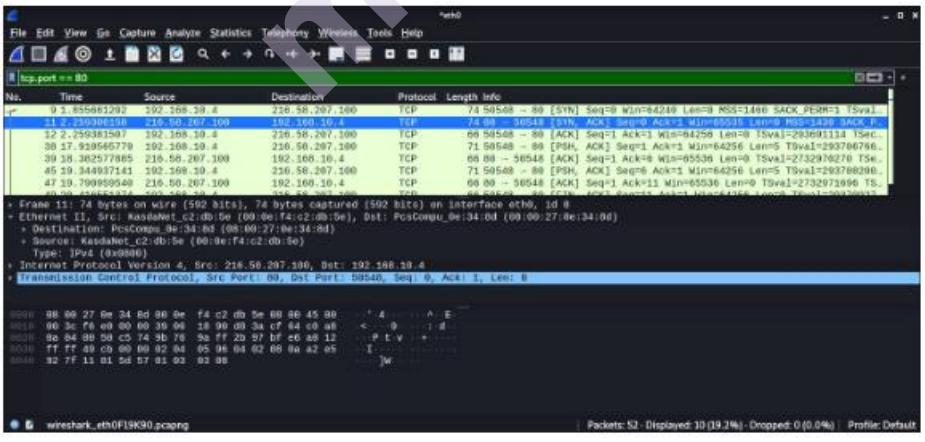


Analyze TCP SYN, ACK traffic

Take a look at the TCP [SYN, ACK] packet. Expand Ethernet and observe the destination address now would be your own MAC address; whereas the source is the default gateway address.

Monitor the acknowledgement code. It's worth noting that the number is one relative ACK number. The real acknowledgement value is one higher than the previous segment's identifier.

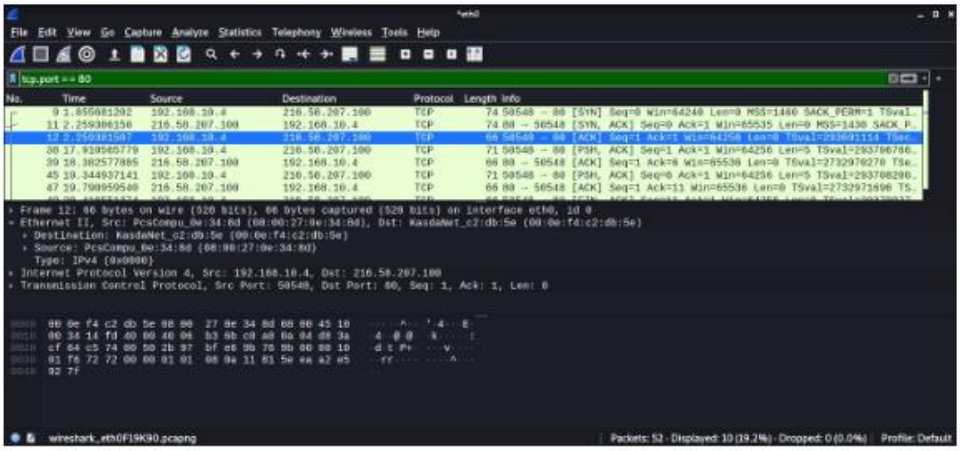
Monitor the flag values. [SYN, ACK], which is enabled, shows the second section of the TCP three-way handshake.



Analyze TCP ACK traffic

Now consider the TCP [ACK] packet. Expand Ethernet and observe the destination address that is the default gateway address; whereas the source is your own MAC address. To view TCP details like port numbers, expand Transmission Control Protocol.

The enable ACK flag signals that the TCP three-way handshake has reached the last phase. The client and server have started a TCP session.

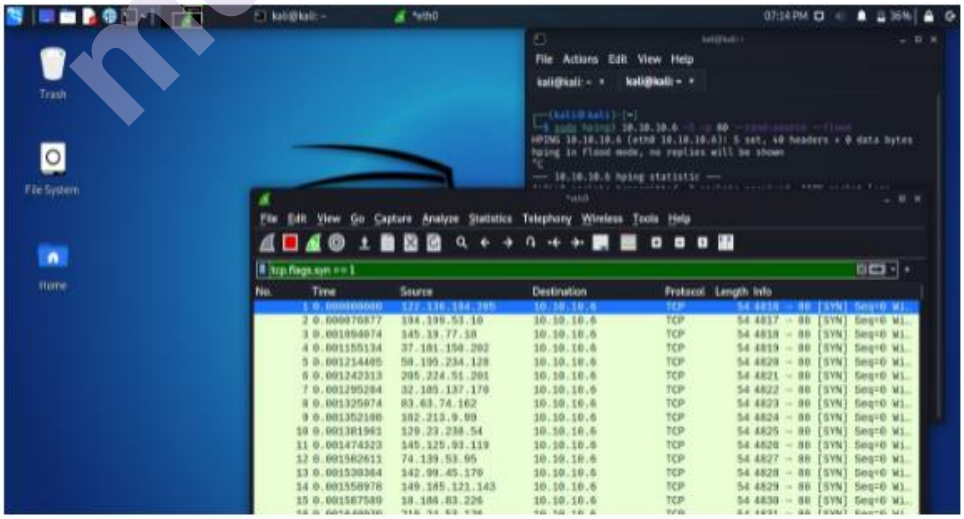


4.16 ANALYZE SYN FLOOD ATTACK

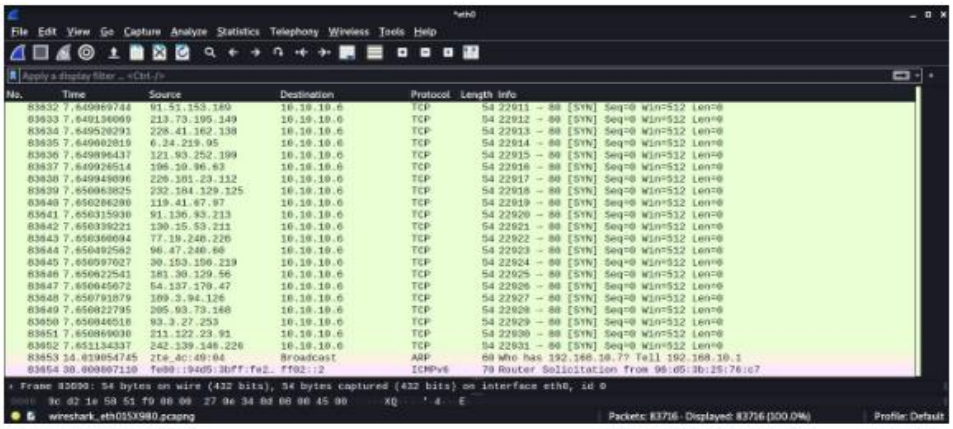
SYN flood occurs when an attacker delivers a substantial amount of SYN packets to a server using fake IPs, causing the server to respond with an SYN+ACK and keep its ports partially open, expecting a response from an invisible client.

By overwhelming a victim with SYN packets, an attacker can effectively overrun the victim's resources. In this state, the victim fights with traffic, which causes processor and memory usage to rise, eventually exhausting the victim's resources.

Use the hping3 tool to flood the victim IP. Simultaneously, start capturing the traffic on Wireshark. Input 'tcp.flags.syn == 1' in the filter box to view SYN packets flood.



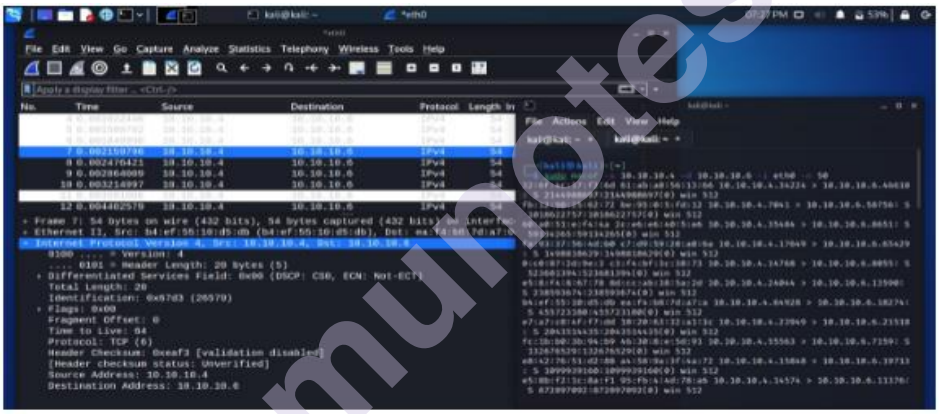
Notice a lot of SYN packets with no lag time.



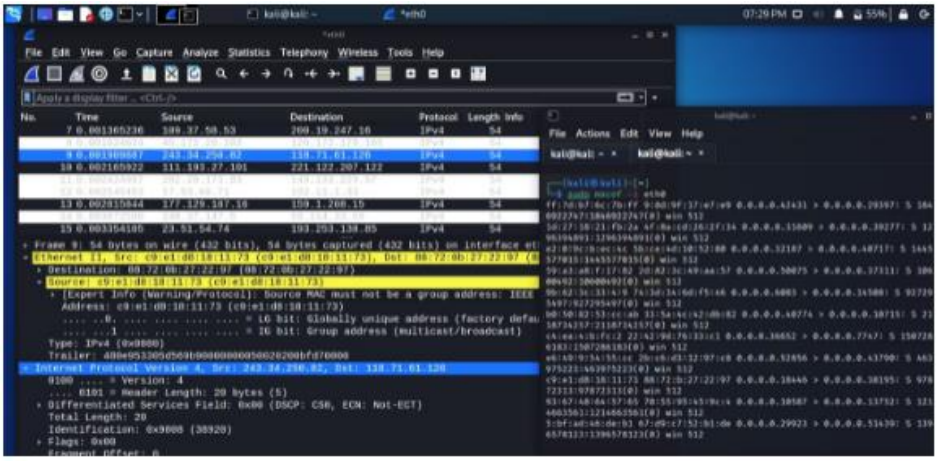
4.17 ANALYZE DOS ATTACKS

Analyze Denial of Service (DoS) attack via Wireshark. We will be using the macof tool, the component of the Dsniff suit toolkit, and flooding a surrounding device's switch with MAC addresses.

The image below shows IP address is generating requests to another device with the same data size repeatedly. This sort of traffic shows a standard network DoS attack.



For a DDoS attack, use the macof tool again to generate traffic. Observe the fake source and destination IP addresses are sending many packets with similar data sizes.



Conclusion

Wireshark is an essential tool that numerous network administrators use daily to analyze network traffic. In this section, we explored numerous network traffic types like HTTPS, TCP, etc. and have seen few attacks using Wireshark, like the DoS attack.

4.18 CAPTURE AND ANALYZE DATA PACKETS USING WIRESHARK

One of the major functions of Wireshark is to capture data packets for conducting detailed network analysis. There are majorly three important steps:

- Get access to administrative privileges to start capturing the real-time data directly the device
- Choose the right network interface to capture packet data
- Choose the right location within the network to capture packet data

After following the above steps, the Wireshark is ready to capture packets. Usually, there are two capturing modes: promiscuous and monitor. Promiscuous mode sets the network interface to capture only the packets for which it's assigned. Monitor mode is used by Unix/Linux systems and sets the wireless interface to capture as much of the network as it can. The data gathered while capturing packets is displayed in a human-readable format, so it's easier to grasp. As Wireshark breaks the captured packets into a readable format, users can perform various other tasks such as select filters to find precise information and color code the crucial information. With Wireshark, administrators can also monitor multiple networks simultaneously.

Usually, promiscuous mode is used by system administrators to get a bird's-eye view of the network packets transfer. On disabling this mode, only a small snapshot of the network is provided, which isn't enough to conduct quality analysis. The promiscuous mode can easily be activated by clicking on the capture options provided in the dialog box. However, promiscuous mode isn't available on every software or operating system. Therefore, users need to cross confirm about software compatibility either by visiting the Wireshark's website or using the Device manager to check the settings (in case you're a Windows user).

4.19 ANALYZE THE CAPTURED NETWORK PACKETS

Once the network data is captured, the list of network packets is displayed on the centralized management console or the dashboard. The screen consists of three panes: packet list, packet bytes, and packet details.

Packet List

The packet list pane consists of browsable captured network packets based on the timestamp to show exactly when the packet was captured, the

packet's protocol name, the source and destination of the packet, and support information.

Packet Details

Packet details pane provides information about the chosen packet. Users can expand each section and apply filters to get information about specific items.

Packet Bytes

Packet bytes pane consists of the internal data of the packet the user selects. The data is displayed in a hexadecimal format by default.

HOW DOES WIRESHARK HELP IN MONITORING NETWORK PERFORMANCE?

Once the network packets are captured, the next step is to monitor and analyze them. Wireshark is a great tool to analyze and monitor the organization's active network. Before doing this, it's important to close down all the active applications on the network to reduce traffic, which helps in giving a clear picture of the network.

4.20 VIEW NETWORK STATISTICS ON WIRESHARK

Statistical data is beneficial in providing in-depth information about your network. To view those stats, click on the statistics drop-down menu providing metrics ranging from timing and size to plotting graphs and charts. Users can also apply a display filter to narrow down the list of options and find out the relevant information. The drop-down statistics menu displays the following metrics:

- **Conversations:** Displays the conversations of two endpoints like two different IP addresses
- **Endpoints:** Displays the list of endpoints
- **IO Graphs:** Displays all graphs
- **Protocol Hierarchy:** Displays the table of captured packets
- **RTP_statistics:** Help users to save RTP audio stream content to Au-file
- **Multicast Stream:** Measures the speed of other components by identifying the multicast streams
- **TcpPduTime:** The time taken to transmit data from Data Protocol Unit
- **VoIP_Calls:** The number of VolP calls received from live captures
- **Service Response Time:** The time taken by the network to respond to a request

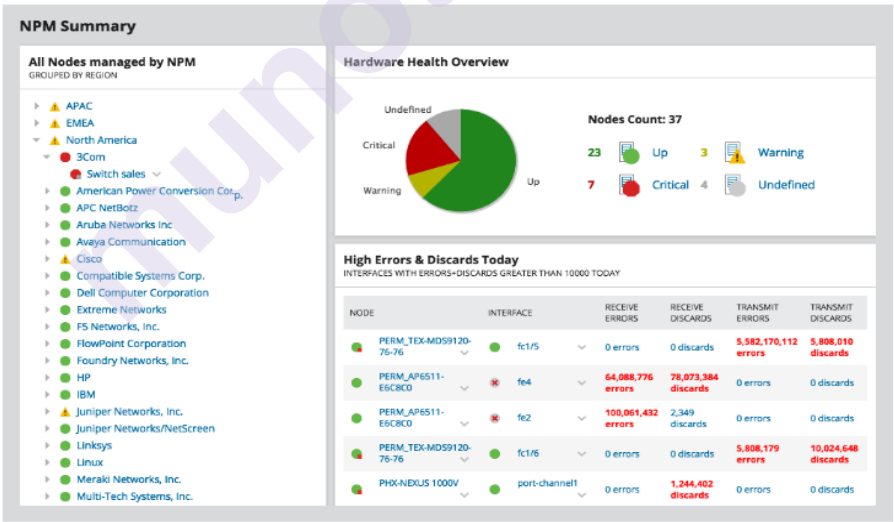
4.21 VISUALIZE NETWORK PACKETS WITH IO GRAPHS

Data packets or the network traffic can be represented in a visual format known as IO graphs. Users can view IO graphs by clicking on the statistics menu and selecting the IP graph tab. Once the graph window opens, users can view the data they wish for by configuring the graph settings accordingly. Default settings, allows only one graph getting displayed at a time. Activation of more graphs simultaneously can be done by simply enabling them. Users can customize the graphs using filters and color codes to find out the relevant information as needed. As the settings are done, the graphs can be stored in a file format for future purposes as required.

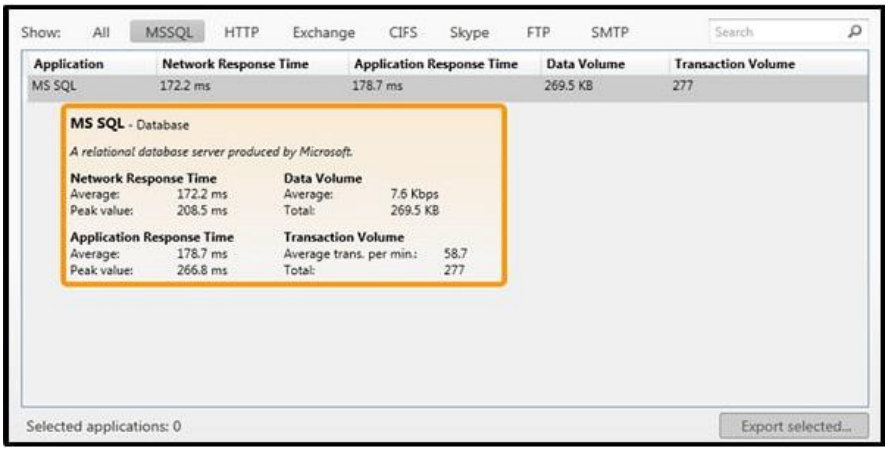
EXPAND WIRESHARK CAPABILITIES

Wireshark is a great packet sniffer, but it lacks as it comes to meet the growing needs of network monitoring. Monitoring capabilities can be improved using complementary tools such as SolarWinds® Response Time Viewer for Wireshark, Show Traffic, Cloudshark, and NetworkMiner. Outlined below are the features, capabilities, functions of these tools, and how they help in enhancing the network monitoring capabilities of Wireshark.

SolarWinds Network Performance Monitor



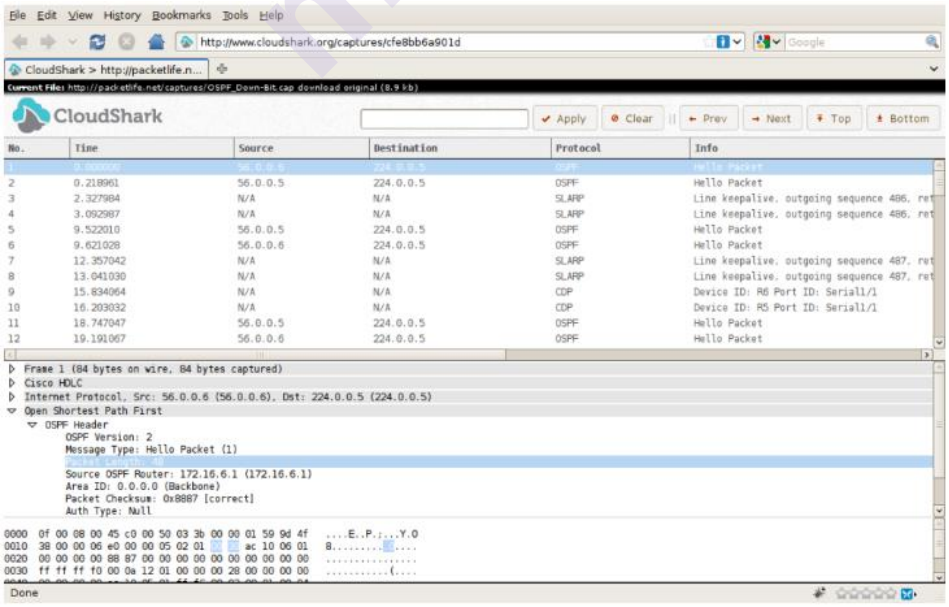
SolarWinds Network Performance Monitor (NPM) is a powerful network monitoring software used to detect, diagnose, and resolve network issues and outages. The tool is specially designed for advanced network troubleshooting for on-premises, hybrid, and cloud services, and detects issues with its hop-by-hop analysis. NPM is also capable of reducing downtime resulting in improved operational efficiency, network availability, and high-performing applications. It offers intuitive dashboards, advanced alerting systems, custom reporting, packet analysis, and more.



SolarWinds Response Time Viewer for Wireshark allows users to detect and analyze Wireshark's packet captures and troubleshoot network performance outages in real-time and can perform multiple tasks such as identify over 1200 applications, calculate their network response time, display data and transaction value, critical path visualization with Netpath, and wireless network monitoring and management. Evaluating these parameters helps users determine network flaws and faults.

Cloudshark

Cloudshark is a platform designed to display network capture files directly in the browser without the need for desktop applications or tools. Simply upload, email, or link the captures files and get the results. It helps to resolve network issues faster and flawlessly. Besides, it includes features such as drag-and-drop capture files, drop-box-like activity, allows sharing of links with co-workers, and offers advanced analysis.



Sysdig

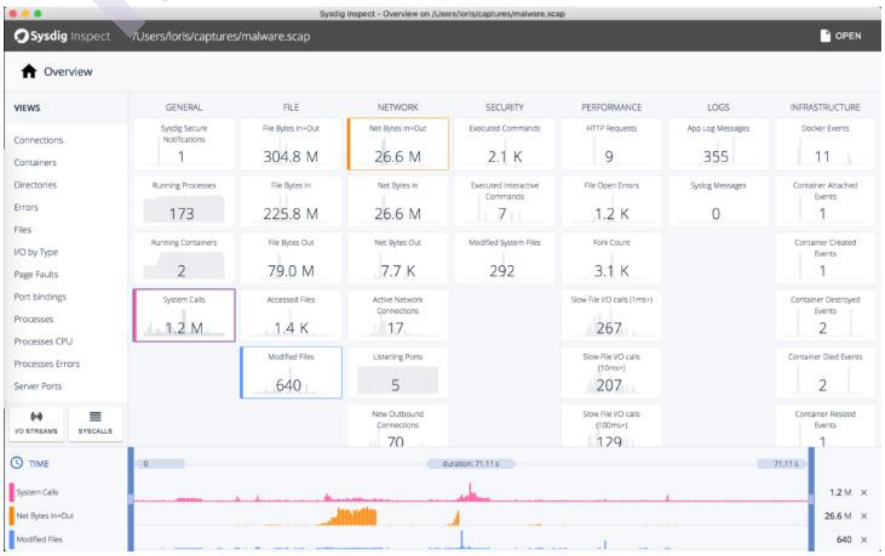
Sysdig is a powerful, cross-platform, and open-source flexible network monitoring system designed specifically for Linux. It works with others but with limited functionalities. Sysdig uses various monitoring and troubleshooting tools for Linux system performance, such as:

- Strace (for discovering system calls)
- htop (for real-time process monitoring)
- iftop (for real-time bandwidth monitoring)
- netstat (for network connection monitoring)
- tcpdump (for raw network traffic monitoring)
- Isof (to know the process used to view the opened files)

Sysdig integrates all the tools mentioned above and offers them in a single program. Users can easily capture, filter, save, modify, track, and examine network traffic and the real behavior of Linux systems.

Sysdig features include:

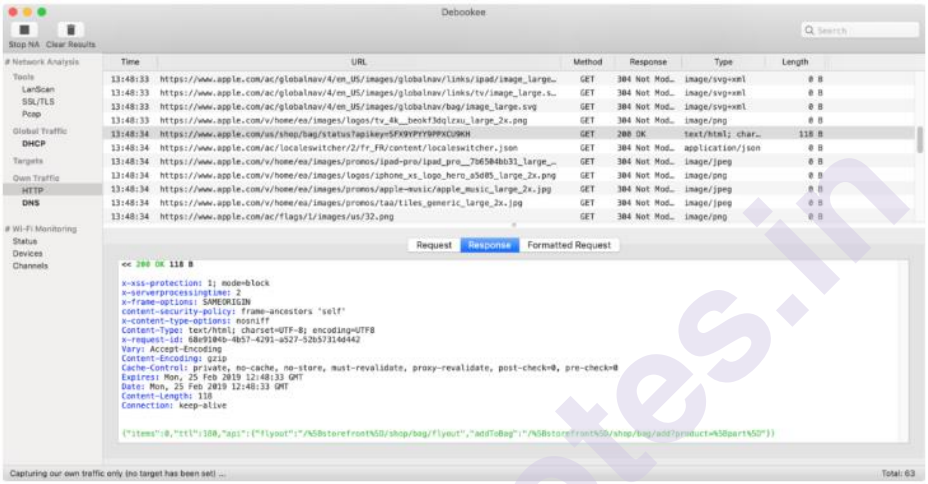
- Orchestrator integrations
- Intuitive dashboards
- Cloud-platform integrations
- Application integrations
- Alerting and event management
- Vulnerability management
- Compliance/Audit report generation
- Topology maps
- Runtime error detection
- Capture files
- Single sign-on



Debooke

Debooke is one of the simplest and most powerful network monitoring and traffic analyzer tools for macOS. It allows users to intercept and monitor the network traffic of any device in the same subnet. It helps to capture data network packet data from mobile, printer, Mac, TV. Features of Debooke:

- Keep track of Wi-Fi bandwidth use and consumption
- Help with network monitoring and in-depth analysis
- Display the Wi-Fi clients and the APIs to which they're associated
- Scan IP, LAN to keep track of all the active and connected devices



4.22 BASIC CONCEPTS OF THE NETWORK TRAFFIC

IP Addresses: It was designed for the devices to communicate with each other on a local network or over the Internet. It is used for host or network interface identification. It provides the location of the host and capacity of establishing the path to the host in that network. Internet Protocol is the set of predefined rules or terms under which the communication should be conducted. The types of IP addresses are **IPv4 and IPv6**.

- IPv4 is a **32-bit address** in which each group represents 8 bits ranging from 0 to 255.
- IPv6 is a 128-bit address.

IP addresses are assigned to the host either dynamically or static IP address. Most of the private users have dynamic IP address while business users or servers have a static IP address. Dynamic address changes whenever the device is connected to the Internet.

Computer Ports: The computer ports work in combination with the IP address directing all outgoing and incoming packets to their proper places. There are well-known ports to work with like **FTP** (File Transfer Protocol), which has port no. 21, etc. All the ports have the purpose of directing all packets in the predefined direction.

Protocol: The Protocol is a set of predefined rules. They are considered as the standardized way of communication. One of the most used protocol is **TCP/IP**. It stands for **Transmission Control Protocol/ Internet Protocol**.

OSI model: OSI model stands for **Open System Interconnect**. OSI model has seven layers, namely, **Application layer, Presentation layer, Session layer, Transport layer, Network layer, Data link layer, and the physical layer**. OSI model gives a detail representation and explanation of the transmission and reception of data through the layers. OSI model supports both connectionless and connection-oriented communication mode over the network layer. The OSI model was developed by ISO (International Standard Organization).

Most used Filters in Wireshark

Whenever we type any commands in the filter command box, it turns **green** if your command is **correct**. It turns **red** if it is **incorrect** or the Wireshark does not recognize your command.

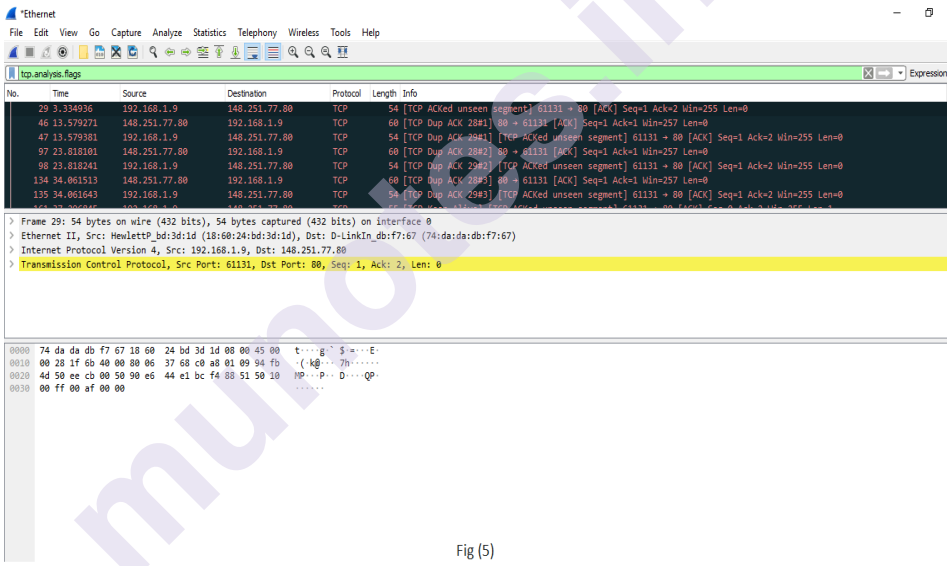


Fig (5)

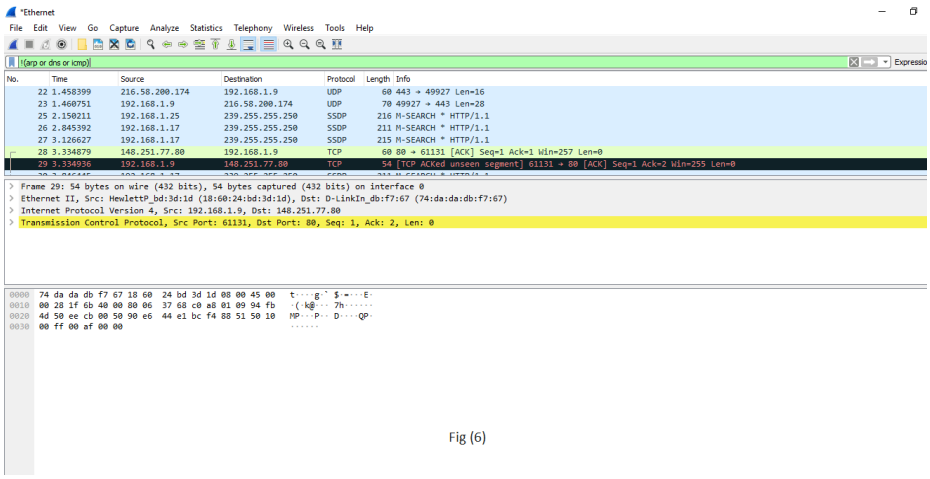


Fig (6)

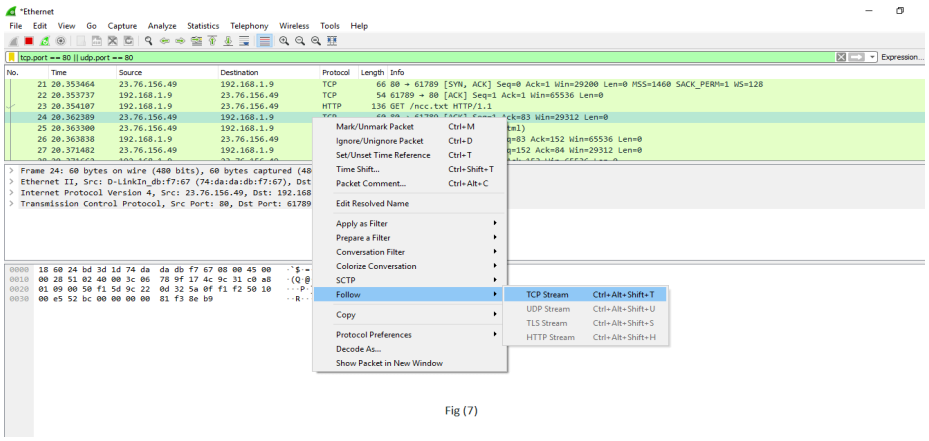


Fig (7)

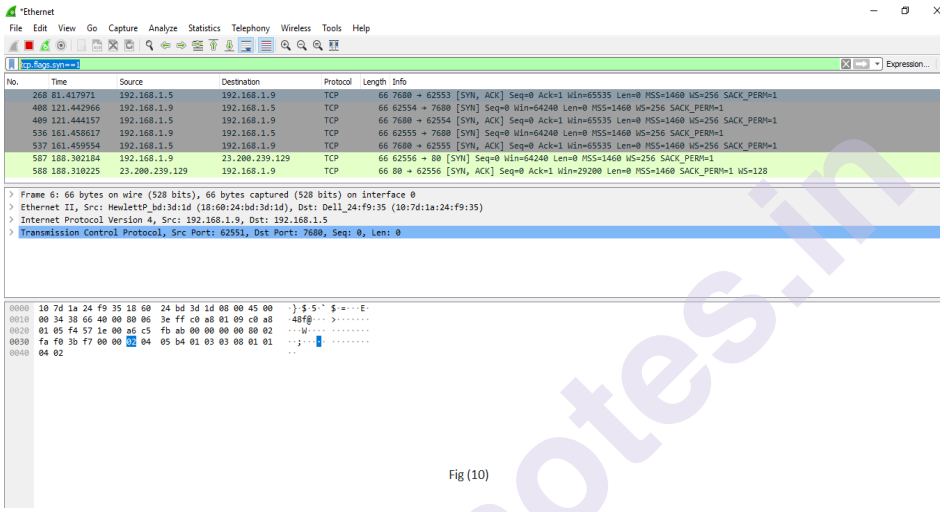


Fig (10)

Below is the list of filters used in Wireshark:

Filters	Description
ip.addr Example- ip.addr==10.0.10.142 ip.src ip.dst	It is used to specify the IP address as the source or the destination.
protocol Example- dns or http 'Dns and http' is never used.	This command filters based on the protocol. It requires the packet to be either dns protocol or http protocol and will display the traffic based on this.
tcp.port Example: tcp.port==443	It sets filter based on the specific port number.
4. udp.port	It is same as tcp.port. Instead, udp is used.
tcp.analysis.flags example is shown in fig(5).	Wireshark can flag TCP problems.
6.!() For example, !(arp or dns or icmp) This is shown in fig (6).	It is used to filter the list of protocols or applications, in which we are not interested.
Select any packet. Right-click on it and select 'Follow' and then select' TCP stream.' Shown in fig. (7).	It is used if you want to work on a single connection on a TCP conversation.
tcp contains the filter For example- tcp contains Facebook Or udp contains Facebook	It is used to display the packets which contain such words.
http.request For the responses or the response code, you can type http.response.code==200	It will display all the http requests in the trace file.
tcp.flags.syn==1 This is shown in fig (10).	This will display all the packets with the sync built-in tcp header set to 1.
tcp.flags.reset	

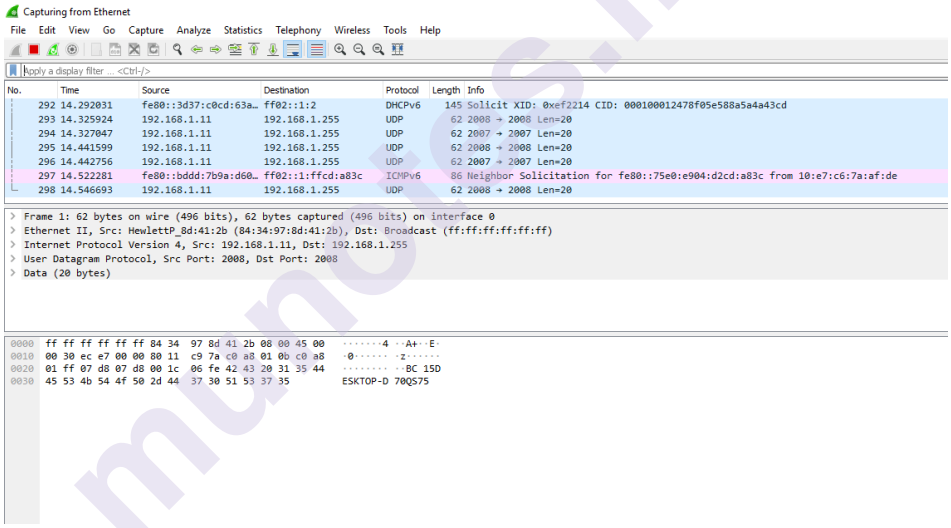
4.23 WIRESHARK PACKET SNIFFING

Wireshark is a packet sniffing program that administrators can use to isolate and troubleshoot problems on the network. It can also be used to capture sensitive data like usernames and passwords. It can also be used in wrong way (hacking) to ease drop.

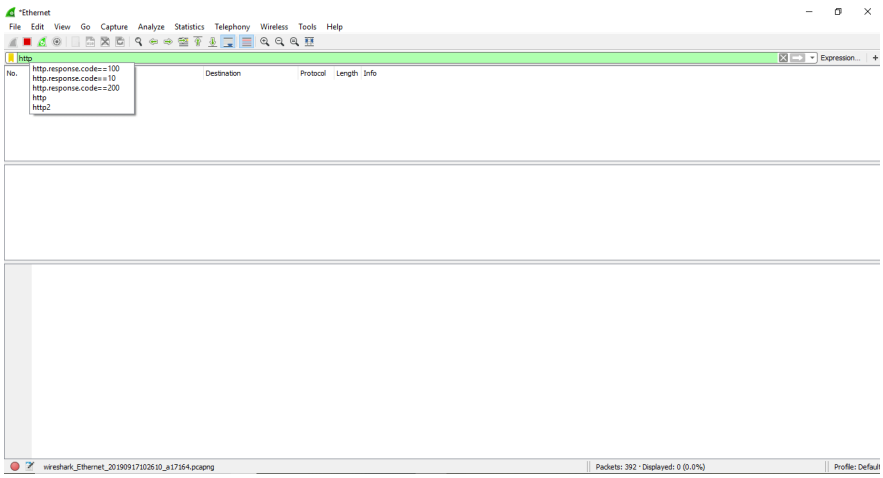
Packet sniffing is defined as the process to capture the packets of data flowing across a computer network. The Packet sniffer is a device or software used for the process of sniffing.

Below are the steps for packet sniffing:

- Open the Wireshark Application.
- Select the current interface. Here in this example, interface is Ethernet that we would be using.
- The network traffic will be shown below, which will be continuous. To stop or watch any particular packet, you can press the red button below the menu bar.



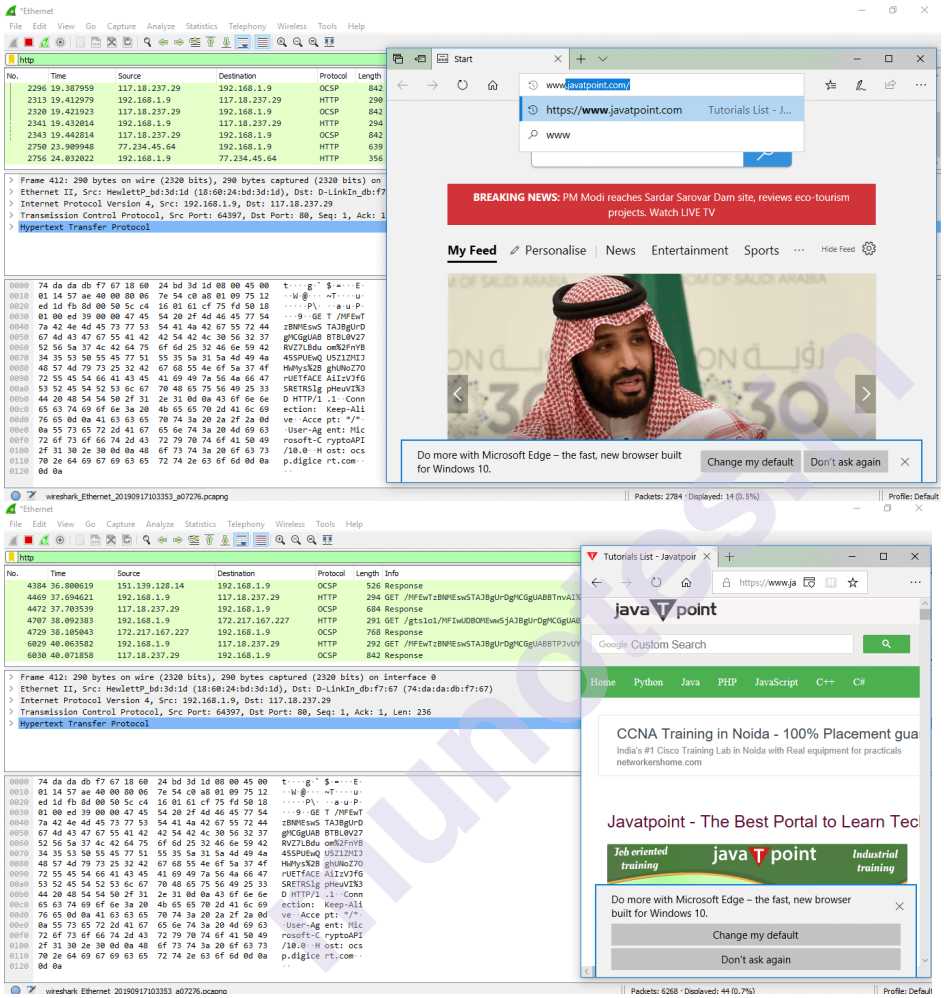
Apply the filter by the name 'http.' After the filter is applied, the screen will look as:



The above screen is blank, i.e.; there is no network traffic as of now.

Open the browser. In this example, we have opened the 'Internet Explorer.' You can choose any browser.

As soon as we open the browser, and type any address of the website, the traffic will start showing, and exchange of the packets will also start. The image for this is shown below:

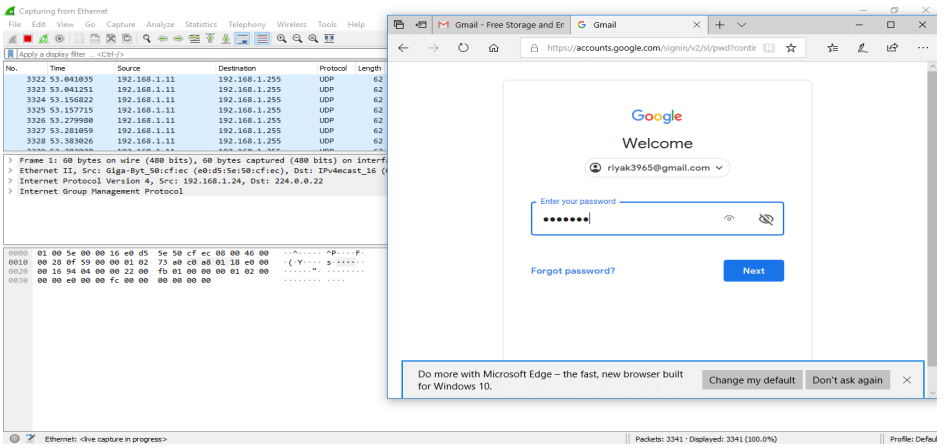


The above process explained is called as **packet sniffing**.

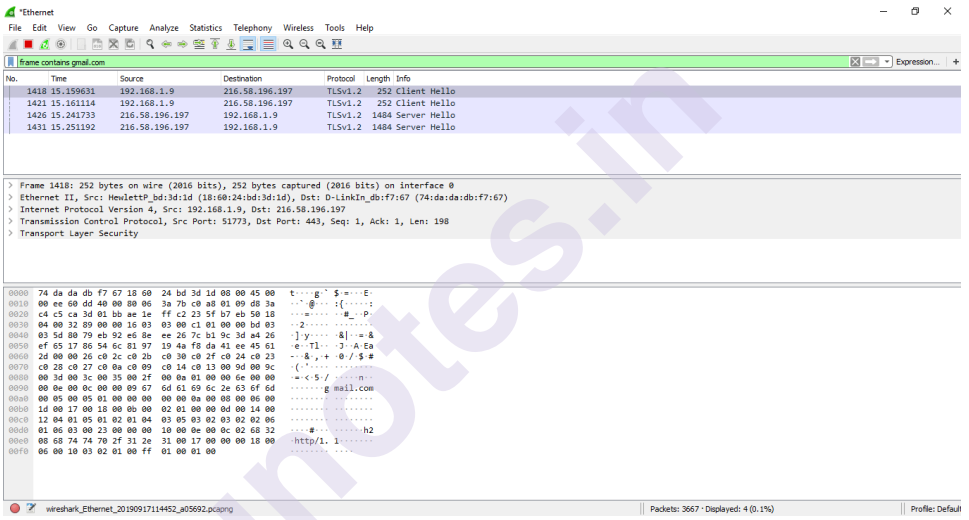
Username and password sniffing

It is the process used to know the passwords and username for the particular website. Let's take an example of gmail.com. Below are the steps:

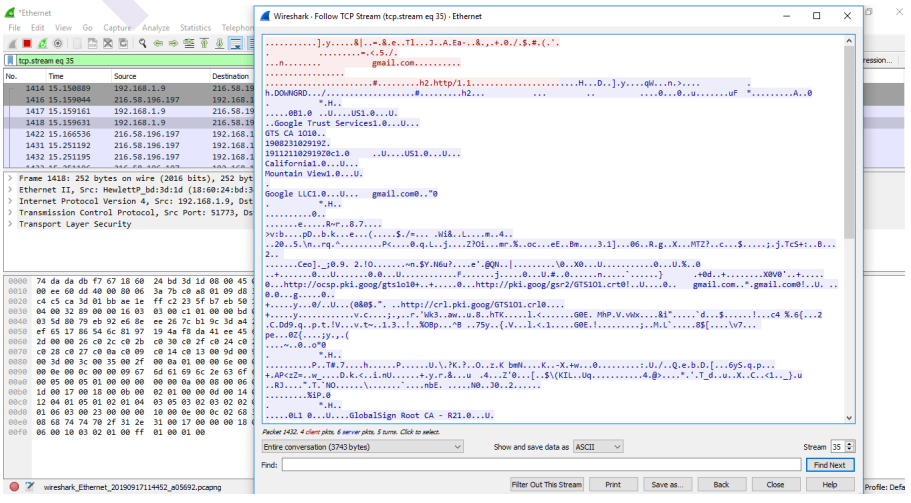
- Open the Wireshark and select the suitable interface.
- Open the browser and enter the web address. Here, we have entered gmail.com, which is highly secured. Enter your email address and the password. The image is shown below:



- Now, go to the Wireshark and on the filters block, enter 'frame contains gmail.com.' Then you can see some traffic.



- Right-click on the particular network and select 'Follow', and then 'TCP Stream.' You can see that all the data is secured in the encrypted form.



In the arrow shown above, the 'show and save data as' has many choices. These options are- **ASCII, C Arrays, EBCDIC (Extended Binary**

Coded Decimal Interchange Code), etc. EBCDIC is used in mainframe and mid-range IBM computer operating systems.

4.24 WIRESHARK STATISTICS

The Wireshark provides a wide domain of statistics. They are listed below:

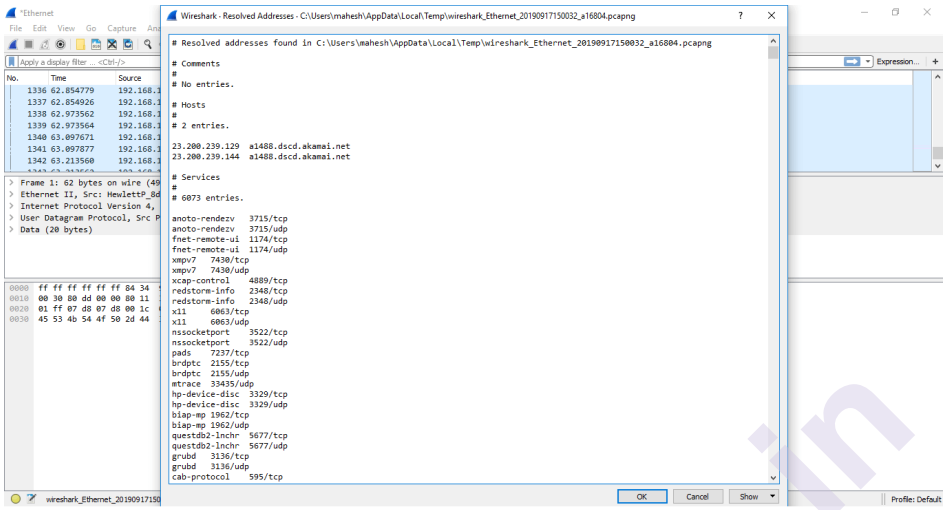


Fig (b)

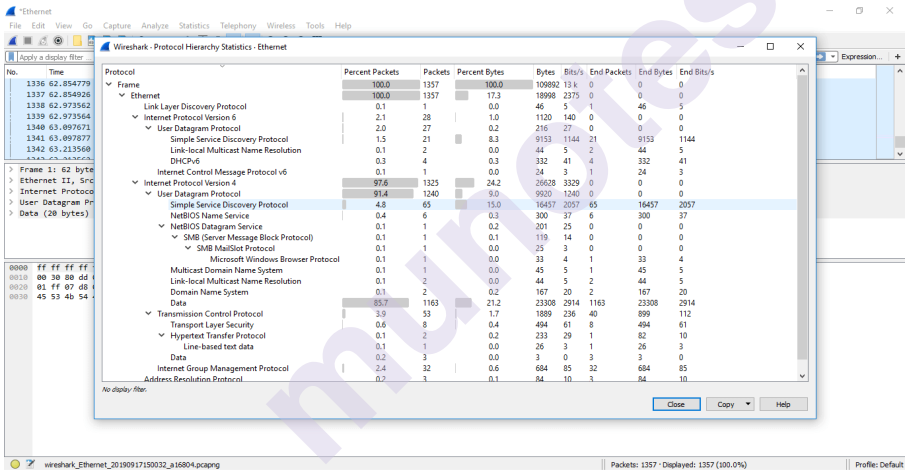


Fig (c)

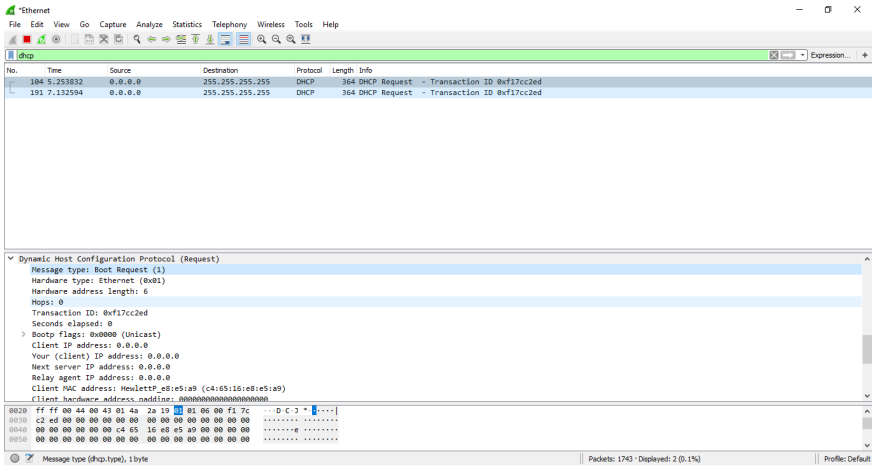
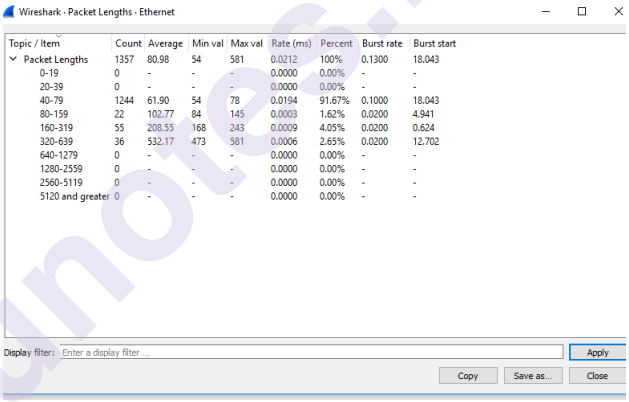


Fig (d)

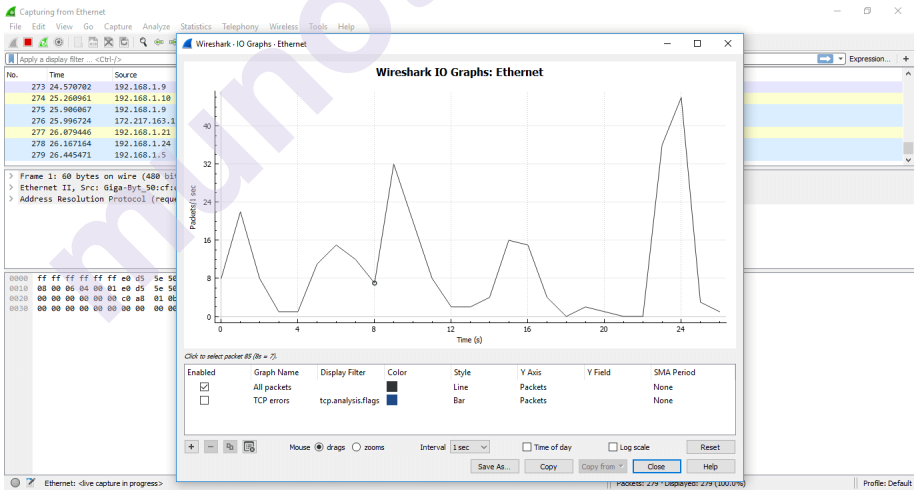
Below is the list of statistics of Wireshark along with the description:

Capture file properties	It includes file, time, capture, interfaces (current interface in use), and Statistics (measurements).
Resolved addresses	This option includes all the types of the Top IP addresses and DNS that were resolved in your packet capture. It gives the idea of the different accessed resources during the packet capture process. It is shown in fig (b).
Protocol hierarchy	It is named as the tree of all the protocols listed in the capture process. The image is shown above in fig (c).
Conversations	Each row of the list gives the statistical value of a particular conversation.
Endpoints	It is defined as a logical endpoint of the separate protocol traffic of the specified protocol layer. For example0 IP address will send and receive all types of the packet to the particular IP addresses.
Packet lengths	It simply displays the characteristics of different packets lengths determined in the network. 
I/O Graphs	It is the term used to display the graph of the captured packets. You can also apply filters during this process. The process is explained below in detail.
Service Response Time	It is the type of information which is available for many protocols. It is defined as the time it takes between the request and the response time. The protocol for which this service is available are: AFP (Apple Filing Protocol) CAMEL DCE-RPC DIAMETER FC (Fiber Channel) GTP (GPRS Tunneling Protocol) H.225 RAS LDAP (Lightweight Directory Access Protocol) MEGACO MGCP (Media Gateway Control Protocol)

	NCP (NetWare Core Protocol) ONC-RPC RADIUS SCSI SMB (Server Message Block Protocol) SMB2 (Server Message Block Protocol version 2)
DHCP (BOOTP) Statistics	It is implemented as the option of BOOTP. DHCP is client/server protocol, dynamically used to assign IP addresses to a DHCP client. If DHCP does not work, then some computer system uses APIPA (Automatic Private IP Address) to assign the IP addresses.
ONC-RPC Programs	It stands for Open Network Computing- Remote Procedure Call. It can use TCP and UDP as its transport protocol. ONC-RPC cannot be applied directly to filter in a capture process, but you can use TCP or UDP to filter on that one. It is shown in fig (d).
29West	It is defined as ULLM technology. It stands for Ultra-Low Latency Messaging.
ANCP	It stands for **Access Node Control Protocol**. It is an L2CP (Layer 2 Control Protocol) and a TCP based one. It has its adjacency layer which decides the messages exchange by the ANCP endpoints with the use of 'Capabilities.'
BACnet	It was designed specially to meet the communication needs of control systems and building automation. It is used for applications such as fire detecting systems, light control, etc. It provides the structure to exchange information despite the particular building service it performs.
Collectd	It is used to monitor the traffic on the specific TCP port.
DNS	It stands for Domain Name Server, which gives a detailed analysis of the DNS traffic. It provides the list of the codes returned in DNS. You can also view the errors through the traffic.
Flow-graph	It is a method to check connections between the client and the server. It is an efficient way to verify the connections between two endpoints. It also assists us with troubleshooting capabilities.
HART-IP	It gives the detail for the response, request, publishes, and error packets. It stands for Highway Addressable Remote Transducer over IP stats.
HPFEEDS	It determines the 'payload size per channel and Opcodes.'
HTTP	It has four options:

	○ Packet counter (request types and response codes) ○ Requests (based on URL and the host) ○ Load distribution (based on server address and host) ○ Request sequences (sequences the HTTP's capture request as a tree)
HTTP2	It is the HTTP version 2.
Sametime	It is used to analyze the slow network traffic when the server and client have the sametime.
TCP Stream Graphs	It is explained below in detail:
UDP Multicast Streams	Through this command, stream parameters and burst parameters can be set. It includes OSPF, IGMP, and video streams.
F5	It includes the virtual server distribution and the tmm distribution. It specifies the tcpdump commands.
IPv4 Statistics IPv6 Statistics	These options determine all addresses, destination and ports, IP protocol types, and the source and destination address.

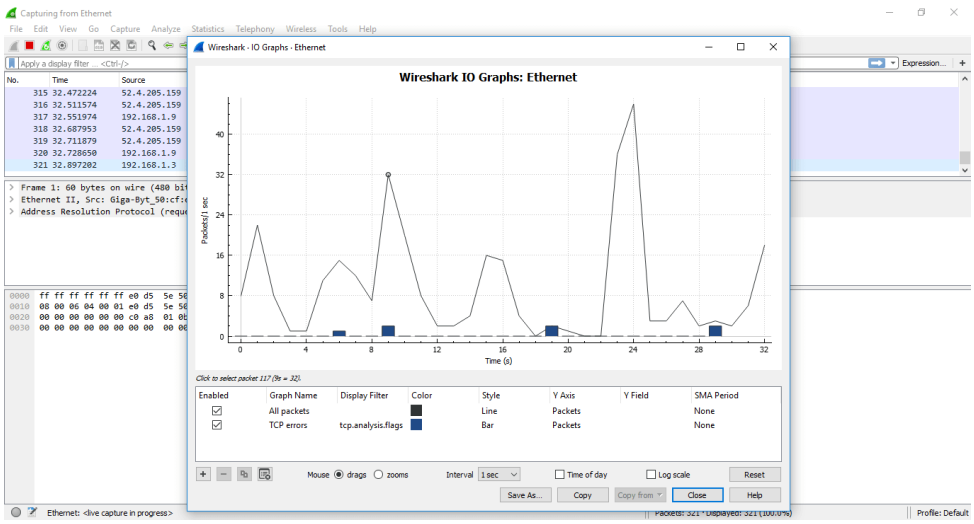
4.25 I/O GRAPHS



It shows the graph for the network traffic. The graph will look similar but changes as per the traffic involved. There is a table below the figure, which has some filters. Using the '+' sign, you can add more filters and use '-sign you can remove the existing filters. You can also change the color. For every particular filter, you can add a colored layer, which increases the visibility of the graph.

The tick option under the 'Enabled,' displays the layer according to your requirements.

For example, we have applied the filter 'TCP errors' and the changes can be viewed easily. The image is shown below:



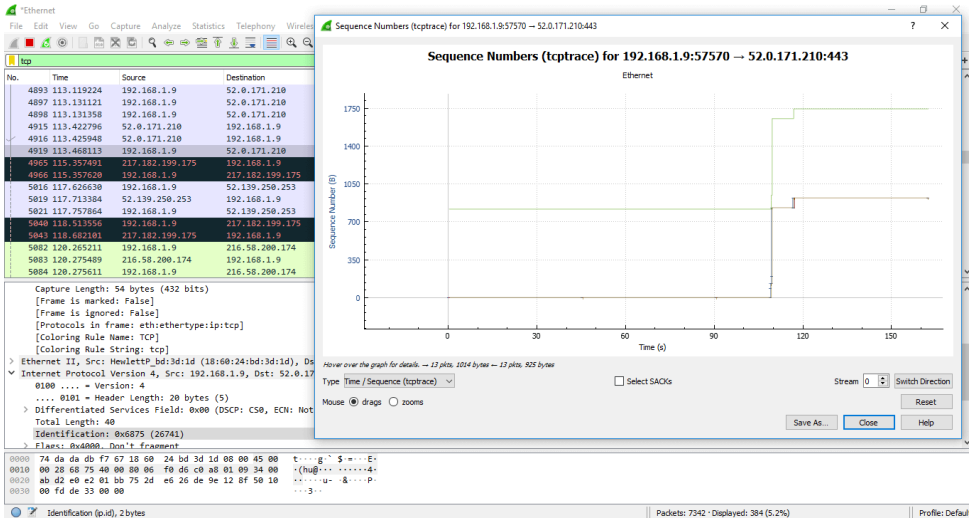
If you click on the particular point on the graph, you can watch the corresponding packet will be shown on the screen of the network traffic. You can also apply a filter on the particular port.

Another category of the graph comes under the option '**TCP Stream graphs**.'

It gives the visualization of the TCP sequence number with time.

Below are the steps to understand the **TCP Stream graphs**:

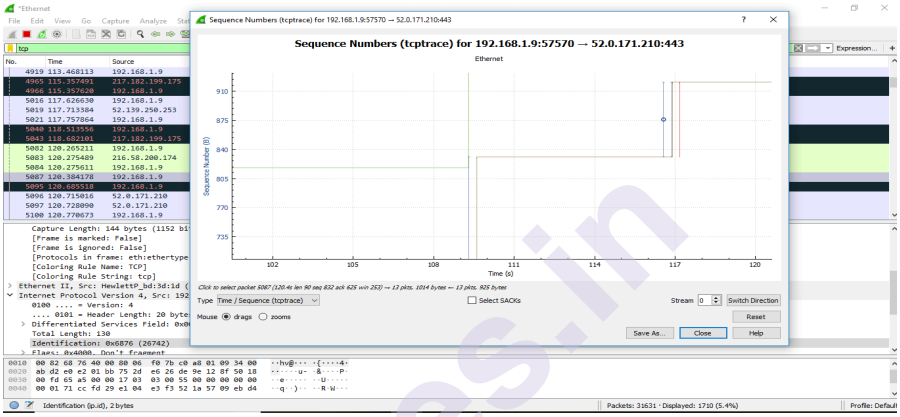
- Open the Wireshark. Click on the interface to watch the network traffic.
- Apply the filter as 'tcp.'
- Click on the option 'Statistics 'on the menu bar and select '**TCP Stream graphs**' and select 'Time sequence (tcptrace). You can also choose other options in the 'TCP Stream graphs' category depending on your requirements. Now the screen will look as:



Now, as you zoom on the graph, you will notice the points in detail. The lines shown are the packets. The length along the Y-axis shows how big the packet is. You can also see the green line going up and then comes at the same level. This means that the data has been ACK (Acknowledged). Here going up means that more data is being sent.

The data is being sent and then ACK, this is the proper use of the TCP. The flat line here signifies that nothing is happening.

The green line above is called '**received window**.' The gap between the received window and the packet, defines how much space is in the received buffer.



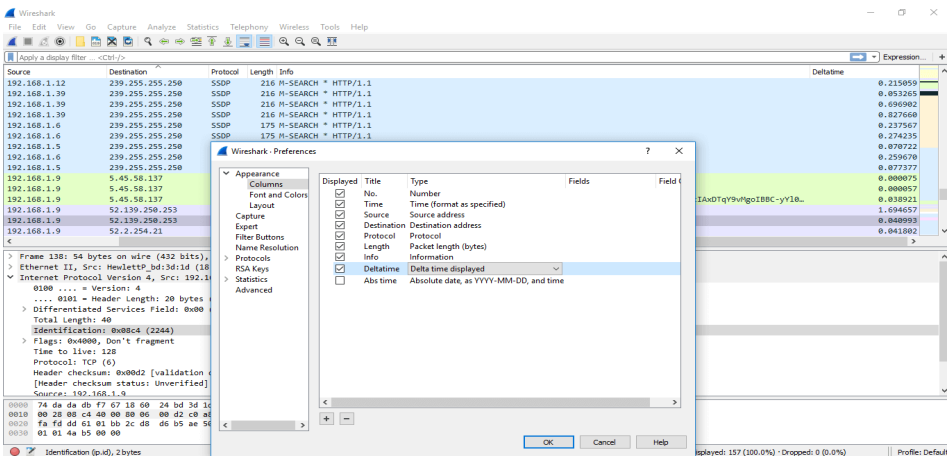
4.26 FACTS ABOUT WIRESHARK

Below are the facts or points implemented in real life:

Adding a delta column: To add any column, below are the steps:

- On any of the column menu, right-click and choose 'Column Preferences' and then select 'Column.'
- Click on the '+' sign, and add the column by name like delta-time and under the 'Type' category, select the delta time or delta time displayed.

The screen will then look as:

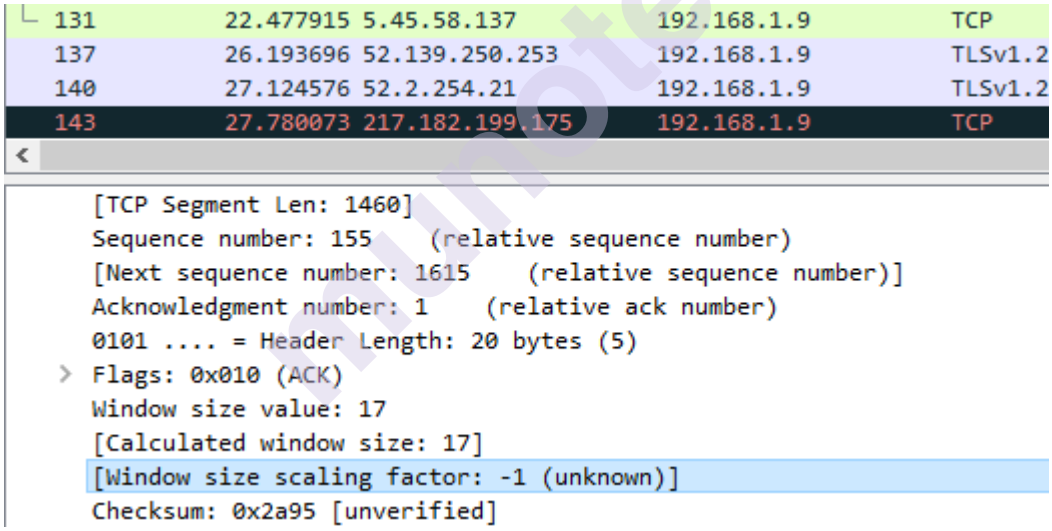


Below the captured packets, the data you see in the **square brackets** is the information that is not available in the packet itself. It is something that Wireshark displays for your benefit. If you want to add anything from this screen to the column area, you can right-click and select 'Apply as column.' That option will be added to the capture screen.

The most important is:

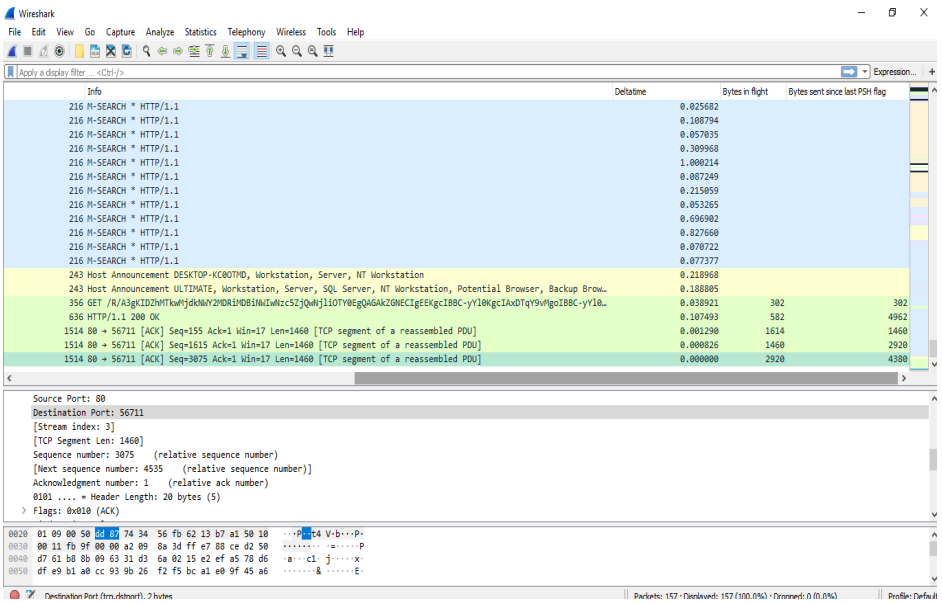
3 Way-Handshake

- When you are capturing your data, analyze the problem, you will get the three-way handshake.
- It contains good options like the TCP options.
- From this, you can determine the shift time and figure out if you have captured packets on the client-side or the server-side. There is a little delay between SYN and SYN- ACK packet at server-side while there is a more delay between the SYN and SYN-ACK at the client-side. There is a delay at the server-side only between the SYN-ACK and ACK. The SYN has to reach to the client. After the three-way handshake, the data has to reach the server.
- We can also notice the difference in the TCP options between the SYN and SYN-ACK packets. The window scaling factor is also essential, as shown below:

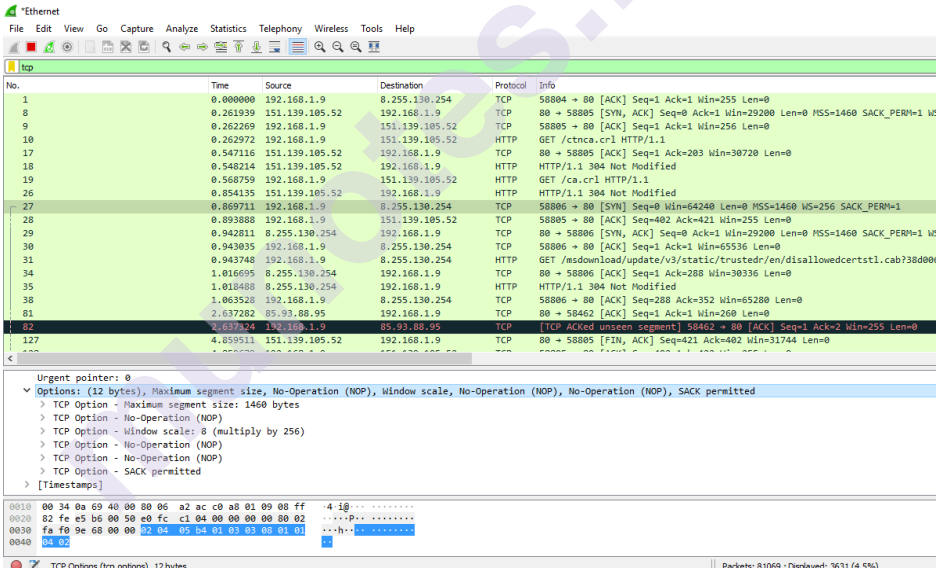


Without three-way handshake, you cannot view the window scaling factor.

- One sequence number means 1 byte of data. It also has an importance of the TCP Stream Graphs which is already explained above.
- Under the TCP options, capture window, you can see the information about the 'PSH byte' and 'Bytes in flight.' Right-click on that and choose 'Apply as Column.' You can see both the columns and data according to it. The image for this is shown below:



- In TCP Header, three-way handshake MSS (Maximum Header Size) means that the maximum amount of data it can receive of TCP payload. The image is shown below:



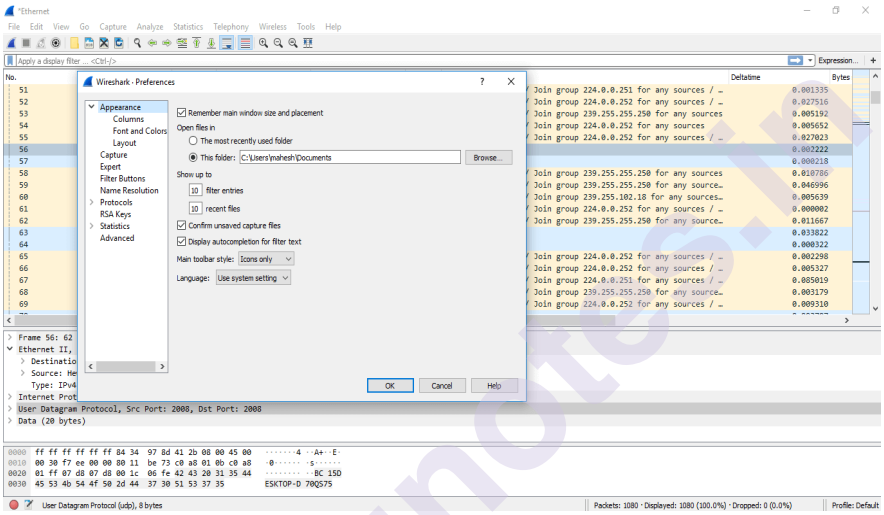
- MSS 1460 implies that this is per packet amount of data. This size varies from packet to packet. Something like a router, firewall, etc. will do MSS clamping because it knows what is going forward. It checks the value greater than 8000 bytes and brings down it to an appropriate level so that it can go across without fragmentation or being dropped.
- The data with the 0 is the ax coming back in the capture window. You can notice that the data and ACK are different at each point. If we are on the acknowledgment side, we know that we have to send the ACK after two packets. A sender can send X amount of packets depending on its congestion window. A sender can send packets at once also. After the packets will go at the receiver and then the acknowledgment comes back. The sender can send all packets

before the ACK reaches it. If the buffer has less space left, then the sender has to send the packets according to space. The ACK arrives on time, and if there is a delay in the ACK, syncing will be delayed. So above it's, just a perspective example explained.

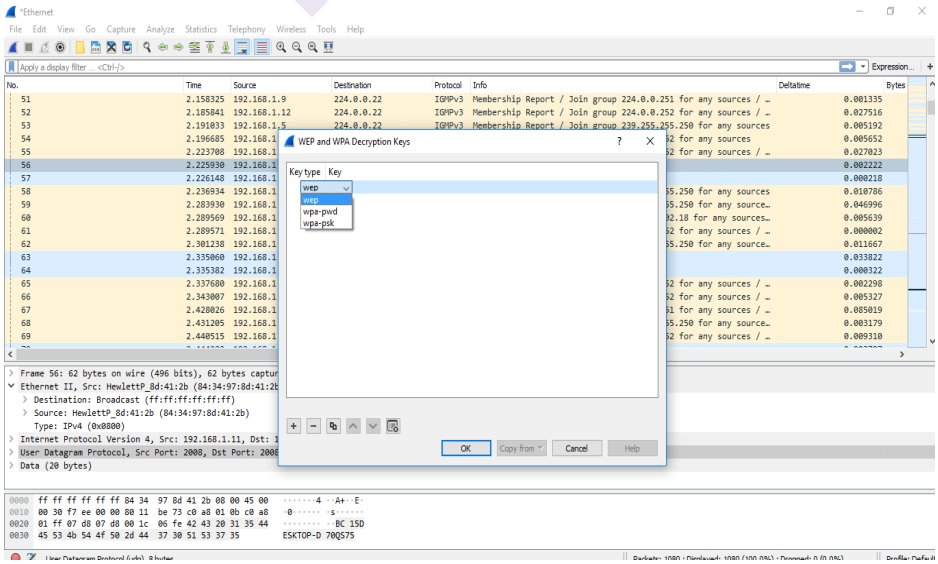
4.27 WIRESHARK DECRYPTION

The decryption process is used for the data to be in a readable format. Below are the steps for the decryption process.

- Open the Wireshark and then select the particular interface as explained above.
- Go to the 'Edit' option and select the 'Preferences' option.
- A dialogue will appear as shown below:



- Select the 'Protocol' option in the left column.
- From the drop-down list, select the 'IEEE 802.11' option. Check the box of decryption and click on the Edit option under it.
- A box will appear. Click on the option shown below:



- Select the option **wpa-pwd** and set the password accordingly.
- The data will be decrypted.
- But the above decryption process is only possible if there is a proper handshake.

REFERENCES

1. https://www.wireshark.org/
2. https://www.tek-tools.com/network/how-to-use-wireshark
3. https://www.dnsstuff.com/how-to-use-wireshark
4. https://www.computerweekly.com/news/1280099499/Using-Wireshark-to-monitor-and-secure-your-network
5. https://cybersecurity.att.com/blogs/security-essentials/network-traffic-analysis-using-wireshark
6. https://www.varonis.com/blog/how-to-use-wireshark#:~:text=Wireshark%20is%20a%20packet%20sniffer,Frame%20Relay%20connections%2C%20and%20more
7. Orzach Y. Network Analysis Using Wireshark Cookbook. Packt Publishing Ltd; 2013 Dec 24.
8. Chappell L, Combs G. Wireshark network analysis: the official Wireshark certified network analyst study guide. Protocol Analysis Institute, Chappell University; 2010 Mar.
9. Munz G, Carle G. Distributed network analysis using TOPAS and wireshark. InNOMS Workshops 2008-IEEE Network Operations and Management Symposium Workshops 2008 Apr 7 (pp. 161-164). IEEE.
10. Ndatinya V, Xiao Z, Manepalli VR, Meng K, Xiao Y. Network forensics analysis using Wireshark. International Journal of Security and Networks. 2015;10(2):91-106.
11. Orebaugh A, Ramirez G, Beale J. Wireshark & Ethereal network protocol analyzer toolkit. Elsevier; 2006 Dec 18.
12. Sanders C. Practical Packet Analysis, 3E: Using Wireshark to Solve Real-World Network Problems. No Starch Press; 2017 Mar 30.
13. Bagyalakshmi G, Rajkumar G, Arunkumar N, Easwaran M, Narasimhan K, Elamaran V, Solarte M, Hernández I, Ramirez-Gonzalez G. Network vulnerability analysis on brain signal/image databases using Nmap and Wireshark tools. IEEE Access. 2018 Oct 1;6:57144-51.
14. https://www.javatpoint.com/wireshark

VIDEO LINKS

1. How to Monitor Network packet details using Wireshark. https://www.youtube.com/watch?v=Tq3CsHmDeNw.
2. Wireshark | 02 | Capture and analyse TCP and IP packets. https://www.youtube.com/watch?v=AWERzjGEyRU
3. Wireshark | 03 | Simulate and analyse ICMP packets. https://www.youtube.com/watch?v=rEtp2Cgkpno.

4. Monitor mode wireshark.
https://www.youtube.com/watch?v=C9E1QPxktis.
5. How to Capture Packets with Wireshark.
https://www.youtube.com/watch?v=k6rx1krSUAo.
6. Wireshark Packet Sniffing Usernames, Passwords, and Web Pages.
https://www.youtube.com/watch?v=r0l_54thSYU.
7. Learn Wireshark in 10 minutes - Wireshark Tutorial for Beginners.
https://www.youtube.com/watch?v=lb1Dw0elw0Q.
8. How to Install WireShark on Ubuntu 20.04.
https://www.youtube.com/watch?v=wVLcxqXwQPw.
9. Wireshark Interface Configuration.
https://www.youtube.com/watch?v=1wB3ku4TSLY.
10. Wireshark Tutorial - How to Capture Network Traffic.
https://www.youtube.com/watch?v=qAuu9gquivU.
11. Using Wireshark to Monitor Network Traffic.
https://www.youtube.com/watch?v=Q0e_abvnMOo.
12. Network Traffic Analysis with Wireshark.
https://www.youtube.com/watch?v=nS8KweAV1g0.

QUIZ

1. _____ framework made cracking of vulnerabilities easy like point and click.

   a) .Net
   b) Metasploit
   c) Zeus
   d) Ettercap

   Answer: b

2. Nmap is abbreviated as Network Mapper.

   a) True
   b) False

   Answer: a

3. _____ is a popular tool used for discovering networks as well as in security auditing.

   a) Ettercap
   b) Metasploit
   c) Nmap
   d) Burp Suit

   Answer: c

4. Which of this Nmap do not check?

   a) services different hosts are offering

   b) on what OS they are running

   c) what kind of firewall is in use

   d) what type of antivirus is in use

   Answer: d

5. Which of the following deals with network intrusion detection and real-time traffic analysis?

   a) John the Ripper

   b) L0phtCrack

   c) Snort

   d) Nessus

   Answer: c

6. Wireshark is a _____ tool.

   a) network protocol analysis

   b) network connection security

   c) connection analysis

   d) defending malicious packet-filtering

   Answer: a

7. Which of the below-mentioned tool is used for Wi-Fi hacking?

   a) Wireshark

   b) Nessus

   c) Aircrack-ng

   d) Snort

   Answer: c

8. Aircrack-ng is used for _____

   a) Firewall bypassing

   b) Wi-Fi attacks

   c) Packet filtering

   d) System password cracking

   Answer: b

9. _____ is a popular IP address and port scanner.

a) Cain and Abel

b) Snort

c) Angry IP Scanner

d) Ettercap

Answer: c

10. _____ is a popular tool used for network analysis in multiprotocol diverse network.

a) Snort

b) SuperScan

c) Burp Suit

d) EtterPeak

Answer: d

11. _____ scans TCP ports and resolves different hostnames.

a) SuperScan

b) Snort

c) Ettercap

d) QualysGuard

Answer: a

12. _____ is a web application assessment security tool.

a) LC4

b) WebInspect

c) Ettercap

d) QualysGuard

Answer: b

13. Which of the following attack-based checks WebInspect cannot do?

a) cross-site scripting

b) directory traversal

c) parameter injection

d) injecting shell code

Answer: d

14. _____ is a password recovery and auditing tool.

a) LC3

b) LC4

c) Network Stumbler

d) Maltego

Answer: b

15. L0phtCrack is formerly known as LC3.

a) True
b) False

Answer: b

SELF LEARNING

1. Wireshark relies on the WinPcap driver when running on a Windows host.
   A. True.
   B. False
2. The TCP handshake consists of SYN, SYN/ACK, and ACK packets.
   A. True
   B. False
3. The Wireshark IO Graph can be used to view the packets-per-second rate of traffic.
   A. True
   B. False
4. The filter ip.addr == 10.10.10.10 can be used as a capture filter.
   A. True
   B. False
5. The promiscuous mode must be enabled when using Wireshark to capture traffic between other hosts on a network.
   A. True
   B. False
6. Wireshark Capture Filters can be applied to saved trace files.
   A. True
   B. False
7. UDP is a connection-oriented transport protocol.
   A. True
   B. False
8. Wireshark was founded in 1990.
   A. True
   B. False
9. Originally it was named Ethereal.
   A. True
   B. False
10. Wireshark is written in C, C++.
    A. True
    B. False

❄❄❄❄❄❄❄

5

REAL TIME PROBLEM SOLVING

Unit Structure

5.0 NS3 Installation Process
5.1 EVALVID Installation Steps
5.2 Network Simulation
5.3 Procedure To Build The Simple Script In Ns3 Projects
5.4 Multi Path Tool

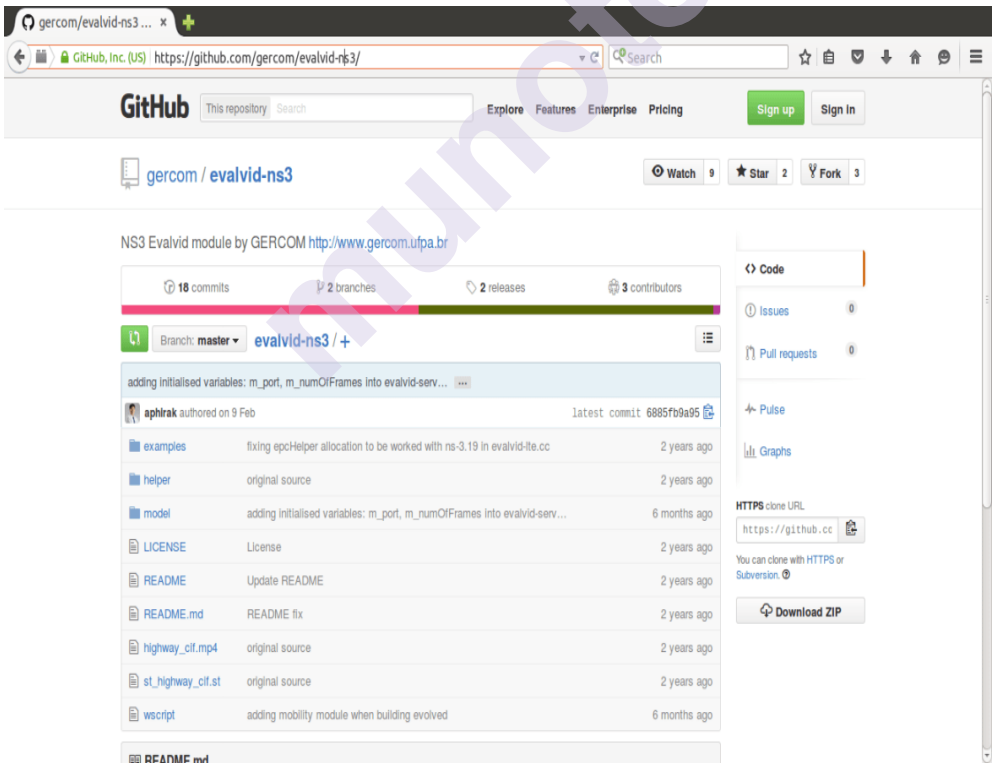
5.0 NS3 INSTALLATION PROCESS:

https://www.youtube.com/watch?v=NQUFgm27FuE

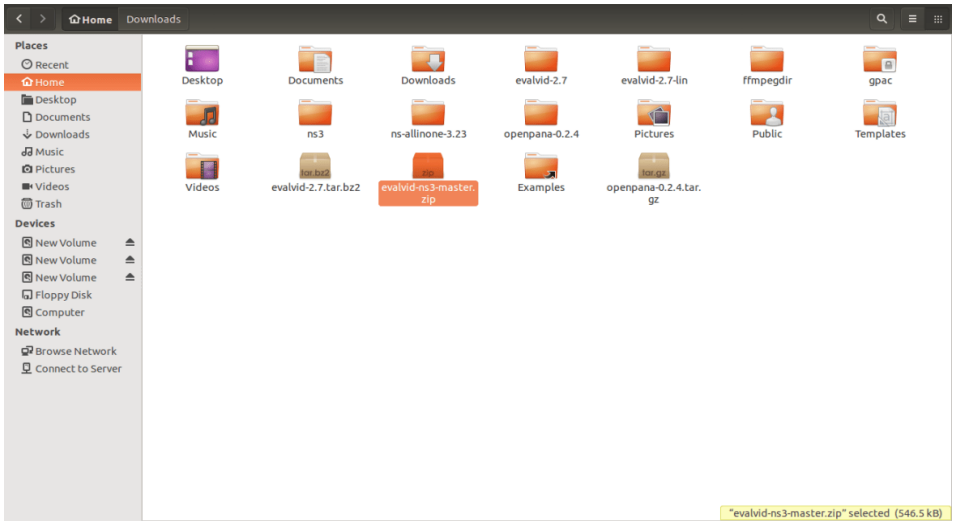
5.1 EVALVID INSTALLATION STEPS

Guidelines To Install EVALID in NS3

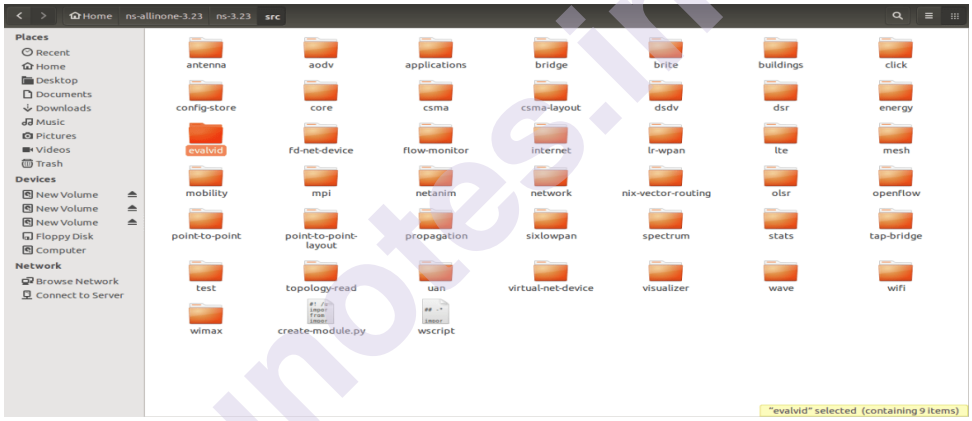
STEP 1: FIRST OF ALL INSTALL THE NS3 INTO YOUR SYSTEM



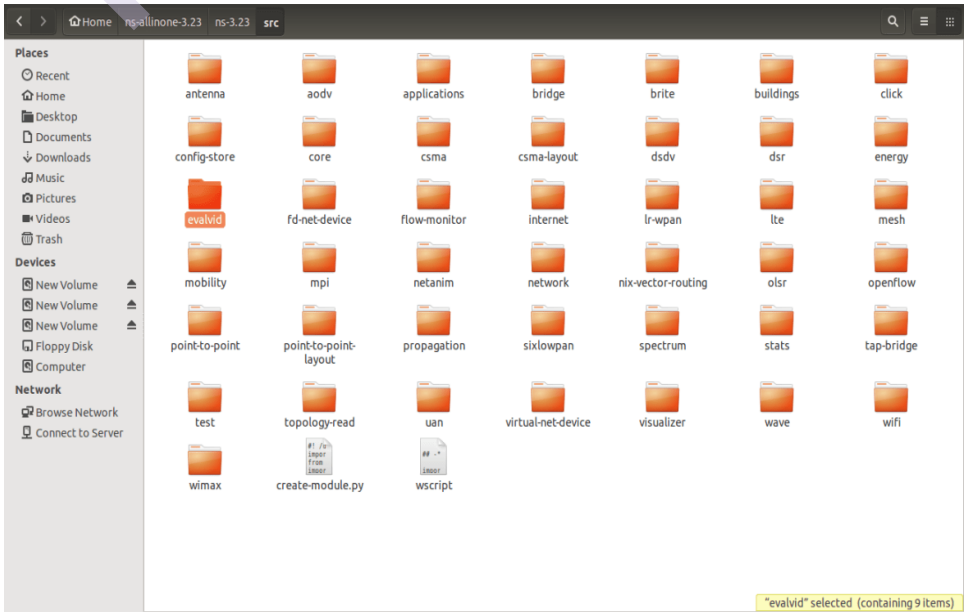
STEP 2:NEXT DOWNLOAD EVALVID MODEL



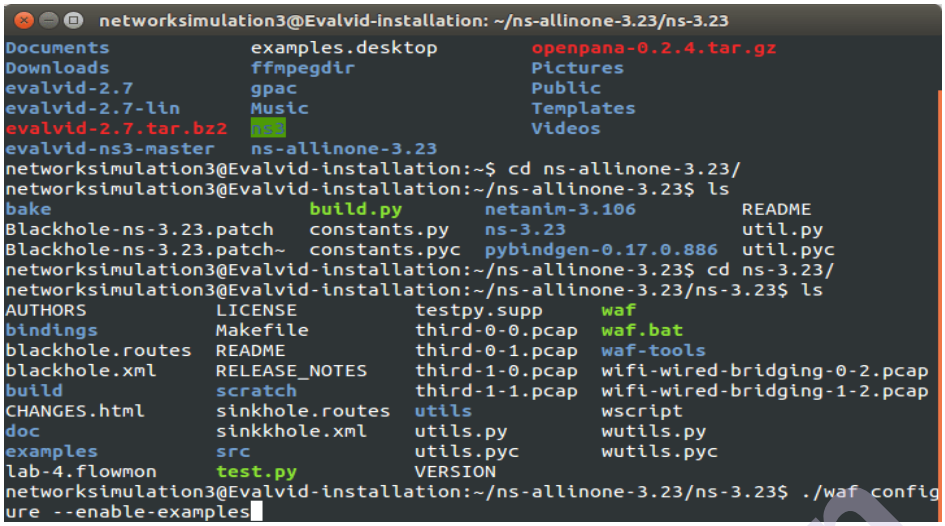
STEP 3: EXTRACT THE EVALVID-NS3-MASTER FILE

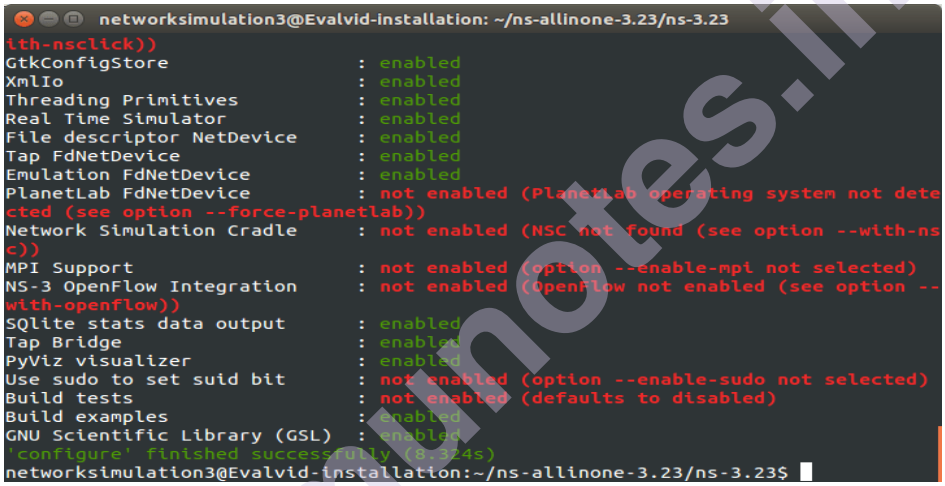


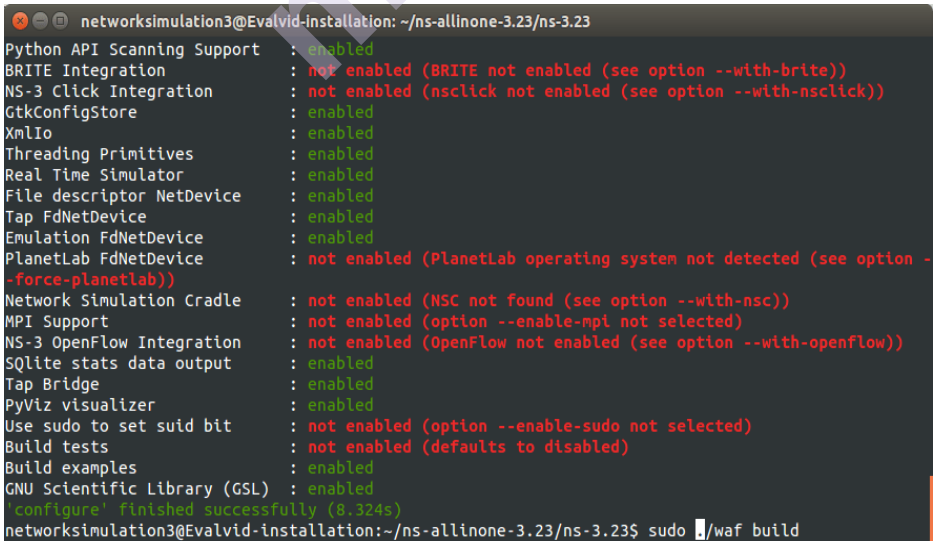
STEP 4: NEXT GO TO NS3 INSTALLATION LOCATION CREATE DIRECTORY EVALID TO PASTE ./NS3/SRC /EVALVID

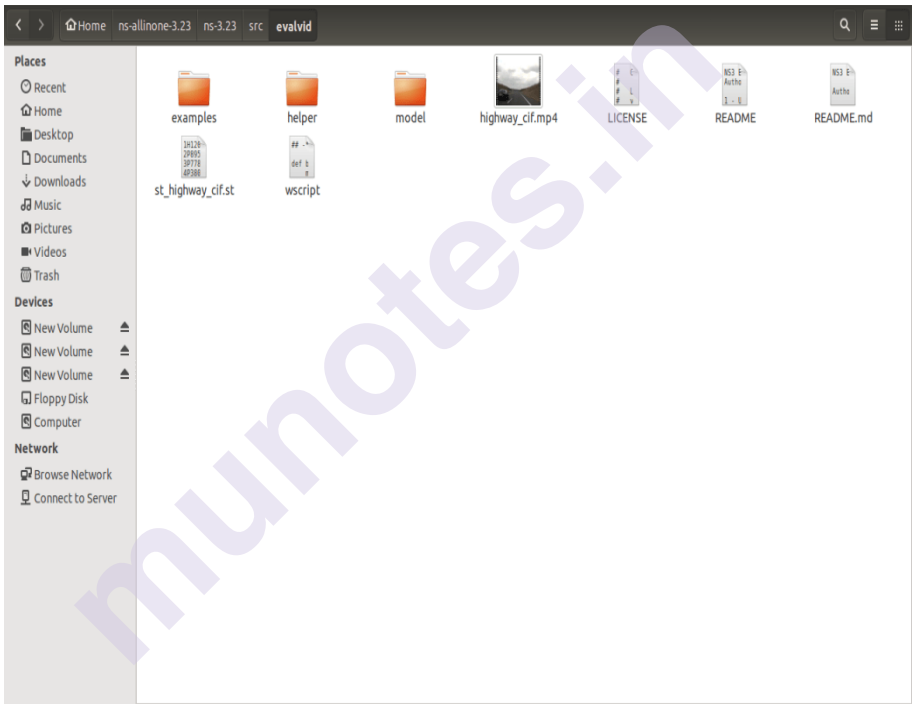


Step 5:Build the evalid model help the commands are :
sudo ./waf configure –enable-examples
sudo ./waf build









5.2 NETWORK SIMULATION

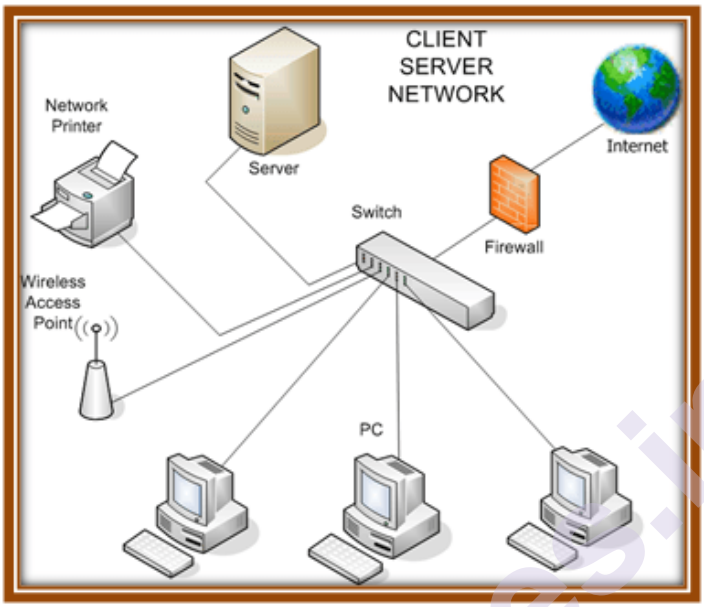
Computer Network Simulation in the sense connection of two or more computer system linked together for communication. Networking is the practice of interfacing two or more computing devices with each other for the purpose of sharing data. Computer networks are built with a combination of hardware and software.

Types of computer network design:

- Peer-to-peer network model.
- Client/server model.

Common computer network topologies:

- Star.
- Mesh.
- Bus.
- Ring.



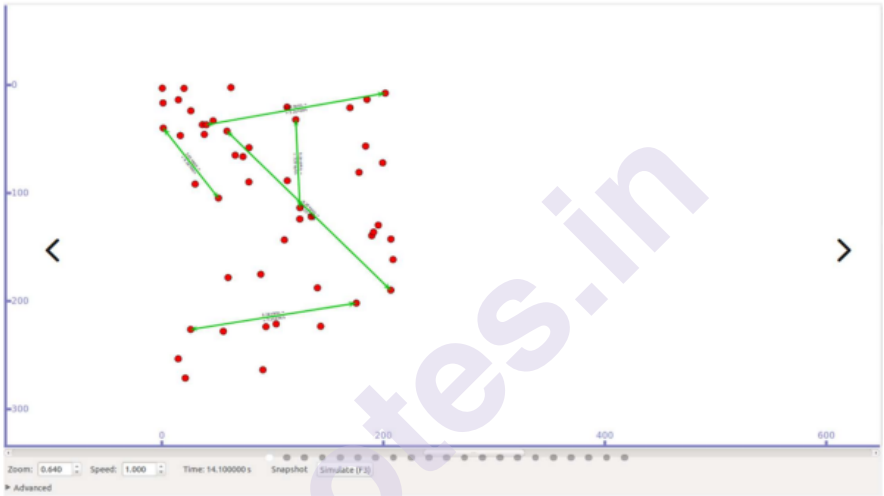
Sample code for computer network simulation:

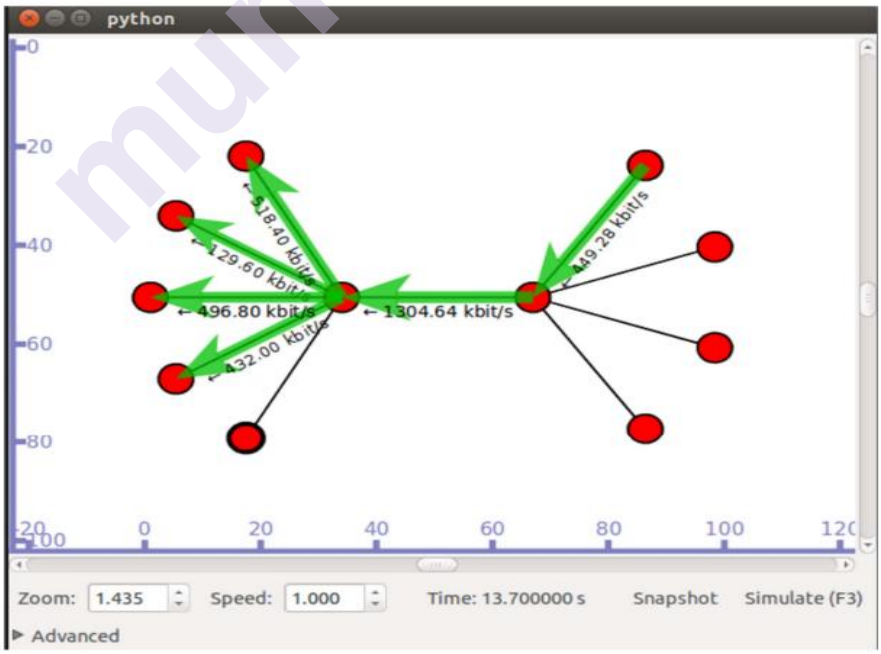
```
NS_LOG_INFO (“Create nodes.”);
NodeContainer nodes;
nodes.Create (2);
NS_LOG_INFO (“Create channels.”);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute (“DataRate”, StringValue (“500Kbps”));
pointToPoint.SetChannelAttribute (“Delay”, StringValue (“5ms”));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
InternetStackHelper internet;
internet.Install (nodes);
NS_LOG_INFO (“Assign IP Addresses.”);
Ipv4AddressHelper ipv4;
ipv4.SetBase (“10.1.1.0”, “255.255.255.0”);
Ipv4InterfaceContainer i = ipv4.Assign (devices);
NS_LOG_INFO (“Create Applications.”);
uint16_t port = 9; // well-known echo port number
BulkSendHelper source (“ns3::TcpSocketFactory”,
InetSocketAddress (i.GetAddress (1), port));
// Set the amount of data to send in bytes. Zero is unlimited.
source.SetAttribute (“MaxBytes”, UintegerValue (maxBytes));
ApplicationContainer sourceApps = source.Install (nodes.Get (0));
```

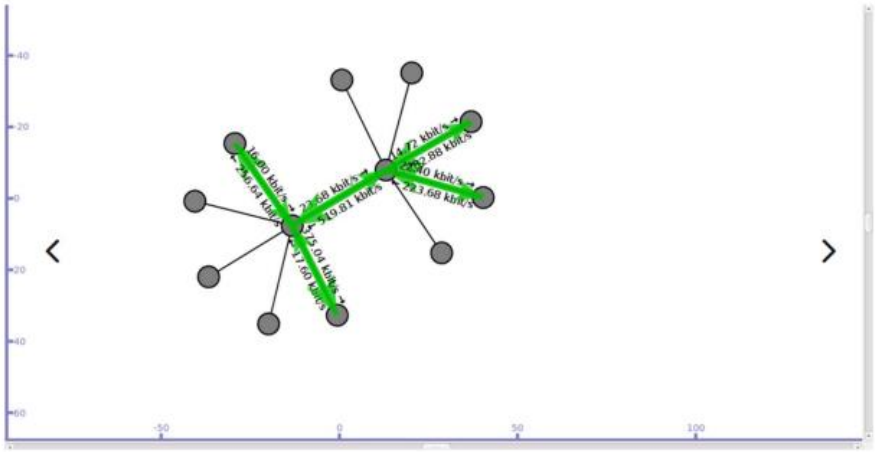
```
sourceApps.Start (Seconds (0.0));
sourceApps.Stop (Seconds (10.0));
```

5.3 PROCEDURE TO BUILD THE SIMPLE SCRIPT IN NS3 PROJECTS:

- Create the network topology which means node container Install the internet stack
- Establish the physical connection this means connect the nodes
- together Assign IP address for each nodes present in network
- Now start the application. For example like client and server

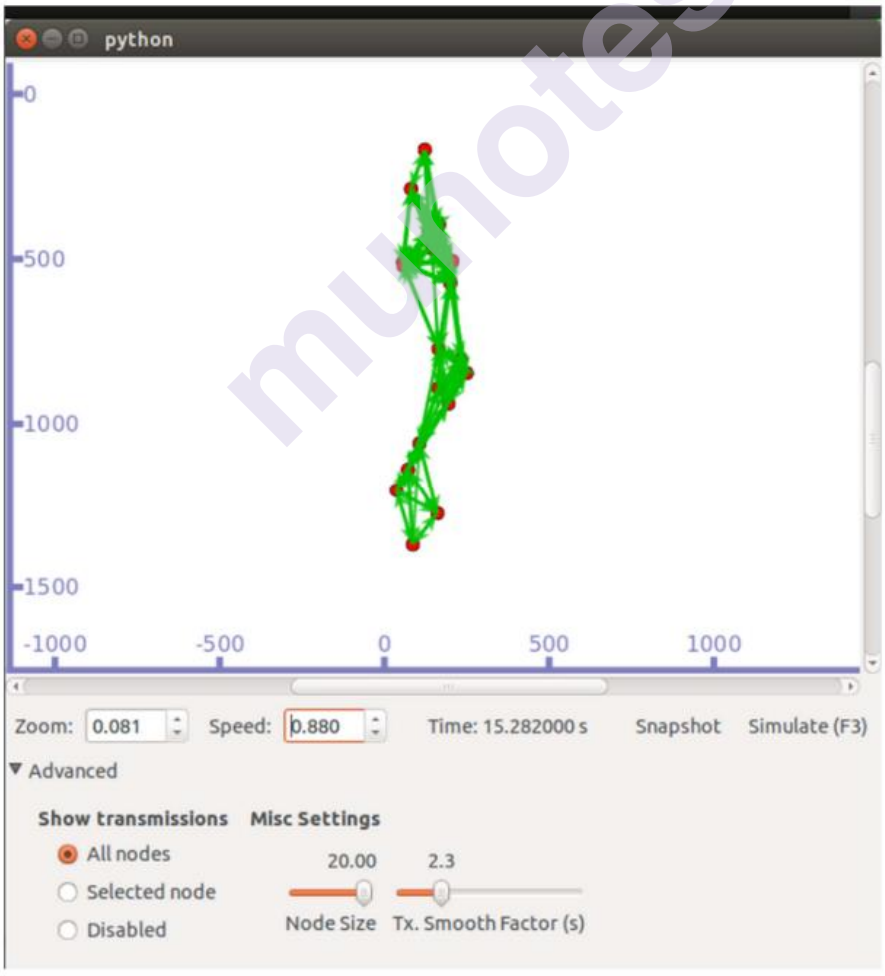


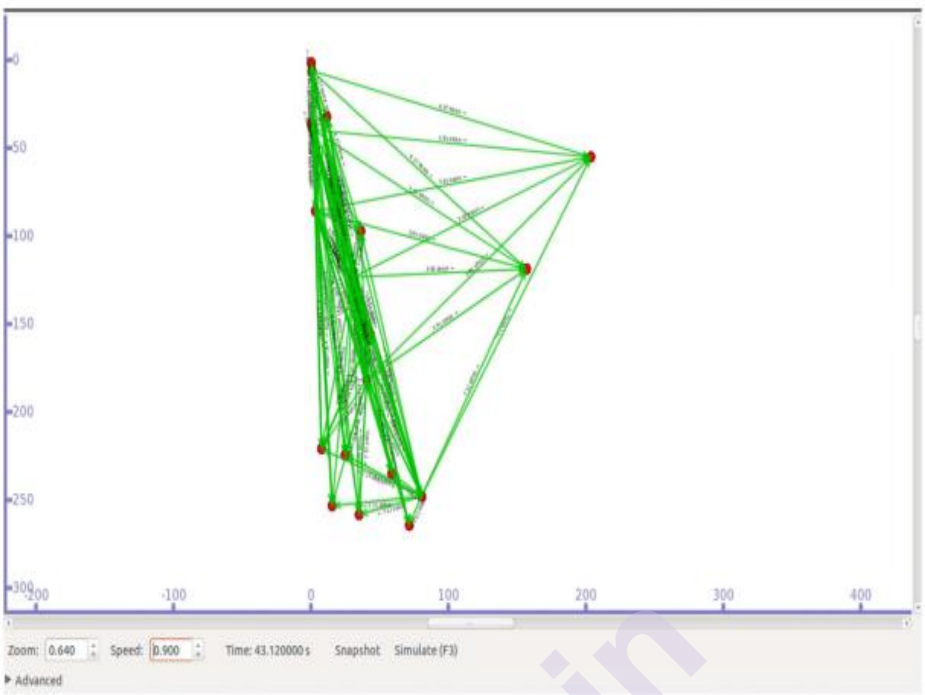


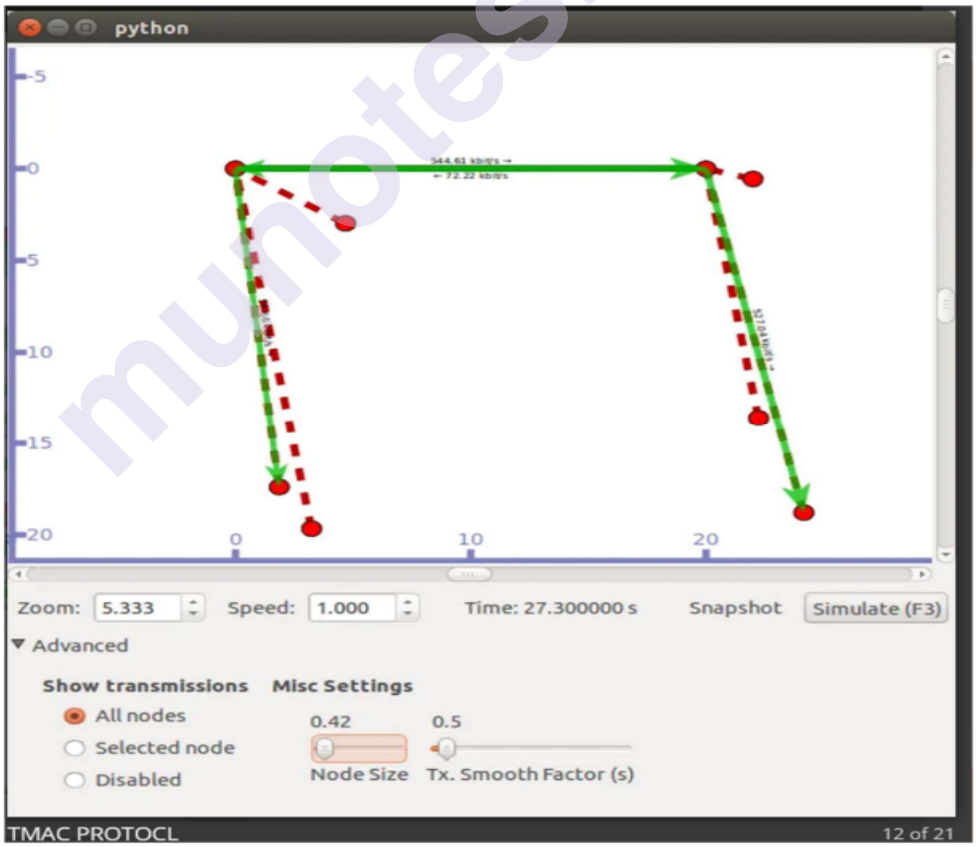


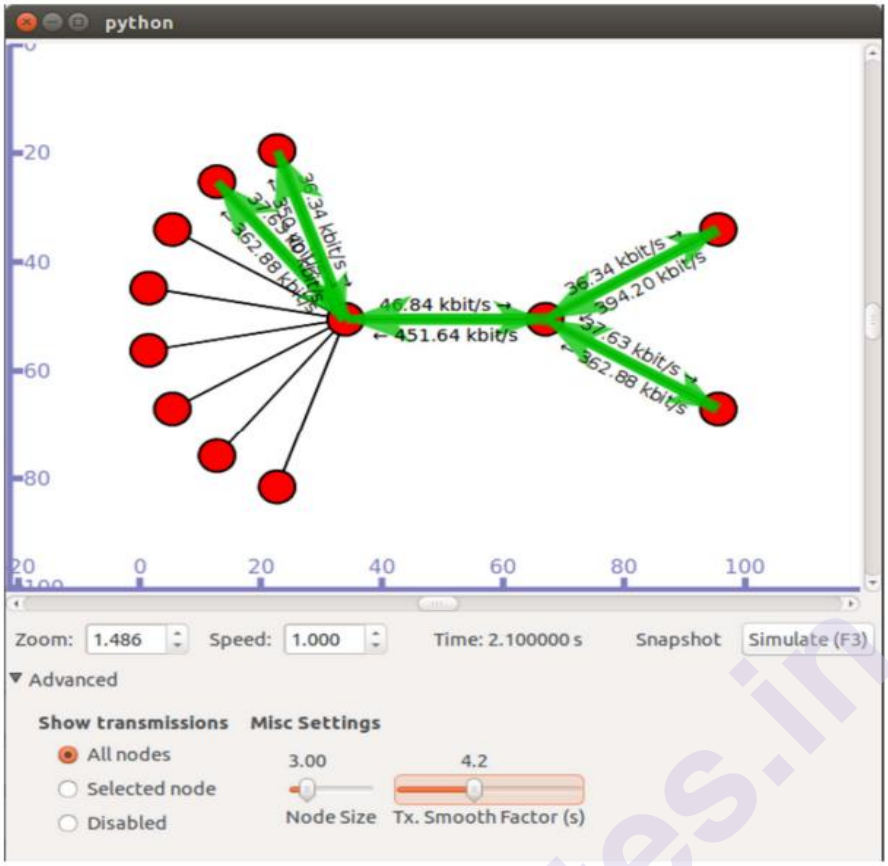
5.4 MULTI PATH TOOL

Modeling of recurring event in ns3 project:
r: a packet was received by the net device
+: an enqueue operation occurred on the device queue
-: a dequeue operation occurred on the device dequeue
d: a packet was dropped, typically because the queue was full.




0
50
100
150
200
250
300
-200
-100
0
100
200
300
400
Zoom: 0.640
Speed: 0.900
Time: 43.120000 s
Snapshot
Simulate (F3)
Advanced


python
5
0
5
10
15
20
0
10
20
Zoom: 5.333
Speed: 1.000
Time: 27.300000 s
Snapshot
Simulate (F3)
Advanced
Show transmissions
All nodes
Selected node
Disabled
Misc Settings
0.42
Node Size
0.5
Tx. Smooth Factor (s)
TMAC PROTOCL
12 of 21


python
20
40
60
80
20 0 20 40 60 80 100
36.34 kbit/s
37.63 kbit/s
362.88 kbit/s
46.84 kbit/s
451.64 kbit/s
36.34 kbit/s
394.20 kbit/s
37.63 kbit/s
362.88 kbit/s
Zoom: 1.486
Speed: 1.000
Time: 2.100000 s
Snapshot
Simulate (F3)
Advanced
Show transmissions
All nodes
Selected node
Disabled
Misc Settings
3.00
Node Size
4.2
Tx. Smooth Factor (s)

❄❄❄❄❄❄

6

NETANIM

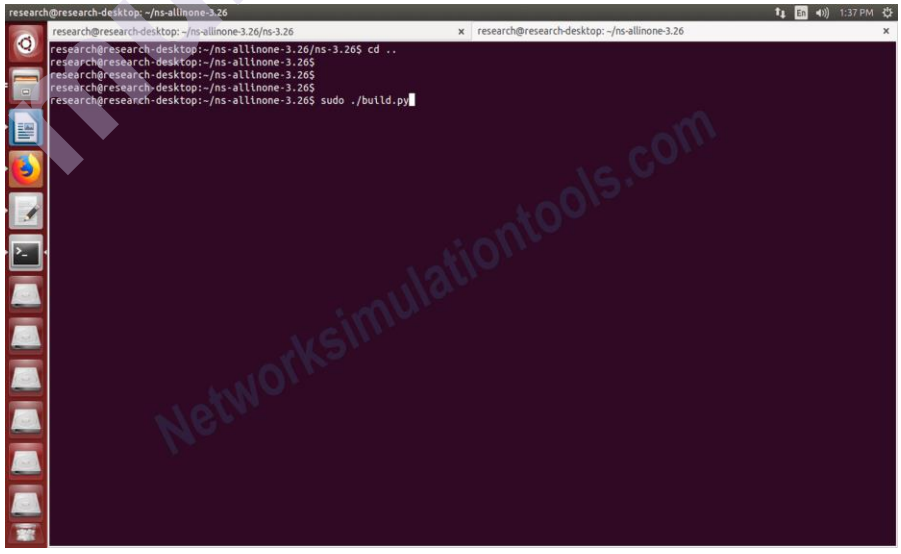
Unit Structure

6.0 HOW TO INSTALL NETANIM IN NS3

1. Follow from Step 1 to Step 10 in order to create NS3 using Simulation projects. Quick guide to create NS3 simulation. Reach us, if you want an customize NetAnim in NS3 simulation projects works for scholars.

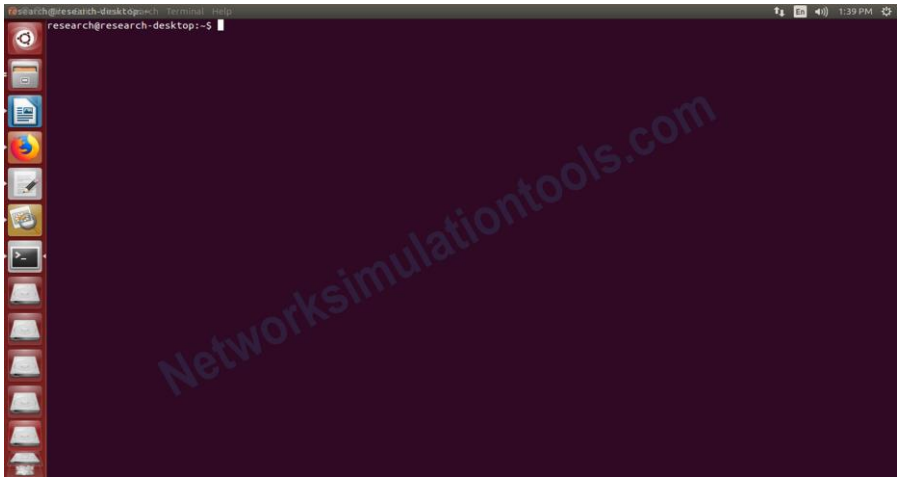
   Install the NS-3.26

   Initially , install the NS-3.26 tool by using the ns-allinone-3.26.tar.bz2 package. For make install use and execute the command ./build.py



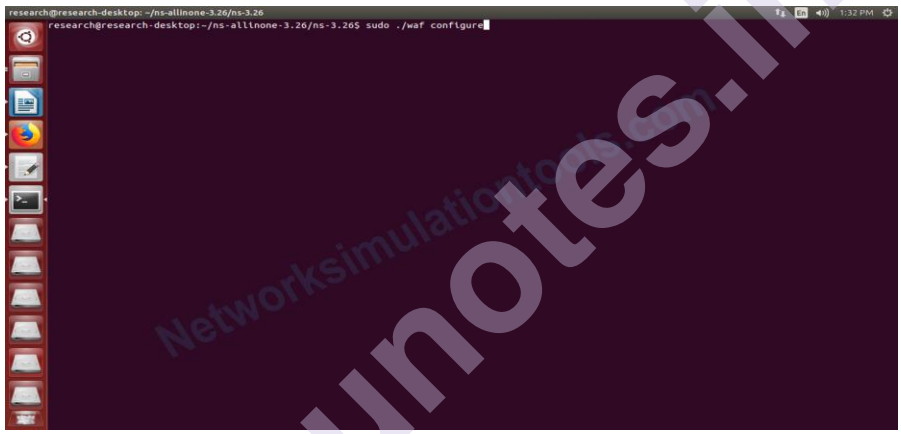
2. **Open the terminal**

   Initially,Next open the terminal by press ctrl+alt+T buttons or search from the installed software list.



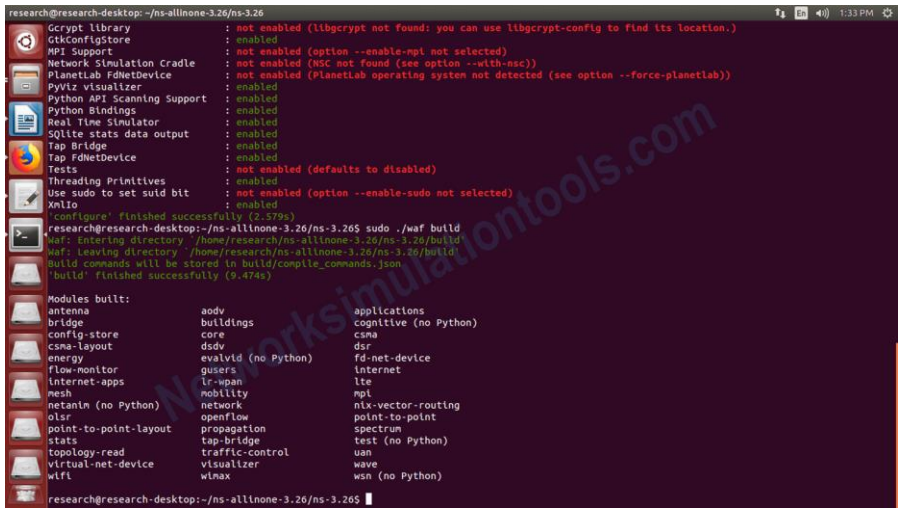
3. Configure the package

Change the ns-allinone installation location in the terminal, using the command cd ns-allinone-3.26/ns-3.26. And execute the command sudo ./waf configure to binding the python



4. Build the package

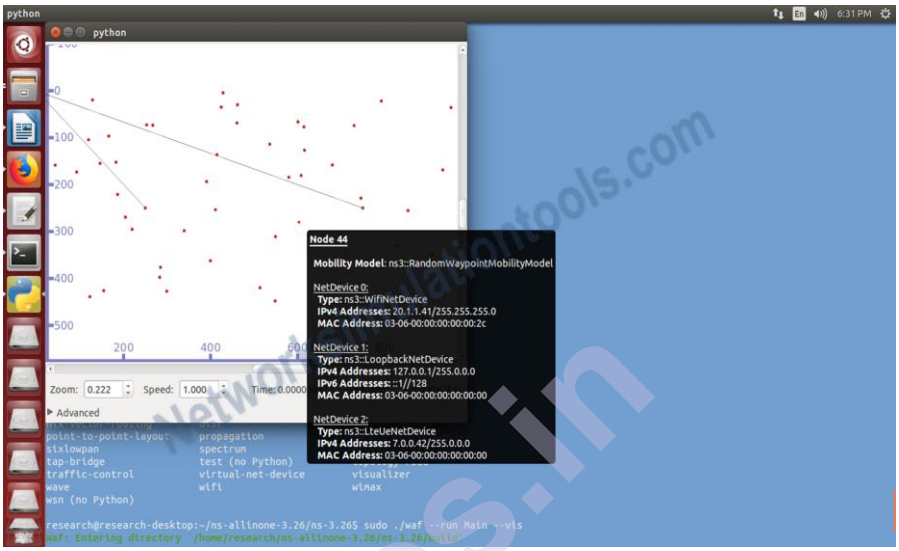
For build the configured package by using the sudo ./waf build command in the ns-allinone-3.26/ns-3.26 . Package and view the build packages.



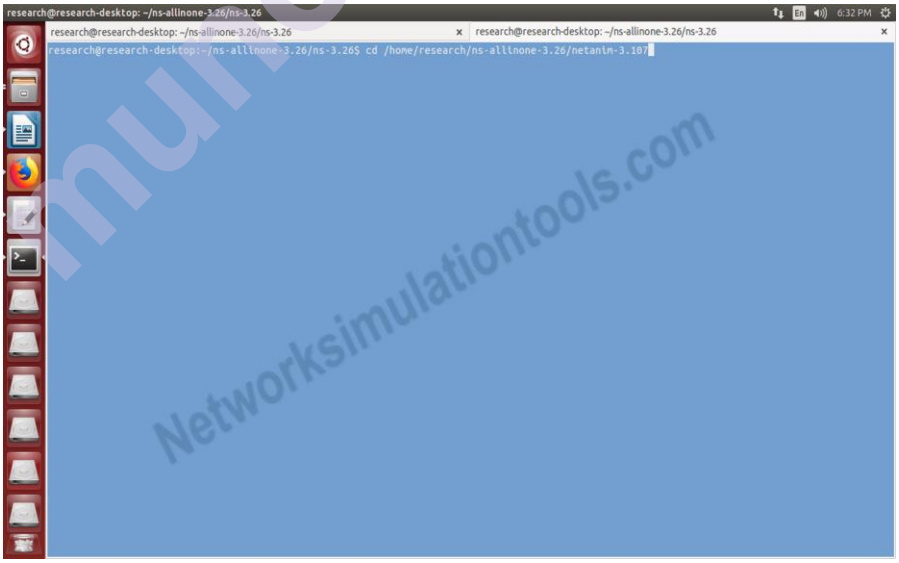
5. Create a main file

Next create new main file in the scratch folder. The main file stored with the file extension .cc. For example , the input file is stored as Main.cc

next, execute the main file by using the command sudo ./waf –run Main –vis

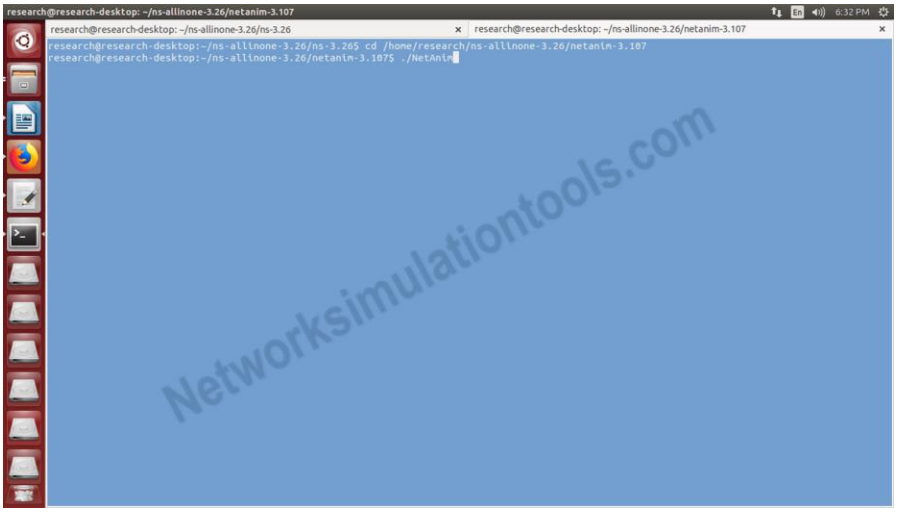


6. NetAnimator result change the location for get the netanimator result by using the command cd /home/research/ns-allinone-3.26/netanim-3.107



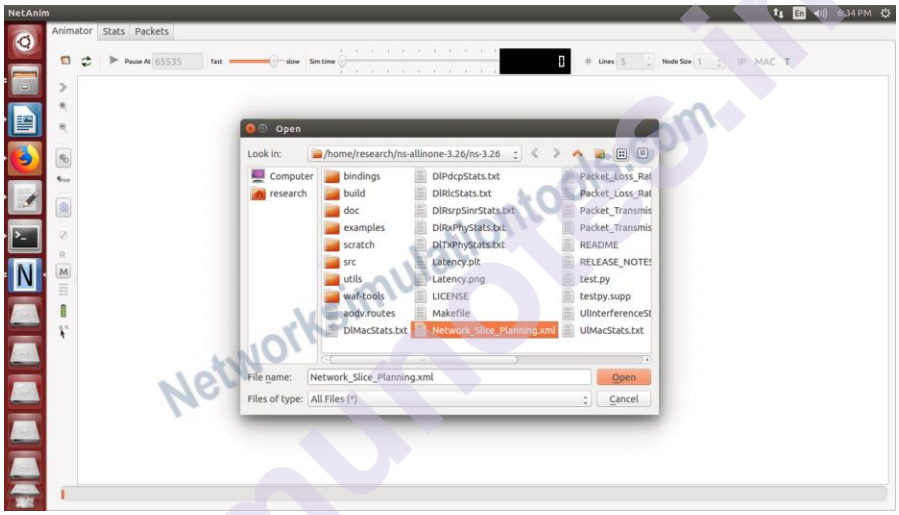
7. Execute the command

Execute the command ./NetAnim,if the file NetAnim is not in executable format, then convert into executable , by using the sudo chmod +x NetAnim



8. Execution the xml file

Select the xml file , during the simulation execution the xml file is auto generated.

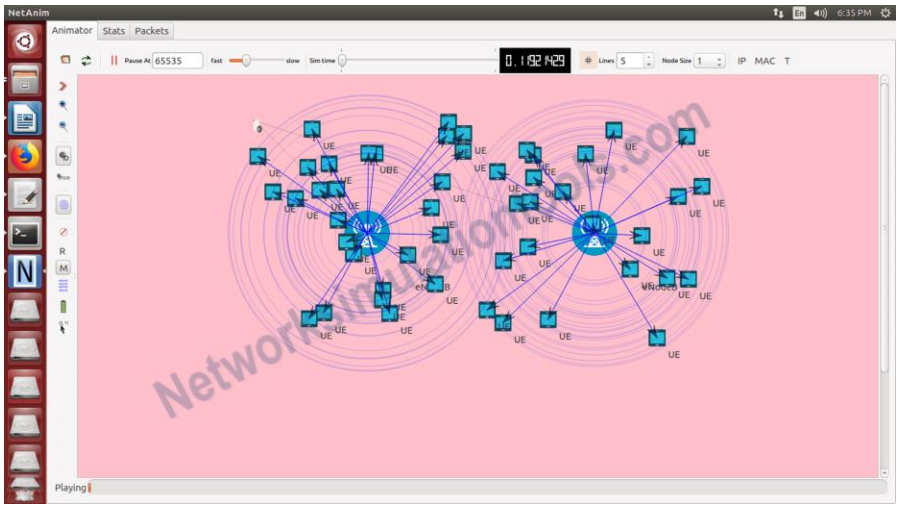


9. Simulation Process

During implementation of the xml file is auto generated. Play the simulation by press the play button

10. NetAnim Result

By using the NetAnim, Play the simulation by press the play button



6.1 RUN NETANIM IN NS3

Follow from Step 1 to Step 9 in order to create NS3 using Simulation projects.

System requirements

Step 1: For install the NS-3.26 , we need the following minimum system requirements.

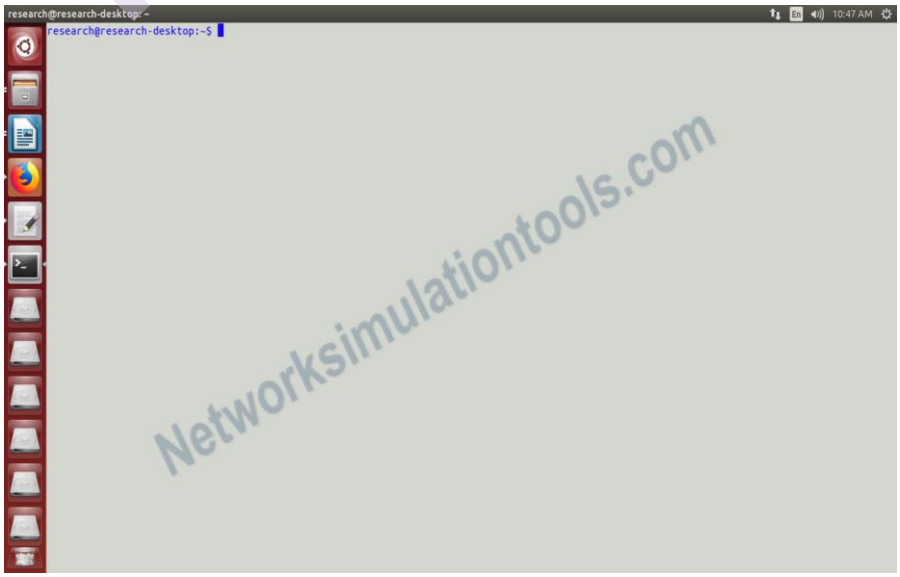
1) OS : ubuntu-14.04 LTS(32 bit)

2) RAM :minimum 2GB

3) Processor: 2.5 GHz and above

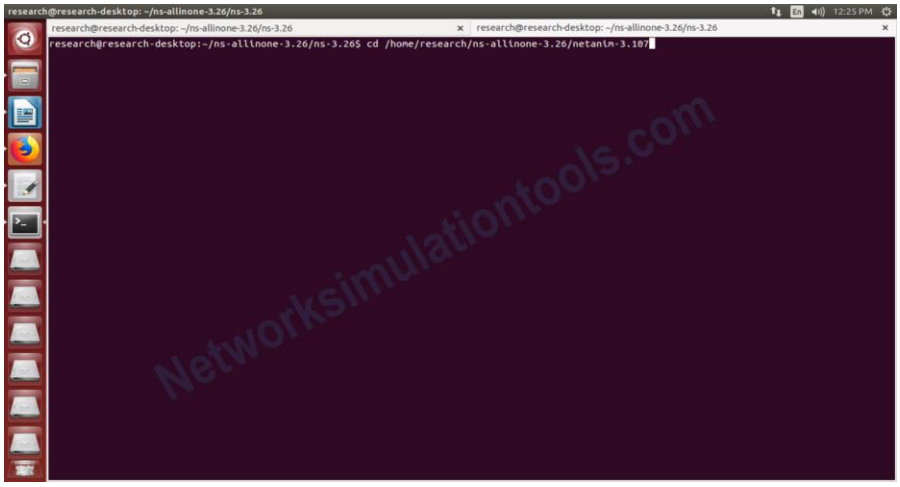
Step 2: Open the terminal and verify the installed package

Initially,Next open the terminal by press ctrl+alt+T buttons or search from the installed software list.



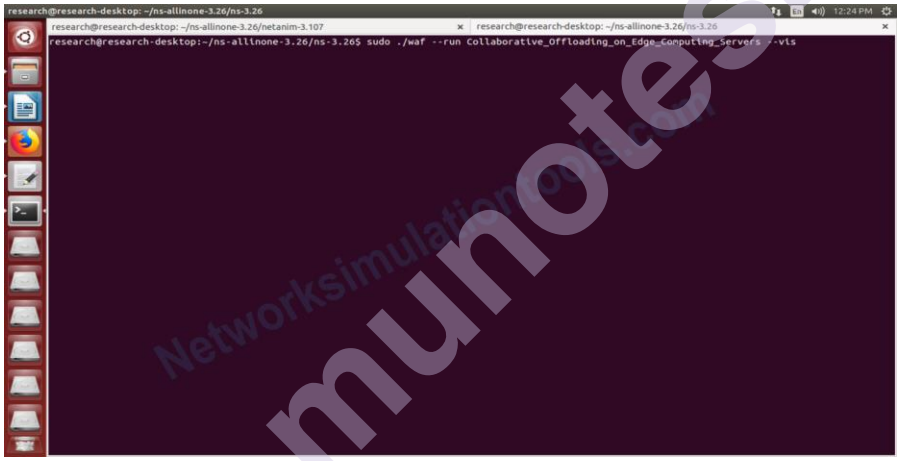
Step 3: Change the location

Change the ns-allinone installation location in the terminal, using the command cd ns-allinone-3.26/ns-3.26



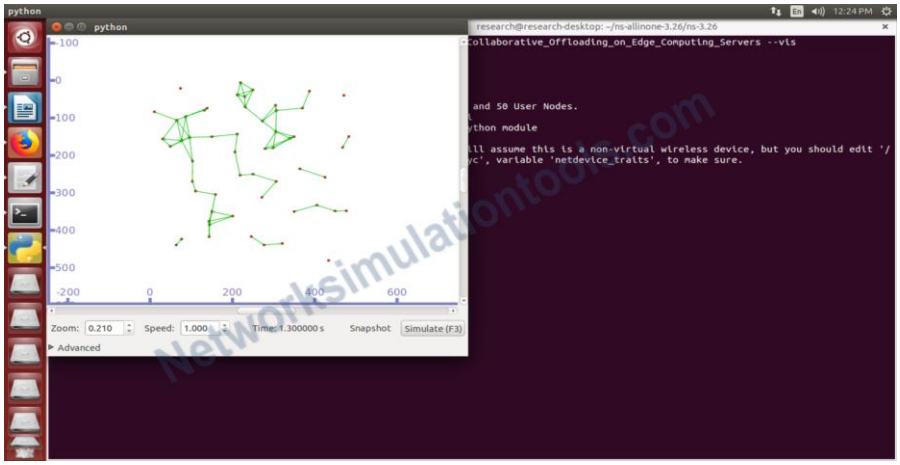
Step 4: Execute the main file

For get the simulation , execute the main file by using the command sudo ./waf –run Mainfile –vis



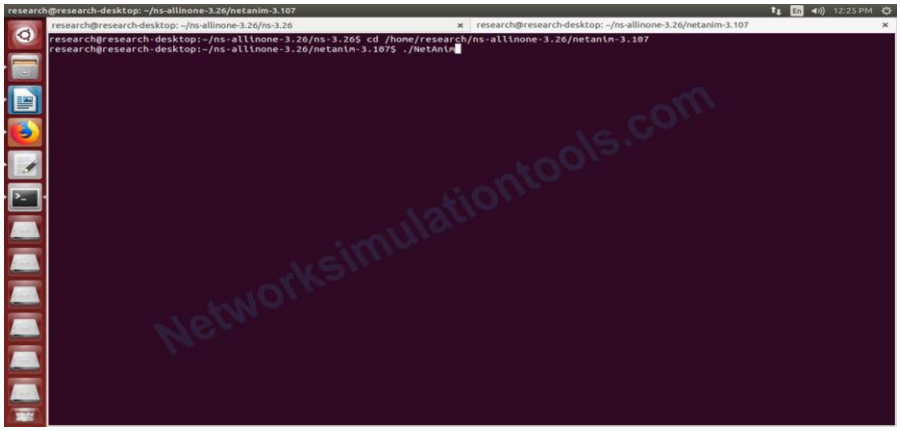
Step 5: Simulation Nodes

Get the simulation window with the specified number of nodes



Step 6: Perform the NetAnim process

Change the location for get the netanimator result , by using the command cd /home/research/ns-allinone-3.26/netanim-3.107

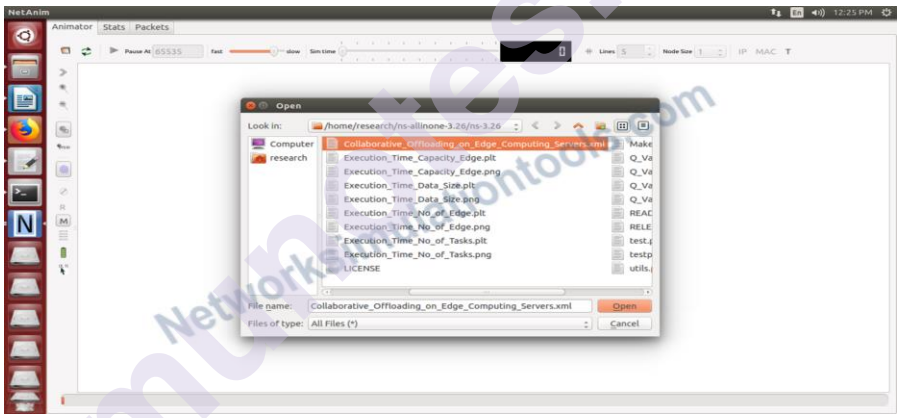


Step 7: Execute NetAnim

by using the following command

cd /home/research/ns-allinone-3.26/netanim-3.107

execute the command ./NetAnim



Step 8: Simulation execution

Select the XML file, the XML file is one of the auto generated file. During the simulation execution the file is generated.

Step 9: Simulation window

Get the simulation window and play the simulation. Play the simulation by press the paly button in the simulation window

WIRESHARK

6.2 SNIFFING HTTP TRAFFIC WITH WIRESHARK

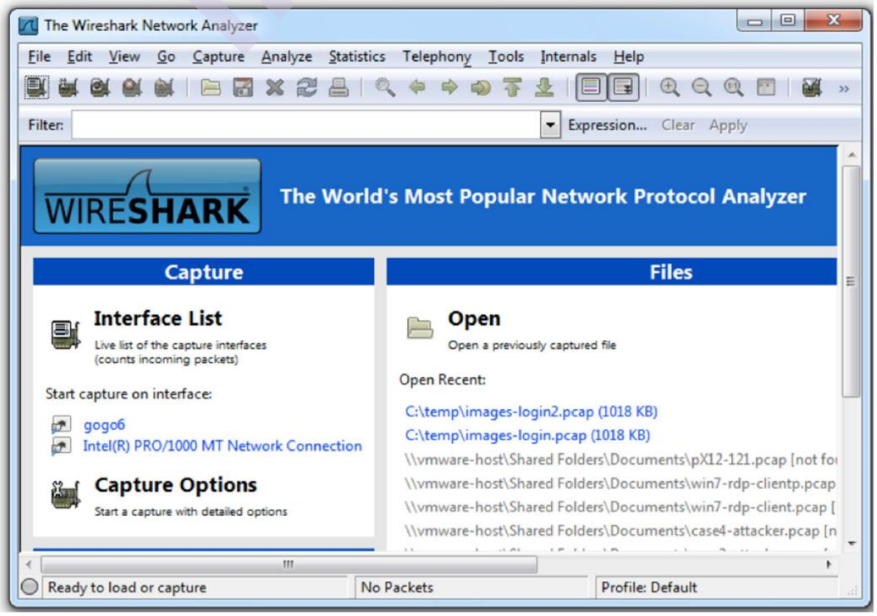
Procedure :

1) Installing the Wire Shark Tool

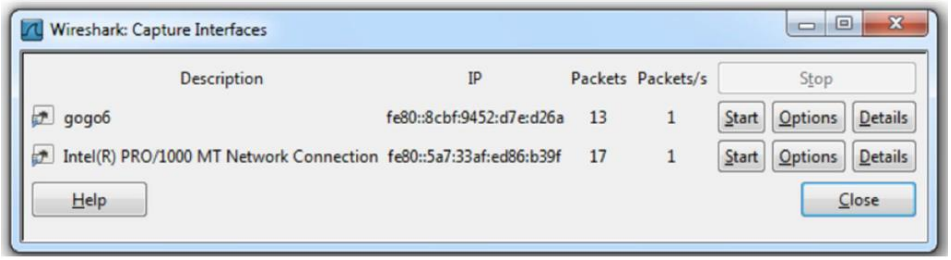
2) http://www.wireshark.org/download.html

Capturing All Network Traffic With WireShark

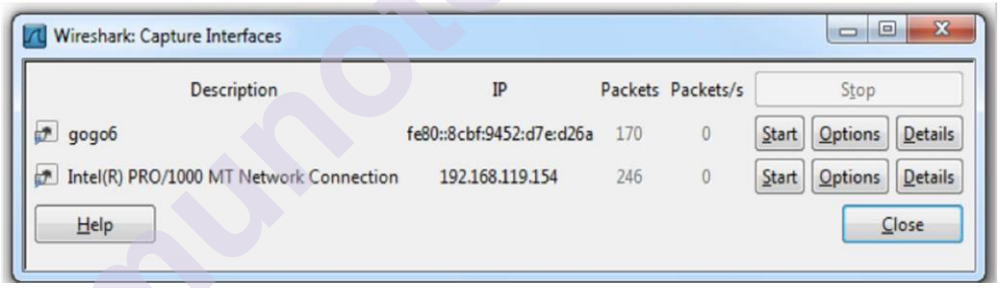
- Click the Start button. In the Search box, type WIRE
- At the top of the menu, a Wireshark item appears. Right-click Wireshark and click "Run as Administrator". If a User Account Control box appears, allow the program to run.



- In the upper left of the Wireshark window, click "Interface List".
- A list of network interfaces appears. Each interface has an IP address and a count of Packets, as shown below.



- At first, all the IP addresses start with fe80: -- these are "Link- Local IPv6 Addresses", and they very useful.
- Find the network interface with the most rapidly increasing number of packets--this is the interface that connects to the Internet. Click its IP address.
- Wireshark will show the other addresses of this interface.
- After one or more clicks, you should see the IPv4 address of the interface, which is four values separated by periods, as shown below:

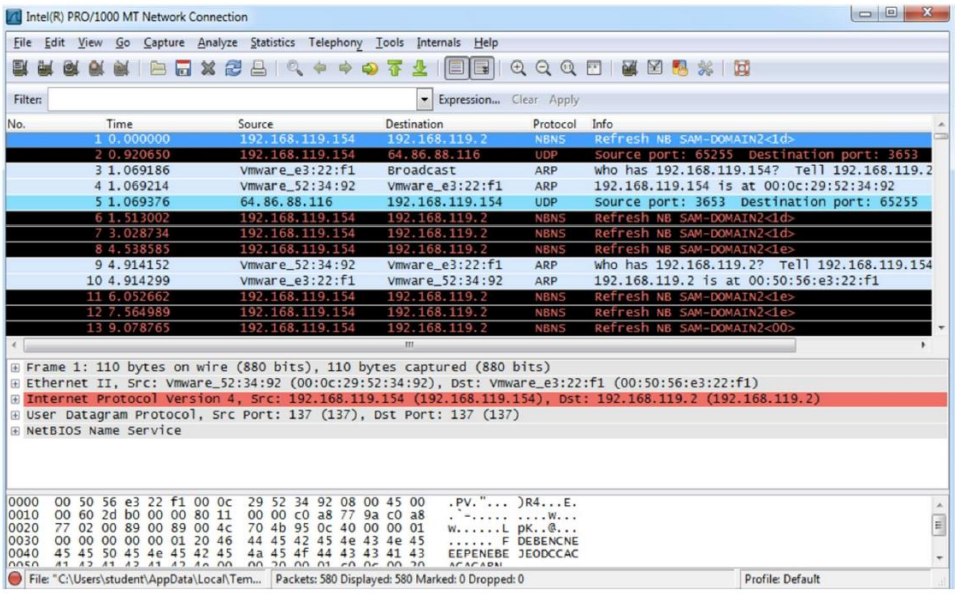


Click the Start button next to the interface that connects to the Internet.

You should see a lot of text scrolling by, as shown below on this page. Each line in the upper pane summarizes one frame (or packet).

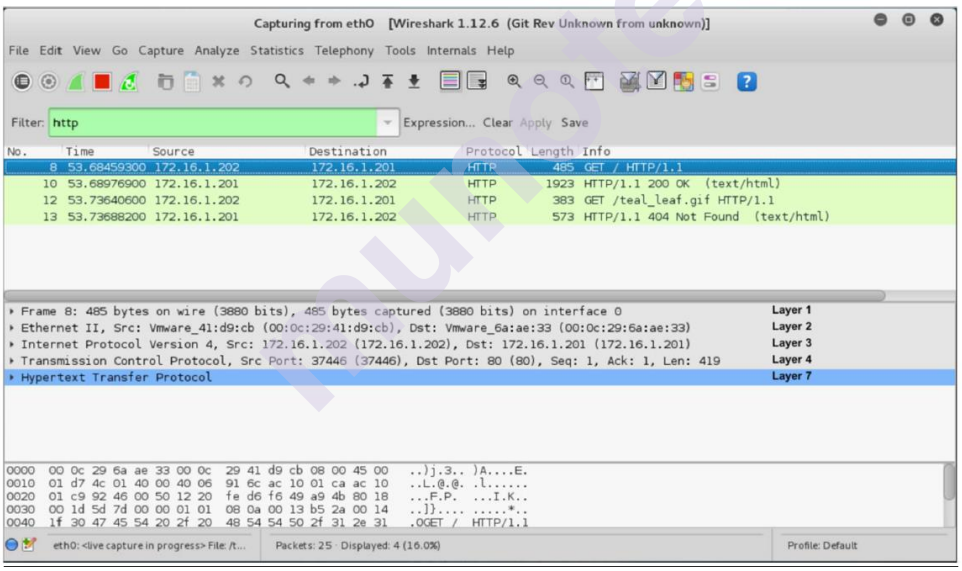
Find these columns in the Wireshark window:

- No Frame Number
- Time Time in seconds that the frame was captured
- Source Source address of the frame
- Destination Destination address of the frame
- Protocol Protocol of the frame
- Info Other information



Capturing The HTTP Traffic with wireshark

- At the upper left of the Wireshark window, in the "Filter" bar, type http
- In Wireshark, on the right side of the Filter bar, click Apply.
- Wireshark now shows only HTTP packets, as shown below.



Understanding HTTP GET Packets

Find the packet with "Info" of "GET / HTTP/1.1", as highlighted in the image above. This packet requests a Web page.

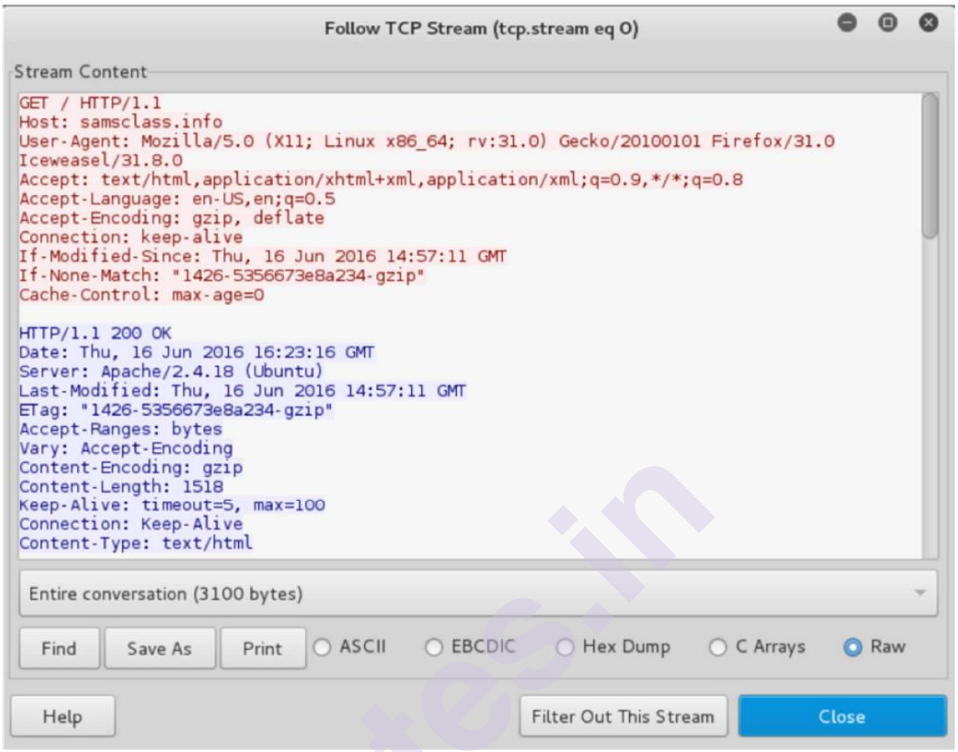
The next packet, with "Info" of "HTTP/1.1 200 OK...", is the response from the Web server.

Following the TCP Stream

In Wireshark, click the packet with "Info" of "GET / HTTP/1.1", to highlight it, as shown in the image above.

From the Wireshark menu bar, click Analyze, "Follow TCP Stream".

This is the most convenient way to examine HTTP traffic. The request is shown in red, and the response is shown in blue, as shown below.



6.3 REFERENCES

How to Install NetAnim in NS3. https://networksimulationtools.com/how-to-install-netanim-in-ns3/

How to Run Netanim in NS3. https://networksimulationtools.com/how-to-run-netanim-in-ns3/

NS3 Projects. https://networksimulationtools.com/ns3-projects/

NS3 Installation. https://ns3-code.com/ns3-projects/

NS -3 Projects. https://github.com/topics/ns-3

Network-Projects. https://ns3simulation.com/network-projects/network-projects-with-source-code-for-students/

6.4 MOOCS

Ns-3. https://www.nsnam.org/consortium/activities/training/

ADVANCE DIPLOMA IN NS3 SIMULATION TECHNOLOGY. https://exuberantsolutions.com/ns3-training.htm

Ns-3. http://www.e2matrix.com/ns3-training-course-module/

Advance Diploma In NS2 & NS3 Technologies (ADNT). https://itronixsolution.com/ns2-ns3-training-jalandhar-punjab/

Network Simulation using NS2. https://www.udemy.com/course/network-simulation-using-ns2/.

A Discrete-Event Network Simulator, https://www.nsnam.org/doxygen/group__internet.html

NS3 Simulator Code. https://ns3tutorial.com/ns3-simulator-code/

Wireshark: Functionality. https://www.linkedin.com/learning/wireshark-functionality?src=go-pa&veh=sem_src.go-pa_c.LLS-C_APAC_IN_T2_EN_DSA_SEM_GoogleAds_NA_All_NA_NA_Core_NA_Course-DSA_Nonbrand_DSA_pkw._pmt._pcrid.473973938317_pdv.c_plc._trgid.aud-1016811499848:dsa-924343687571_net.g_learning&trk=sem_src.go-pa_c.LLS-C_APAC_IN_T2_EN_DSA_SEM_GoogleAds_NA_All_NA_NA_Core_NA_Course-DSA_Nonbrand_DSA_pkw._pmt._pcrid.473973938317_pdv.c_plc._trgid.aud-1016811499848:dsa-924343687571_net.g_learning&mcid=6841855808129646771&cname=&camid=11414361728&asid=114147496960&targetid=aud-1016811499848:dsa-924343687571&crid=473973938317&placement=&dev=c&ends=1&gclid=Cj0KCQjw3v6SBhCsARIsACyrRAmEb08IQTrZhoYtkdDf-YLHzXPgCqoetofs_N9mmZUElqqmvwAUiD4aAqwfEALw_wcB&gclsrc=aw.ds

Wireshark: Malware and Forensics. https://www.linkedin.com/learning/wireshark-malware-and-forensics?trk=learning-course_similar-courses&upsellOrderOrigin=sem_src.go-pa_c.LLS-C_APAC_IN_T2_EN_DSA_SEM_GoogleAds_NA_All_NA_NA_Core_NA_Course-DSA_Nonbrand_DSA_pkw._pmt._pcrid.473973938317_pdv.c_plc._trgid.aud-1016811499848%3Adsa-924343687571_net.g_learning

Wireshark Essential Training. https://www.linkedin.com/learning/wireshark-essential-training?trk=learning-course_similar-courses&upsellOrderOrigin=sem_src.go-pa_c.LLS-C_APAC_IN_T2_EN_DSA_SEM_GoogleAds_NA_All_NA_NA_Core_NA_Course-DSA_Nonbrand_DSA_pkw._pmt._pcrid.473973938317_pdv.c_plc._trgid.aud-1016811499848%3Adsa-924343687571_net.g_learning

Wireshark 2.0: Fundamentals. https://www.pluralsight.com/courses/wireshark-2-0-fundamentals?aid=7010a000002BWq6AAG&promo=&utm_source=non_branded&utm_medium=digital_paid_search_google&utm_campaign=IN_Dynamic&utm_content=&gclid=Cj0KCQjw3v6SBhCsARIsACyrRAkh62nsQuMRXnxTBIp6QXzbWbc_g5qSqNH1-Ypjy_L3uL-_HLrwFeQaAlgLEALw_wcB

Wireshark: Packet Analysis and Ethical Hacking: Core Skills. https://www.udemy.com/course/wireshark-packet-analysis-and-ethical-hacking-core-skills/

The Complete Wireshark Course: Go from Beginner to Advanced!. https://www.udemy.com/course/wireshark/

Start Using Wireshark to Hack like a Pro.
https://www.udemy.com/course/start-using-wireshark-to-hack-like-a-pro/

Wireshark Tutorial - Get Wireshark Certification.
https://www.udemy.com/course/wireshark-tutorial/

The Complete Wireshark Course: Beginner to Network Admin!.
https://josephdelgadillo.com/wireshark-course-free/

6.5 VIDEO LINKS

External Tools for Network Simulator 3.
https://www.youtube.com/watch?v=XcRXb3NYd3I

ns-3 Network Simulator - Introduction Lecture.
https://www.youtube.com/watch?v=2W5mdzQrwXI

Comparison of TCP Protocols using NS3.
https://www.youtube.com/watch?v=I8jn4vKm5QA

Introduction to installation of network simulator 3 ns3.
https://www.youtube.com/watch?v=T8NwCPROYYA

NS3 NETWORK SIMULATOR. https://youtu.be/ImZyhzZmOwM

NS3 || Tutorial 1. https://www.youtube.com/watch?v=FLttSa45_MI

ns3 Network Simulator - Tips & Advice for beginners.
https://www.youtube.com/watch?v=IMC07d9FNr4

NS3 Introduction | NS3 Tutorial 2.
https://www.youtube.com/watch?v=IZzd2tDox2E

Introduction to Network Simulator 3.
https://www.youtube.com/watch?v=80raQ3VnvyI

The Complete Wireshark Course Beginner To Advanced.
https://www.youtube.com/watch?v=zOYohNOnWp4

The Complete Wireshark Course: Go from Beginner to Advanced!.
https://www.youtube.com/watch?v=vUdOxcRJgME

Learn Wireshark in 10 minutes - Wireshark Tutorial for Beginners.
https://www.youtube.com/watch?v=lb1Dw0elw0Q

❄❄❄❄❄❄❄