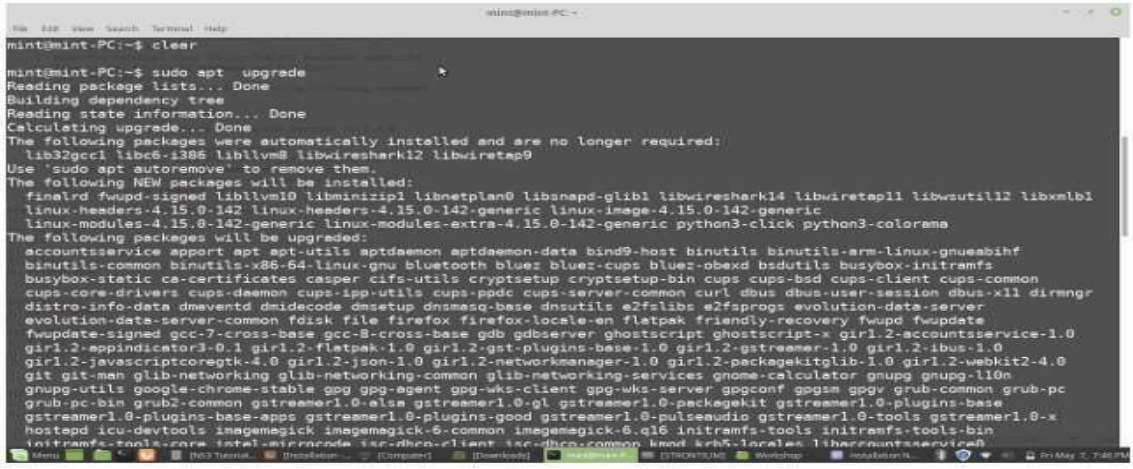
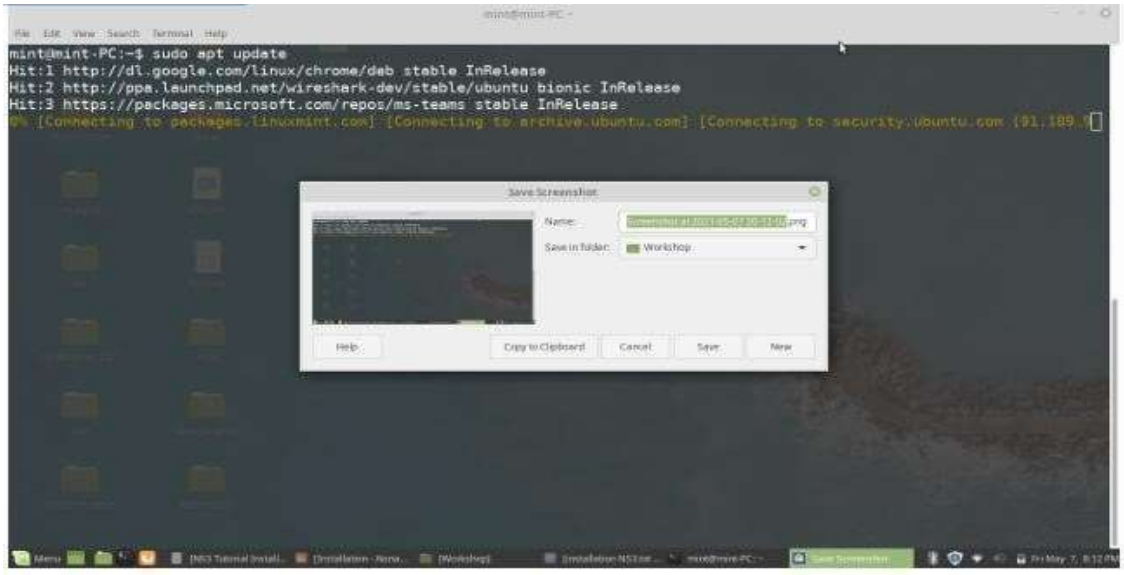
Practical 1

AIM: Installation of NS-3 in Linux.
NS3 Installation
List of Packages for installing ns-3 in Ubantu/Mint Systems
Perquisite for installing NS3.32
Solution

1. sudo apt upgrade

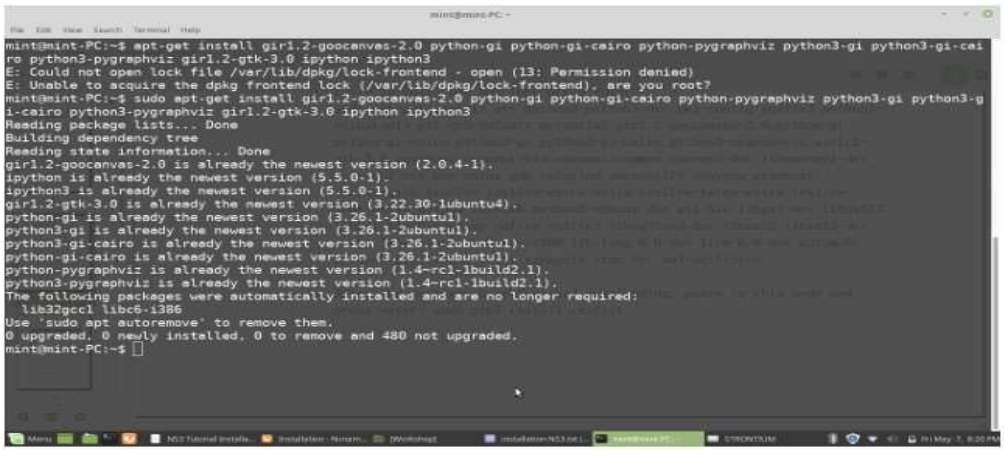


2. Sudo apt update

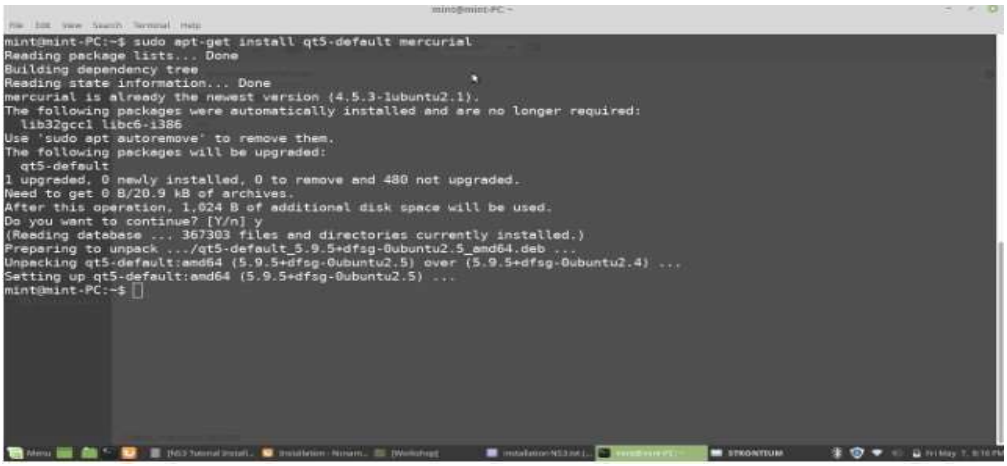


3. Minimal requirements for C++ users
apt-get install g++ python3

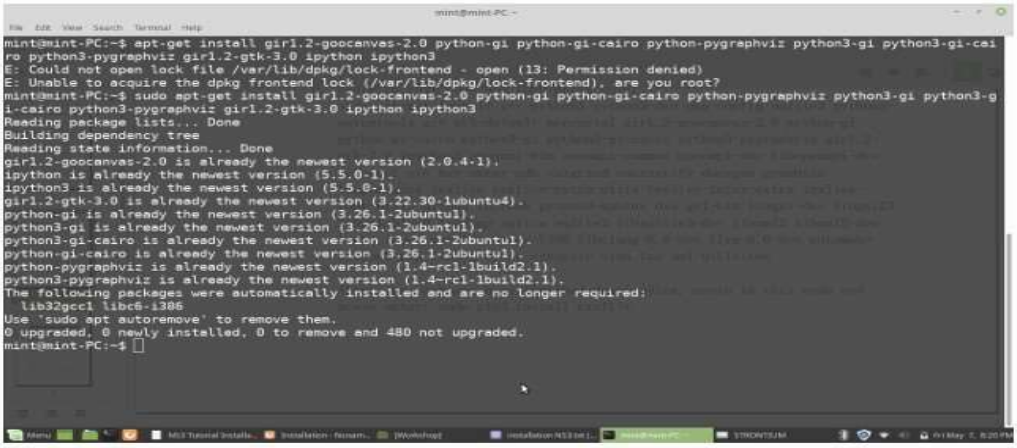
4. **Minimal requirements for Python API users apt-get install g++ python3 python3-dev pkg-config sqlite3**



5. **Netanim animator: qt5 development tools are needed for Netanim animator; apt-get install qt5-default mercurial**

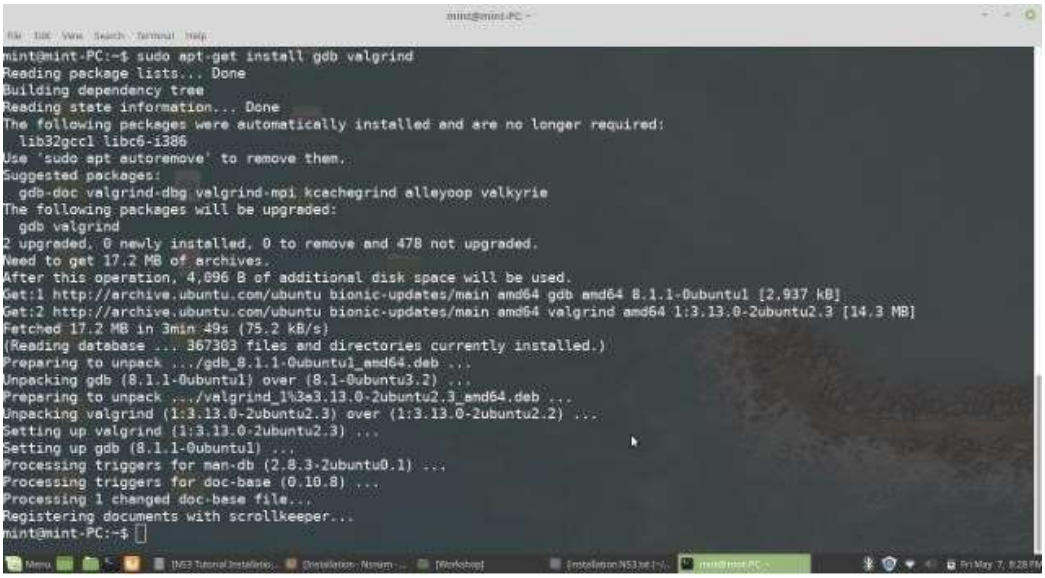


6. **ns-3-pyviz visualizer apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python pygraphviz python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython ipython3**



7. Debugging:

8. apt-get install gdb valgrind

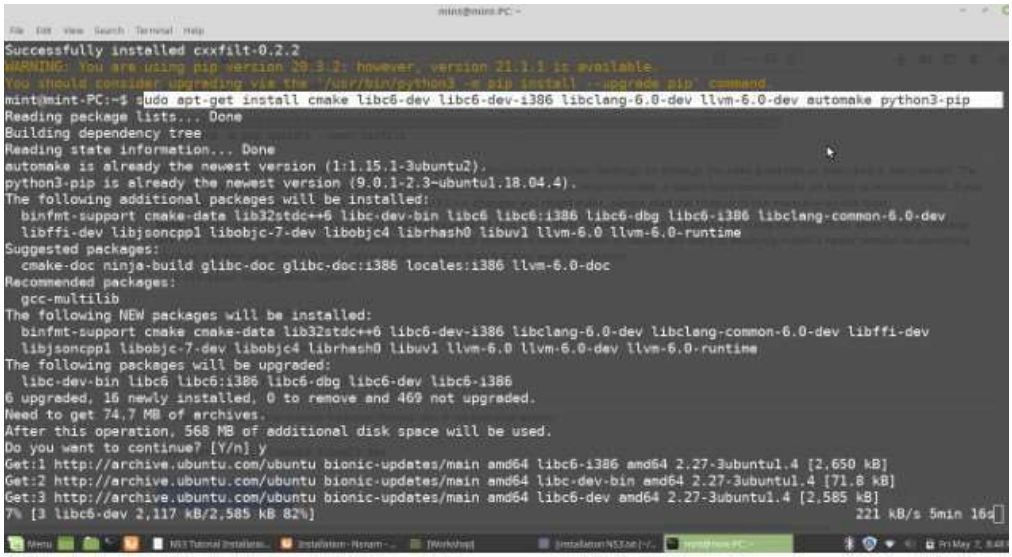


9. Doxygen and related inline documentation: apt-get install doxygen graphviz imagemagick apt-get install texlive texlive-extra-utils texlive-latex extra texlive-font-utils dvipng latexmk

10. The ns-3 manual and tutorial are written in reStructuredText for Sphinx (doc/tutorial, doc/manual, doc/models), and figures typically in dia (also needs the texlive packages above):
apt-get install python3-sphinx dia

11. To read pcap packet traces apt-
get install tcpdump

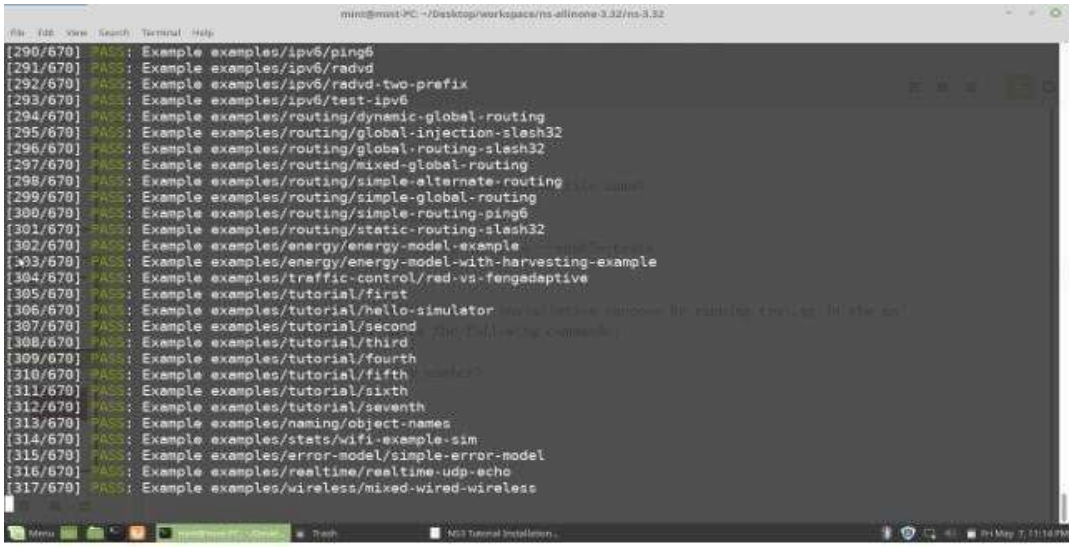
12. Support for generating modified python bindings
apt-get install cmake libc6-dev libc6-dev-i386 libclang-6.0 dev llvm-6.0-dev automake python3-pip python3 m pip install -- user cxxfilt



After installing the required packages, create a folder named workspace in the home directory and then put the NS3 tar package into the workspace. Go to terminal and input these commands consecutively after each command finishes executing:

cd cd
workspa
ce tar xjf
cd
./build.py --enable-examples --enable-
tests It takes time be patient !!

Test the NS3 build and installation success by running test.py in the ns directory using the following commands: cd ns- ./test.py



AIM 2: Installation setup of Net Anim

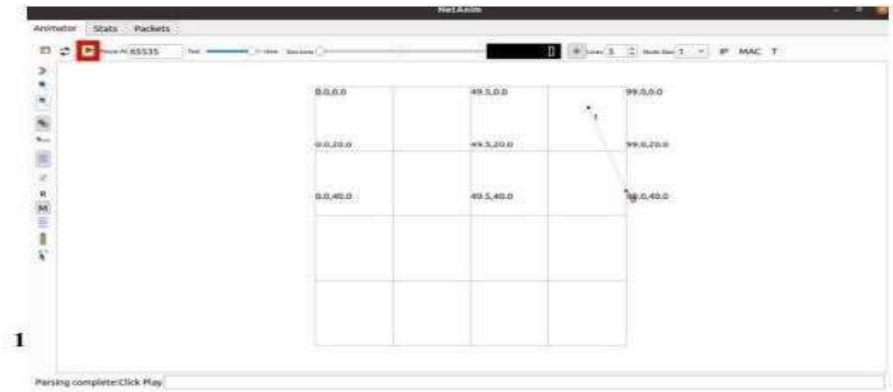
Net Anim Solution:

Net Anim 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.
Installation of NetAnim

You can directly install NetAnim Otherwise, you have to execute some commands but for this we needNS3 installed or compiled.
Step1: sudo apt-get install NetAnim

Step2: NetAnim file.xml

Step3: Select Xml File



Step4: Run the simulation by clicking, NS3 NetAnim successfully.

AIM 3: Installation of Wireshark

Wireshark:

It is the network protocol for analyzing 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.
Features of Wireshark:

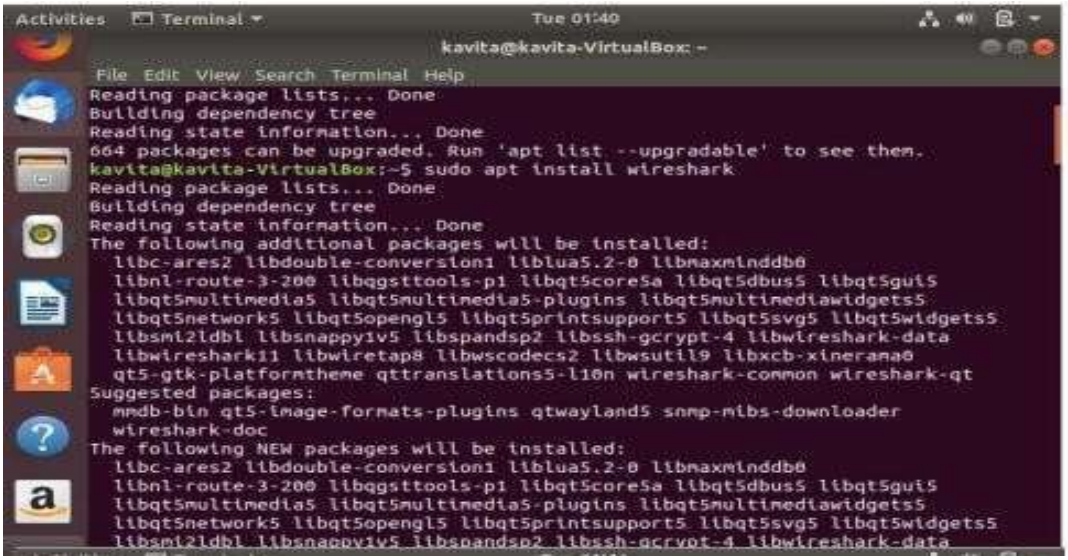
1) **Live capture or offline analyze the packets.**
2) **It runs on multiplatform like Windows,Linux,MacOS,FreeBSD etc.**
3) **It's used in industry and education.**
4) **Many different Read/Write capture file format.**
5) **Colouring the packets for fast analysis.**
6) **In section of hundreds of different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

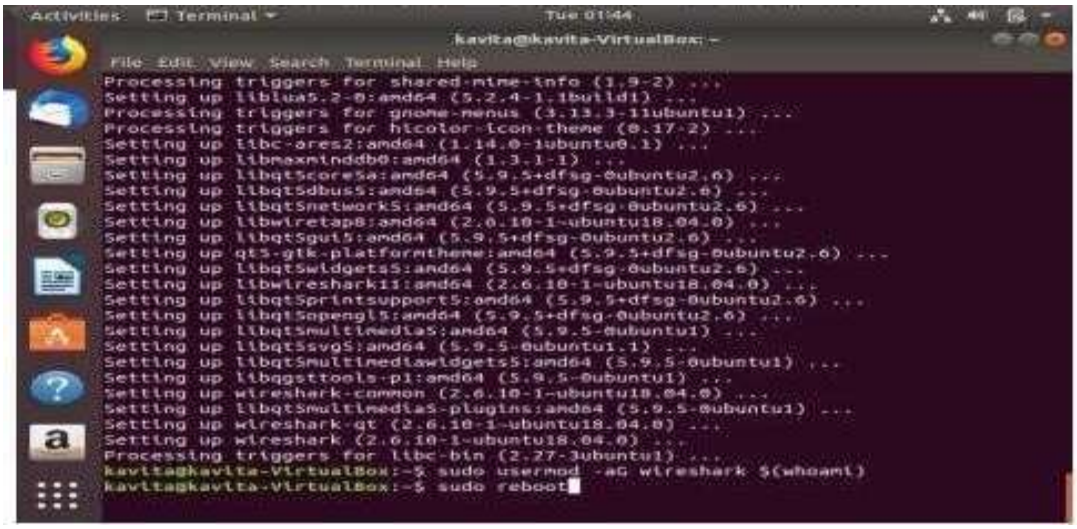
Solution:
Installation of Wireshark:

Step 1: Update the system

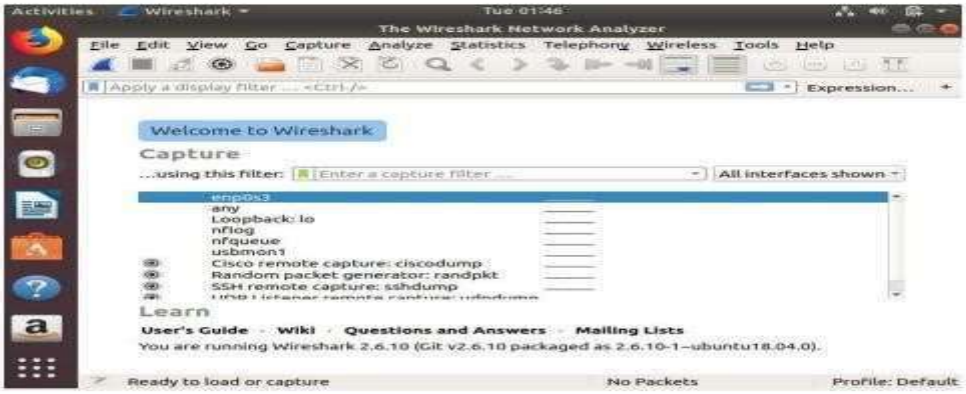
Step2: Install wireshark using sudo apt install wireshark

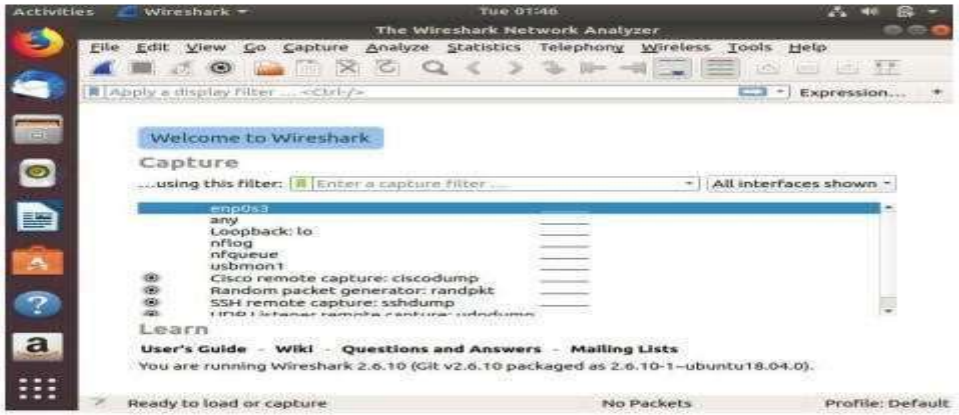
Step3: Add user in wireshark by using usermod -aG wireshark $(whoami) Reboot the system-sudo reboot





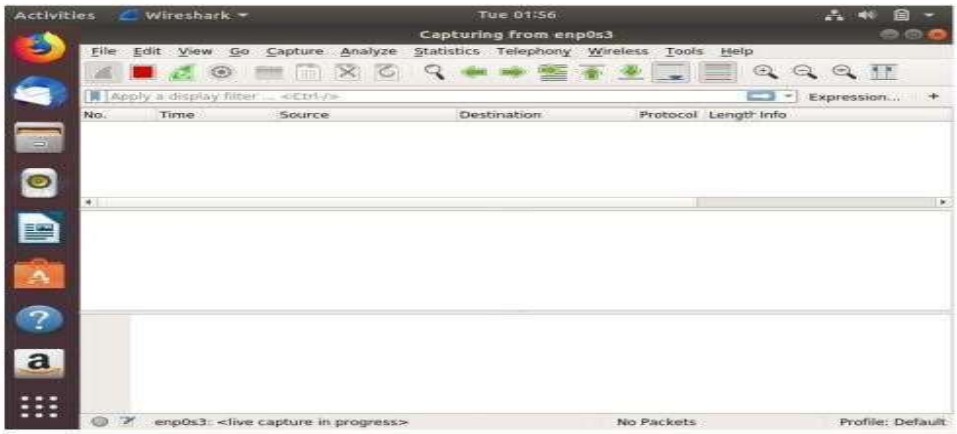
Step4: To run Wireshark type the command (CI) on terminal Wireshark





Wireshark

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



Practical 2

AIM: Linux Network Commands -Ifconfig,Ip,ping ,netstat,traceroute,nslookup,route,hostname.

TERMINAL:

```
admin24@admin24-virtual-machine:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.145.134  netmask 255.255.255.0  broadcast 192.168.145.255
        inet6 fe80::c9:77cb:a197:b11c  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:5b:34:06  txqueuelen 1000  (Ethernet)
        RX packets 243  bytes 224181 (224.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 163  bytes 15180 (15.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 149  bytes 12627 (12.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 149  bytes 12627 (12.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
admin24@admin24-virtual-machine:~$ ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | address | addrlabel | route | rule | neigh | ntable |
                   tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm
 |
                   netns | l2tp | fou | macsec | tcp_metrics | token | netconf
| ila |
                   vrf | sr | nexthop }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -h[uman-readable] | -iec | -j[son] | -p[retty] |
                    -f[amily] { inet | inet6 | mpls | bridge | link } |
                    -4 | -6 | -I | -D | -M | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                    -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename]
 |
                    -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
                    -c[olor]}
```

```
admin24@admin24-virtual-machine:~$ netstat [options]
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 admin24-virtual-m:48058 ec2-13-41-103-230:https TIME_WAIT
udp        0      0 admin24-virtual-:bootpc 192.168.145.254:bootps  ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  4      [ ]         DGRAM      CONNECTED     26621    /run/systemd/notify
unix  2      [ ]         DGRAM                    26635    /run/systemd/journal
/syslog
unix  17     [ ]         DGRAM      CONNECTED     26645    /run/systemd/journal
/dev-log
unix  2      [ ]         DGRAM                    47203    /run/user/1000/syste
md/notify
```

```
admin24@admin24-virtual-machine:~$ traceroute google.com
traceroute to google.com (142.251.42.78), 30 hops max, 60 byte packets
 1  _gateway (192.168.145.2)  0.479 ms  0.403 ms  0.304 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
```

```
admin24@admin24-virtual-machine:~$ route [options]
Usage: route [-nNvee] [-FC] [<AF>]           List kernel routing tables
       route [-v] [-FC] {add|del|flush} ...  Modify routing table for AF.

       route {-h|--help} [<AF>]              Detailed usage syntax for specifie
d AF.
       route {-V|--version}                  Display version/author and exit.

        -v, --verbose            be verbose
        -n, --numeric            don't resolve names
        -e, --extend             display other/more information
        -F, --fib                display Forwarding Information Base (default)
        -C, --cache              display routing cache instead of FIB

  <AF>=Use -4, -6, '-A <af>' or '--<af>'; default: inet
  List of possible address families (which support routing):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
    x25 (CCITT X.25)
admin24@admin24-virtual-machine:~$ hostname
admin24-virtual-machine
```

AIM: ns3 programs-simulate ,visualize, animate the network ,trace and analyze the capture packets.

Udp-echo.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */ /*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
 *
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of *
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
 *
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */
// Network topology
//
//n0    n1   n2   n3
//|    |   |   |
//================
//      LAN
// - UDP flows from n0 to n1 and back
// - DropTail queues
// - Tracing of queues and packet receptions to file "udp-echo.tr"
#include <fstream>
#include "ns3/core-
module.h" #include
"ns3/csma-module.h"
#include
"ns3/applications-
module.h" #include
"ns3/internet-
module.h"

//Net-Anim NameSpace
#include
"ns3/network-
module.h"
#include
"ns3/mobility-
module.h"
#include
"ns3/netanim-
module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE
("UdpEchoExample"); int
main (int argc, char *argv[])
```

Practical 3

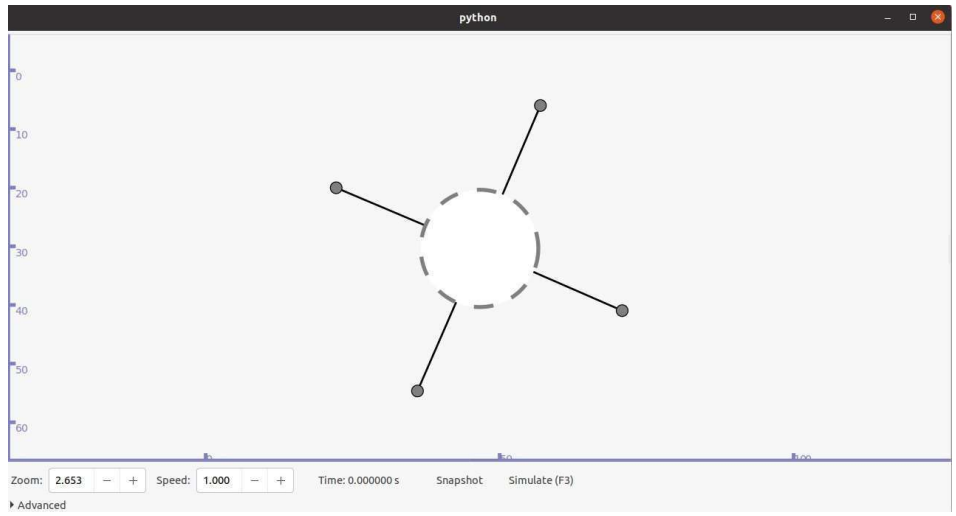
```
{
//
// Users may find it convenient to turn on explicit debugging
// for selected modules; the below lines suggest how to do this
/
/
#
i
f
0
 LogComponentEnable ("UdpEchoExample", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoClientApplication",
LOG_LEVEL_ALL); LogComponentEnable
("UdpEchoServerApplication", LOG_LEVEL_ALL); #endif
//
// Allow the user to override any of the defaults and the above Bind() at
// run-time, via command-line arguments
//
 bool useV6 =
false; Address
serverAddress;

 CommandLine cmd (__FILE__);
cmd.AddValue ("useIpv6", "Use
Ipv6", useV6); cmd.Parse (argc,
argv);
//
// Explicitly create the nodes required by the topology (shown above).
//
 NS_LOG_INFO ("Create
nodes."); NodeContainer n;
n.Create (4);

 InternetStackHel
per internet;
internet.Install (n);

 NS_LOG_INFO ("Create channels.");
//
// Explicitly create the channels required by the topology (shown above).
//
 CsmaHelper csma;
 csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));
```

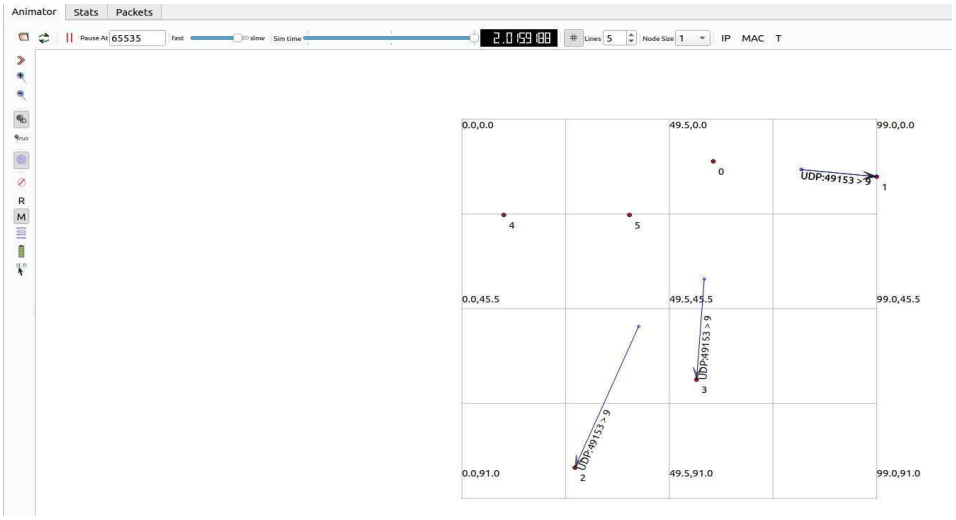
```
 NetDeviceContainer d = csma.Install (n);

//
// We've got the "hardware" in place.  Now we need to add IP addresses.
//
 NS_LOG_INFO ("Assign IP
Addresses."); if (useV6 == false)
  {
   Ipv4AddressHelper ipv4;
                                    i
pv4.SetBase ("10.1.1.0",
"255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign (d);
    serverAddress = Address(i.GetAddress
    (1));
  }
 else
  {
   Ipv6AddressHelper ipv6;
   ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix
(64)); Ipv6InterfaceContainer i6 = ipv6.Assign (d);
    serverAddress = Address(i6.GetAddress (1,1));
  }
 NS_LOG_INFO ("Create Applications.");
//
// Create a UdpEchoServer application on node one.
//
 uint16_t port = 9;  // well-known echo port
number UdpEchoServerHelper server
(port); ApplicationContainer apps =
server.Install (n.Get (1)); apps.Start
(Seconds (1.0));
 apps.Stop (Seconds (10.0));

//
// Create a UdpEchoClient application to send UDP datagrams from
node zero to // node one.
//
 uint32_t packetSize = 1024;  uint32_t maxPacketCount = 1;
Time interPacketInterval = Seconds (1.);
UdpEchoClientHelper client (serverAddress, port);
client.SetAttribute ("MaxPackets", UintegerValue
(maxPacketCount));  client.SetAttribute ("Interval",
TimeValue (interPacketInterval));  client.SetAttribute
("PacketSize", UintegerValue (packetSize));
 apps = client.Install (n.Get
(0)); apps.Start (Seconds
(2.0));
 apps.Stop (Seconds (10.0));
```

```
#if 0
//
// Users may find it convenient to initialize echo packets with actual data;
// the below lines suggest how to do this
//
  client.SetFill (apps.Get (0),
  "Hello World"); client.SetFill
  (apps.Get (0), 0xa5, 1024);
  uint8_t fill[] = { 0, 1, 2, 3, 4, 5, 6};
  client.SetFill (apps.Get (0), fill,
sizeof(fill), 1024); #endif

  AsciiTraceHelper ascii;
  csma.EnableAsciiAll (ascii.CreateFileStream ("udp-
echo.tr")); csma.EnablePcapAll ("udp-echo", false);

//Net Animator Code

 NodeContainer csmaNodes;
  csmaNodes.Create(2);  // Create
  two nodes

  MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(csmaNodes);

  AnimationInterface anim("udp.xml");
  AnimationInterface::SetConstantPosition
  (csmaNodes.Get(0),10,25);
AnimationInterface::SetConstantPosition (csmaNodes.Get(1),40,25);
anim.EnablePacketMetadata((true));
//End
//
// Now, do the actual simulation.
//
  NS_LOG_INFO ("Run
  Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO
  ("Done.");
}
```

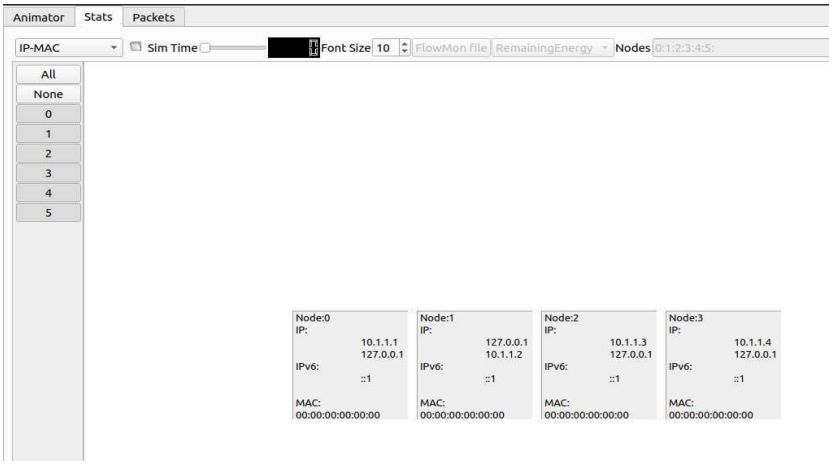
TERMINAL:

```
ubantu@ubuntu:~/workspace/ns-allinone-3.32$ cd ns-3.32/
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/udp-echo
Waf: Entering directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.293s)
```


python
Zoom: 2.653
Speed: 1.000
Time: 0.000000 s
Snapshot
Simulate (F3)
Advanced


Animator
Stats
Packets
IP
MAC
0.0,0.0
49.5,0.0
99.0,0.0
0.0,45.5
49.5,45.5
99.0,45.5
0.0,91.0
49.5,91.0
99.0,91.0
UDP:49153 > 9

OUTPUT:


Animator
Stats
Packets
IP-MAC
Sim Time
Font Size
10
FlowMon file
RemainingEnergy
Nodes
All
None
0
1
2
3
4
5
Node:0
IP:
10.1.1.1
127.0.0.1
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:1
IP:
127.0.0.1
10.1.1.2
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:2
IP:
10.1.1.3
127.0.0.1
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:3
IP:
10.1.1.4
127.0.0.1
IPv6:
::1
MAC:
00:00:00:00:00:00

udptable

From Id	To Id	Tx	Meta
0	1	2.008	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	1	2.008	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	2	2.008	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	2	2.008	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	3	2.008	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	3	2.008	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.1 DstIp : 10.1.1.2
1	0	2.0101	Arp reply SMac: 00:00:00:00:00:02 DMac: 00:00:00:00:00:01 SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	0	2.0101	Arp reply SMac: 00:00:00:00:00:02 DMac: 00:00:00:00:00:01 SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	2	2.0101	Arp reply SMac: 00:00:00:00:00:02 DMac: 00:00:00:00:00:01 SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	3	2.0101	Arp reply SMac: 00:00:00:00:00:02 DMac: 00:00:00:00:00:01 SrcIp : 10.1.1.2 DstIp : 10.1.1.1
0	1	2.01221	UDP 49153 > 9
0	1	2.01221	UDP 49153 > 9
0	2	2.01221	UDP 49153 > 9
0	3	2.01221	UDP 49153 > 9
1	0	2.01692	Arp request SMac: 00:00:00:00:00:02 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	0	2.01692	Arp request SMac: 00:00:00:00:00:02 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	2	2.01692	Arp request SMac: 00:00:00:00:00:02 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	2	2.01692	Arp request SMac: 00:00:00:00:00:02 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	3	2.01692	Arp request SMac: 00:00:00:00:00:02 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.2 DstIp : 10.1.1.1
1	3	2.01692	Arp request SMac: 00:00:00:00:00:02 DMac: ff:ff:ff:ff:ff:ff SrcIp : 10.1.1.2 DstIp : 10.1.1.1
0	1	2.01902	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:02 SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	1	2.01902	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:02 SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	2	2.01902	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:02 SrcIp : 10.1.1.1 DstIp : 10.1.1.2
0	3	2.01902	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:02 SrcIp : 10.1.1.1 DstIp : 10.1.1.2
1	0	2.02113	UDP 9 > 49153
1	0	2.02113	UDP 9 > 49153
1	2	2.02113	UDP 9 > 49153
1	3	2.02113	UDP 9 > 49153

Practical 4

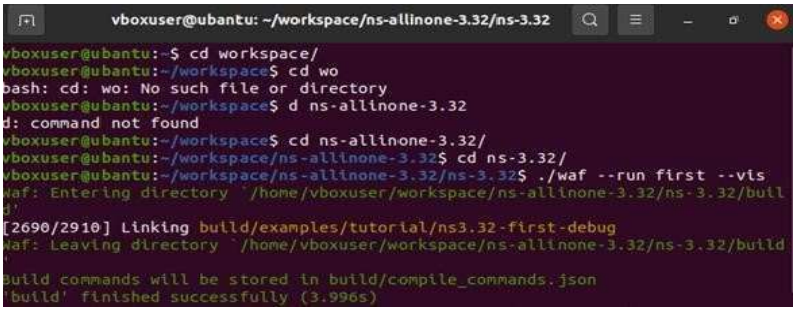
AIM: Program to simulate Point to Point topology.

CODE:

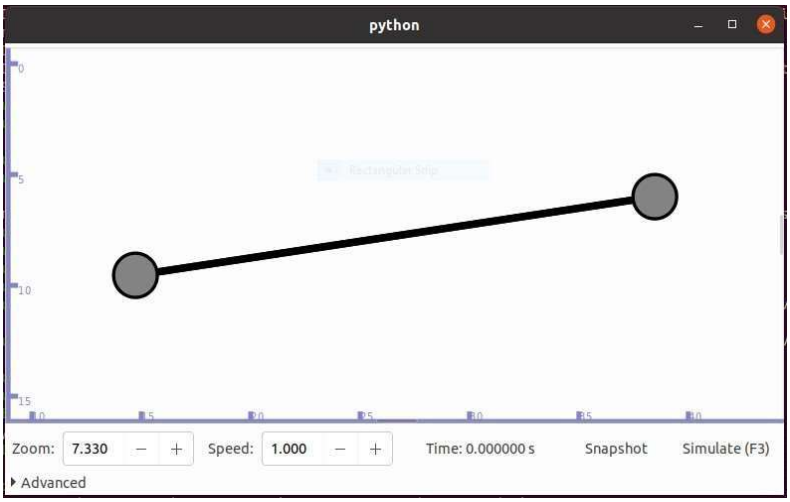
```
#include "ns3/core-
module.h" #include
"ns3/network-
module.h" #include
"ns3/internet-
module.h"
#include  "ns3/point-to-point-module.h"
#include "ns3/applications-
module.h" using
namespace ns3;
 NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int
main (int argc, char *argv[])
{
 Time::SetResolution (Time::NS);
 LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
                                                    LogComponentEnable
("UdpEchoServerApplication", LOG_LEVEL_INFO);     NodeContainer nodes;
                                                    nodes.Create (2);
 PointToPointHelper pointToPoint;
                                  pointToPoint.SetDeviceAttrib
ute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue
("2ms")); NetDeviceContainer devices;    devices =
pointToPoint.Install (nodes);
InternetStackHelper stack;      stack.Install (nodes);
                                 Ipv4AddressHelper address;
 address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);      UdpEchoServerHelper
echoServer (9); ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
                                                                   serverApps.Start (Seconds
(1.0)); serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient
(interfaces.GetAddress (1), 9); echoClient.SetAttribute
("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds
(1.0))); echoClient.SetAttribute ("PacketSize",
UintegerValue (1024)); ApplicationContainer clientApps
= echoClient.Install (nodes.Get (0)); clientApps.Start
(Seconds (2.0));                      clientApps.Stop
(Seconds (10.0));
 Simulator::Run
();
```

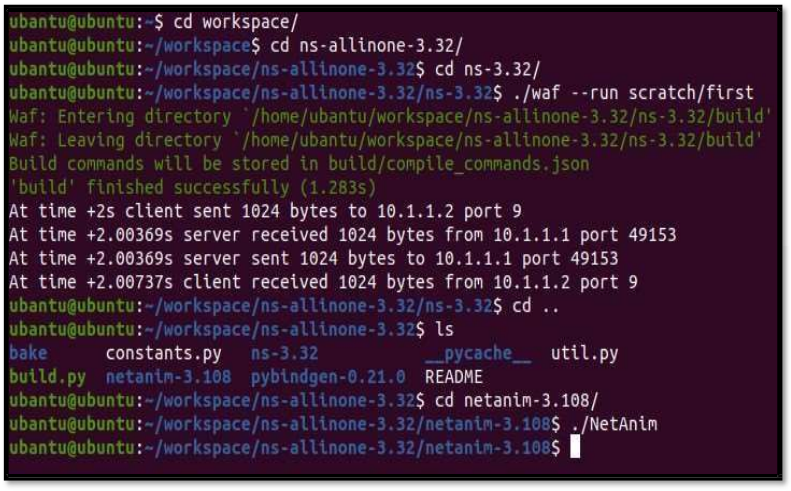
Simulator::Destroy (); **return 0;**
}

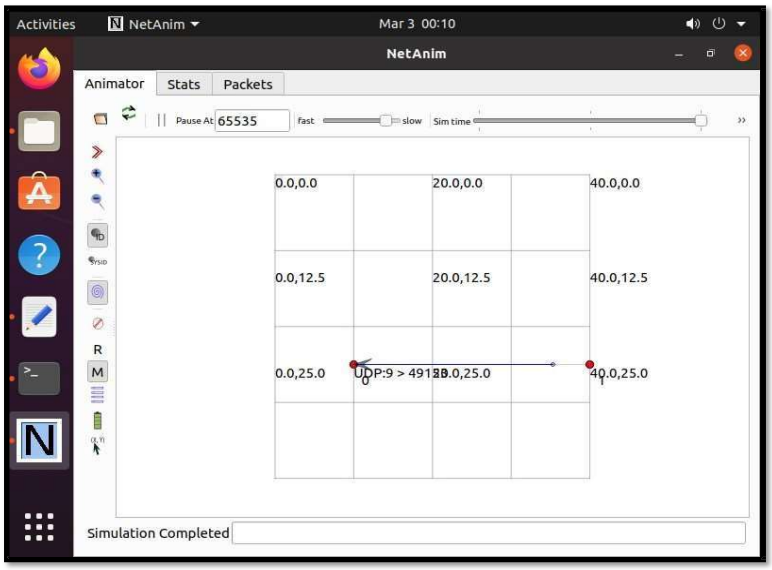
TERMINAL:



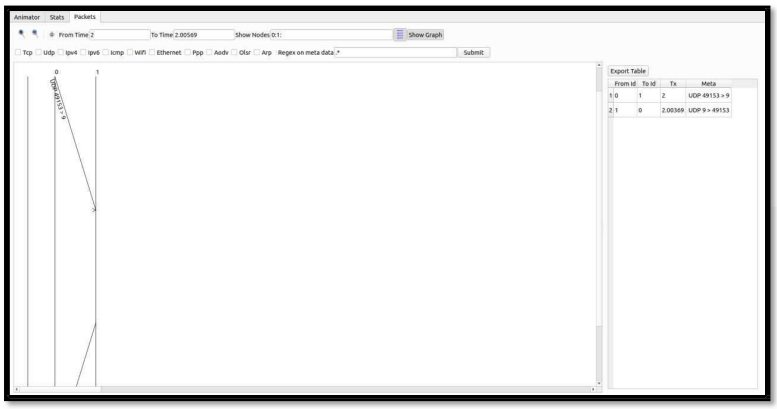
OUTPUT:






Activities
NetAnim
Mar 3 00:10
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
0.0,0.0
20.0,0.0
40.0,0.0
0.0,12.5
20.0,12.5
40.0,12.5
0.0,25.0
20.0,25.0
40.0,25.0
Simulation Completed


Animator
Stats
Packets
IP-MAC
Sim Time
Font Size
Nodes
All
None
0
1


Animator
Stats
Packets
Show Graph
Export Table

Practical 05

AIM: Program to simulate Bus topology.

CODE:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2 /*
3 * This program is free software; you can redistribute it and/or modify
4 * it under the terms of the GNU General Public License version 2 as
5 * published by the Free Software Foundation;
6 *
7 * This program is distributed in the hope that it will be useful,
8 * but WITHOUT ANY WARRANTY; without even the implied warranty of 9 *
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
10 * GNU General Public License for more details.
11 *
12 * You should have received a copy of the GNU General Public License
13 * along with this program; if not, write to the Free Software
14 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
15 */
  16
17 #include "ns3/core-module.h"
18 #include "ns3/network-module.h"
19 #include "ns3/csma-module.h"
20 #include "ns3/internet-module.h"
21 #include "ns3/point-to-point-module.h"
22 #include "ns3/applications-module.h"
23 #include "ns3/ipv4-global-routing-helper.h"
  24
25 // Default Network Topology
26 //
27 // 10.1.1.0
28 // n0 ---------- n1 n2 n3 n4
29 // point-to-point | | | |
30 // ================
31 // LAN 10.1.2.0
  32
  33
  34 using
  namespace
  ns3; 35
  36 NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
3
7
 3
 8
 i
 n
 t
```

```
39  main (int argc, char *argv[])
40  {
41  bool verbose = true;
42  uint32_t nCsma = 3;
  43
44  CommandLine cmd (__FILE__);
  45  cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);    46
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  47
  48  cmd.Parse (argc,argv);
  49
50  if (verbose)
51  {
52  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
53  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
54  }
  55
  56  nCsma = nCsma == 0 ?
  1 : nCsma; 57
  58  NodeContainer p2pNodes;
  59  p2pNodes.Create (2);
  60
61  NodeContainer csmaNodes;
62  csmaNodes.Add (p2pNodes.Get (1));
  63  csmaNodes.Create (nCsma);
  64
65  PointToPointHelper pointToPoint;
66  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
67  pointToPoint.SetChannelAttribute ("Delay", StringValue
  ("2ms")); 68
69  NetDeviceContainer p2pDevices;
70  p2pDevices = pointToPoint.Install
  (p2pNodes); 71
  72  CsmaHelper csma;
73  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
74  csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
  75
76  NetDeviceContainer csmaDevices;
77  csmaDevices = csma.Install
  (csmaNodes); 78
  79  InternetStackHelper stack;
80  stack.Install (p2pNodes.Get (0));
81  stack.Install (csmaNodes);
  82
  83  Ipv4AddressHelper address;
84  address.SetBase ("10.1.1.0", "255.255.255.0");
85  Ipv4InterfaceContainer p2pInterfaces;
  86  p2pInterfaces = address.Assign
  (p2pDevices); 87
88  address.SetBase ("10.1.2.0", "255.255.255.0");
```

```
89  Ipv4InterfaceContainer csmaInterfaces;
90  csmaInterfaces = address.Assign
  (csmaDevices);
91
  92  UdpEchoServerHelper echoServer (9);
  93
  94  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
95  serverApps.Start (Seconds (1.0));
96  serverApps.Stop (Seconds (10.0));
  97
  98  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
99   echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
100  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
101  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));   102
 103  ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
104  clientApps.Start (Seconds (2.0));
105  clientApps.Stop (Seconds (10.0));
 106
 107  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
 108
109  pointToPoint.EnablePcapAll ("second");
110  csma.EnablePcap ("second", csmaDevices.Get (1), true);
 111
 112  Simulator::Run ();   113  Simulator::Destroy ();
 114  return 0;
}
```

NETANIM CODE :

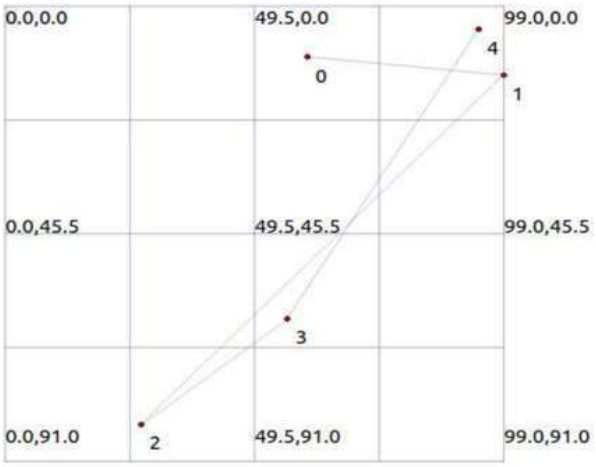
```
      #include "ns3/network-module.h"
1.    #include "ns3/csma-module.h"
2.    #include "ns3/internet-module.h"
3.    #include "ns3/point-to-point-module.h"
4.    #include "ns3/applications-module.h"
5.    #include "ns3/ipv4-global-routing-helper.h"
6.    #include "ns3/netanim-module.h"
7.    #include "ns3/mobility-module.h"
8.    // Default Network Topology
9.    //
10.   //       10.1.1.0
11.   // n0 ---------- n1   n2   n3   n4
12.   //    point-to-point |   |   |   |
13.   //                   ================
14.   //                     LAN 10.1.2.0
15.   using namespace ns3;
16.   NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
17.   int
18.   main (int argc, char *argv[])
19.   {
20.   bool verbose = true;
21.   uint32_t nCsma = 3;
```

```
22. CommandLine cmd (__FILE__);
23. cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
24. cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose); 25. cmd.Parse
    (argc,argv);
26. if (verbose)
27. {
28. LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
29. LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
30. }
31. //
32. nCsma = nCsma == 0 ? 1 : nCsma;
33. NodeContainer p2pNodes;
34. p2pNodes.Create (2);
35. NodeContainer csmaNodes;
36. csmaNodes.Add (p2pNodes.Get (1));
37. csmaNodes.Create (nCsma);
38. // creating node
39. PointToPointHelper pointToPoint;
40. pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
41. pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
42. //creating channel
43. NetDeviceContainer p2pDevices;
44. p2pDevices = pointToPoint.Install (p2pNodes);
45. //
46. CsmaHelper csma;
47. csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
48. csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

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

51. InternetStackHelper stack;
52. stack.Install (p2pNodes.Get (0));
53. stack.Install (csmaNodes);
54. Ipv4AddressHelper address;
55. address.SetBase ("10.1.1.0", "255.255.255.0");
56. Ipv4InterfaceContainer p2pInterfaces;
57. p2pInterfaces = address.Assign (p2pDevices);
58. //point to point device
59. address.SetBase ("10.1.2.0", "255.255.255.0");
60. Ipv4InterfaceContainer csmaInterfaces;
61. csmaInterfaces = address.Assign (csmaDevices);
62. //cmsa ddevice
63. UdpEchoServerHelper echoServer (9);
64. ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
65. serverApps.Start (Seconds (1.0));
66. serverApps.Stop (Seconds (10.0));
67. UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
68. echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
69. echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
```

```
70. echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
71. ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
72. clientApps.Start (Seconds (2.0));
73. clientApps.Stop (Seconds (10.0));
74. Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
75. MobilityHelper mobility;
76. mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
77. mobility.Install(csmaNodes);
78. AnimationInterface anim("second.xml");
79. AnimationInterface::SetConstantPosition (csmaNodes.Get(0), 10,25);
80. AnimationInterface::SetConstantPosition (csmaNodes.Get(1), 40,25);
81. anim.EnablePacketMetadata((true));
82. pointToPoint.EnablePcapAll ("second.xml");
83. csma.EnablePcap ("second", csmaDevices.Get (1), true);
84. Simulator::Run ();
85. Simulator::Destroy ();
86. return 0;
87. }
```

TERMINAL:

OUTPUT:



PRATICAL 6

AIM: Program to simulate star topology.

CODE:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of *
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 */
#include "ns3/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"

// Network topology (default)
//
//       n2 n3 n4              .
//        \ | /                .
//         \|/                 .
//     n1--- n0---n5  .
//          /|\                .
//         / | \               .
//       n8 n7 n6      .
//
using namespace ns3;
NS_LOG_COMPONENT_DEFINE
("StarAnimation"); int
main (int argc, char *argv[])
```

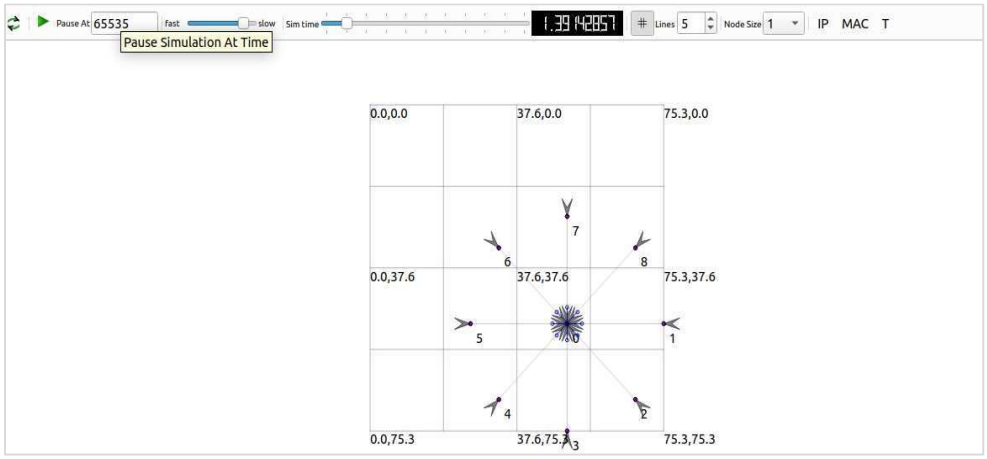
```
    {
// Set up some default values for the simulation.
//
Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (137));

// 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;
std::string animFile = "star-animation.xml"; uint8_t useIpv6 = 0;
Ipv6Address ipv6AddressBase = Ipv6Address("2001::"); Ipv6Prefix ipv6AddressPrefix = Ipv6Prefix(64);

CommandLine cmd;
cmd.AddValue ("nSpokes", "Number of spoke nodes to place in the star", nSpokes); cmd.AddValue
("animFile", "File Name for Animation Output", animFile); cmd.AddValue ("useIpv6", "use Ipv6",
useIpv6);

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."); if (useIpv6 == 0)
{
star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));
}
else
{
star.AssignIpv6Addresses (ipv6AddressBase, ipv6AddressPrefix);
}

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

uint16_t port = 50000; Address hubLocalAddress; if (useIpv6 == 0)
{

hubLocalAddress = InetSocketAddress (Ipv4Address::GetAny (), port);
} else
{
```

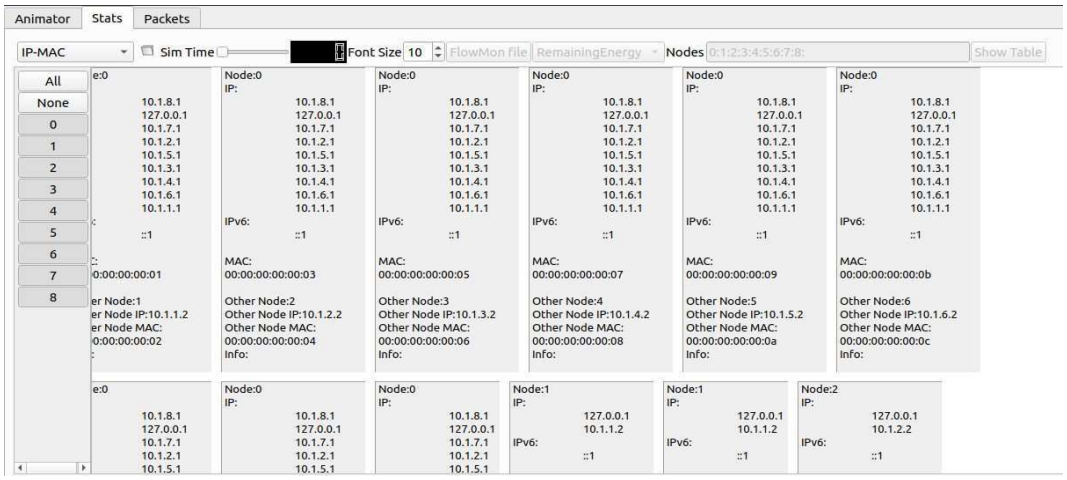
```
hubLocalAddress = Inet6SocketAddress (Ipv6Address::GetAny (), port);
}
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", hubLocalAddress);
ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ()); hubApp.Start (Seconds
(1.0));
hubApp.Stop (Seconds (10.0));

//
// 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; if (useIpv6 == 0)
{
remoteAddress = AddressValue(InetSocketAddress (star.GetHubIpv4Address (i), port));
} else
{
remoteAddress = AddressValue(Inet6SocketAddress (star.GetHubIpv6Address (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.
//
if (useIpv6 == 0)
{
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
}

// Set the bounding box for animation star.BoundingBox (1, 1, 100, 100);
// Create the animation object and configure for specified output AnimationInterface anim (animFile);
NS_LOG_INFO ("Run Simulation."); Simulator::Run (); Simulator::Destroy (); NS_LOG_INFO
("Done.");

return 0;
}
```

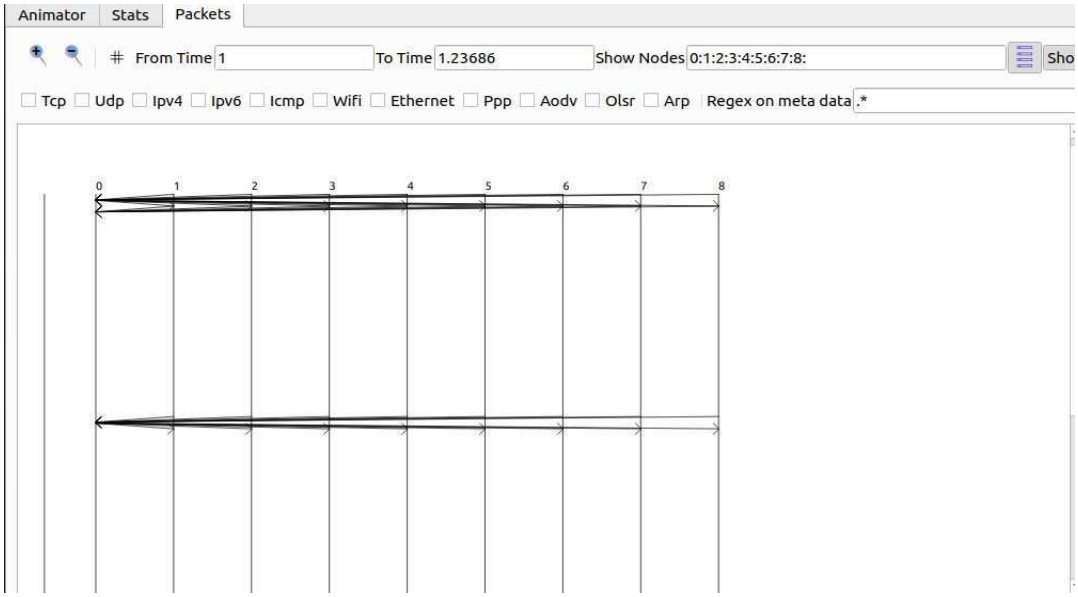
TERMINAL:

```
admin24@admin24-virtual-machine:~$ cd workspace/
admin24@admin24-virtual-machine:~/workspace$ cd ns-allinone-3.32/
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32$ cd ns-3.32/
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf build
Waf: Entering directory `/home/admin24/workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/admin24/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.957s)

Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
dsdv                     dsr                      energy
fd-net-device            flow-monitor             internet
internet-apps            lr-wpan                  lte
mesh                     mobility                 netanim
network                  nix-vector-routing       olsr
point-to-point           point-to-point-layout    propagation
sixlowpan                spectrum                 stats
tap-bridge               test (no Python)         topology-read
traffic-control          uan                      virtual-net-device
wave                     wifi                     wimax

Modules not built (see ns-3 tutorial for explanation):
brite                    click                    dpdk-net-device
mpi                      openflow                 visualizer

admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32$ cd ..
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32$ cd netanim-3.108/
```

OUTPUT:




Animator
Stats
Packets
IP-MAC
Sim Time
Font Size
10
FlowMon File
RemainingEnergy
Nodes
Show Table
All
None
0
1
2
3
4
5
6
7
8


Animator
Stats
Packets
From Time
1
To Time
1.23686
Show Nodes
0:1:2:3:4:5:6:7:8:
Tcp
Udp
Ipv4
Ipv6
Icmp
Wifi
Ethernet
Ppp
Aodv
Olsr
Arp
Regex on meta data
.*
0
1
2
3
4
5
6
7
8

Practical 7

AIM: Program to simulate hybrid topology.
CODE:

```
1  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2  /*
3  * This program is free software; you can redistribute it and/or modify
4  * it under the terms of the GNU General Public License version 2 as
5  * published by the Free Software Foundation;
6  *
7  * This program is distributed in the hope that it will be useful,
8  * but WITHOUT ANY WARRANTY; without even the implied warranty of     9 *
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
10  * GNU General Public License for more details.
11  *
12  * You should have received a copy of the GNU General Public License
13  * along with this program; if not, write to the Free Software
14  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
15  */
  16
17 #include "ns3/core-module.h"
18 #include  "ns3/point-to-point-module.h"
19 #include  "ns3/network-module.h"
20 #include  "ns3/applications-module.h"
21 #include "ns3/wifi-module.h"
22 #include  "ns3/mobility-module.h"
23 #include  "ns3/csma-module.h"
24 #include  "ns3/internet-module.h"
  25
26 // Default Network Topology
27 //
28 // Number of wifi or csma nodes can be increased up to 250
29 // |
30 // Rank 0 | Rank 1
31 //  -------------------------|----------------------------     32 // Wifi 10.1.3.0
33 // AP
34 // * * * *
35 // | | | | 10.1.1.0
36 // n5 n6 n7 n0 ---------------- n1 n2 n3 n4
37 // point-to-point | | | |
38 // ================
39 // LAN 10.1.2.0
  40
  41 using
  namespace
  ns3; 42
```

```
      43 NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
4
4
45 i
   n
   t
    46 main (int argc, char *argv[])
    47 {
    48 bool verbose = true;
    49 uint32_t nCsma = 3;
    50 uint32_t nWifi = 3;
    51 bool
      tracing =
      false; 52
      53 CommandLine cmd;
    54 cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
    55 cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
    56 cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
    57 cmd.AddValue ("tracing", "Enable pcap
      tracing", tracing); 58
      59 cmd.Parse (argc,argv);
      60
    61 // Check for valid number of csma or wifi nodes
    62 // 250 should be enough, otherwise IP addresses
    63 // soon become an issue
    64 if (nWifi > 250 || nCsma > 250)
    65 {
    66 std::cout << "Too many wifi or csma nodes, no more than 250 each." << std::endl;    67 return 1;
      68 }
      69
    70 if (verbose)
    71 {
    72 LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    73 LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    74 }
      75
      76 NodeContainer p2pNodes;
      77 p2pNodes.Create (2);
      78
    79 PointToPointHelper pointToPoint;
    80 pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    81 pointToPoint.SetChannelAttribute ("Delay", StringValue
      ("2ms")); 82
    83 NetDeviceContainer p2pDevices;
    84 p2pDevices = pointToPoint.Install
      (p2pNodes); 85
      86 NodeContainer csmaNodes;
    87 csmaNodes.Add (p2pNodes.Get (1));
    88 csmaNodes.Create (nCsma);
      89
```

```
90  CsmaHelper csma;
91  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
  92 csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
93
94  NetDeviceContainer csmaDevices;
95  csmaDevices = csma.Install (csmaNodes);
96
97  NodeContainer wifiStaNodes;
98  wifiStaNodes.Create (nWifi);
  99  NodeContainer wifiApNode = p2pNodes.Get (0);
 100
101  YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
102  YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
103  phy.SetChannel
 (channel.Create ()); 104
 105  WifiHelper wifi = WifiHelper::Default ();
 106  wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
 107
 108  NqosWifiMacHelper mac =
 NqosWifiMacHelper::Default (); 109
110  Ssid ssid = Ssid ("ns-3-ssid");
111  mac.SetType ("ns3::StaWifiMac",
112  "Ssid", SsidValue (ssid),
113  "ActiveProbing",
 BooleanValue (false)); 114
115  NetDeviceContainer staDevices;
116  staDevices = wifi.Install (phy, mac,
 wifiStaNodes); 117
118  mac.SetType ("ns3::ApWifiMac",
119  "Ssid",
 SsidValue (ssid));
 120
121  NetDeviceContainer apDevices;
122  apDevices = wifi.Install (phy, mac,
 wifiApNode); 123
 124  MobilityHelper mobility;
 125
126  mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
127  "MinX", DoubleValue (0.0),
128  "MinY", DoubleValue (0.0),
129  "DeltaX", DoubleValue (5.0),
130  "DeltaY", DoubleValue (10.0),
131  "GridWidth", UintegerValue (3),
132  "LayoutType", StringValue
 ("RowFirst")); 133
134  mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
135  "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
136  mobility.Install (wifiStaNodes);
 137
```

```
138  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
139  mobility.Install (wifiApNode);
140
141    InternetStackHelper stack;
142  stack.Install (csmaNodes);
143  stack.Install (wifiApNode);
144  stack.Install (wifiStaNodes);
145
146 Ipv4AddressHelper address;
147
148  address.SetBase ("10.1.1.0", "255.255.255.0");
149  Ipv4InterfaceContainer p2pInterfaces;
150  p2pInterfaces = address.Assign
 (p2pDevices); 151
152  address.SetBase ("10.1.2.0", "255.255.255.0");
153  Ipv4InterfaceContainer csmaInterfaces;
154  csmaInterfaces = address.Assign
 (csmaDevices); 155
156  address.SetBase ("10.1.3.0", "255.255.255.0");
157  address.Assign (staDevices);
158  address.Assign (apDevices);
159
160  UdpEchoServerHelper echoServer (9);
161
162  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
163  serverApps.Start (Seconds (1.0));
164  serverApps.Stop (Seconds (10.0));
165
166  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
167  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
168  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
169  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));   170
171  ApplicationContainer clientApps =
172  echoClient.Install (wifiStaNodes.Get (nWifi - 1));
173  clientApps.Start (Seconds (2.0));
174  clientApps.Stop (Seconds (10.0));
175
176  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
177
178  Simulator::Stop (Seconds (10.0));
179
180  if (tracing == true)
181  {
182  pointToPoint.EnablePcapAll ("third");
183  phy.EnablePcap ("third", apDevices.Get (0));
184  csma.EnablePcap ("third", csmaDevices.Get (0), true);
185  }
186
187  Simulator::Run ();   188  Simulator::Destroy ();
189  return 0;
```

```
190 }

NETANIM CODE:
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
1.  /*

2.  *

3.  *

4.  */

•   This program is free software; you can redistribute it and/or modify
•   it under the terms of the GNU General Public License version 2 as
•   published by the Free Software Foundation;

•   This program is distributed in the hope that it will be useful,
•   but WITHOUT ANY WARRANTY; without even the implied warranty of •
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
•   GNU General Public License for more details.

•   You should have received a copy of the GNU General Public License
•   along with this program; if not, write to the Free Software
•   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA


5.  #include "ns3/core-module.h"
6.  #include "ns3/point-to-point-module.h"
7.  #include "ns3/network-module.h"
8.  #include "ns3/applications-module.h"
9.  #include "ns3/mobility-module.h"
10. #include "ns3/csma-module.h"
11. #include "ns3/internet-module.h"
12. #include "ns3/yans-wifi-helper.h"
13. #include "ns3/ssid.h"
14. // second cc
15. #include "ns3/netanim-module.h"
16. #include "ns3/mobility-module.h"

17. // Default Network Topology
18. //
19. // Wifi 10.1.3.0
20. //          AP
21. // *        *          *          *
22. // |        |          | |  10.1.1.0
23. // n5 n6 n7 n0 -------------- n1 n2 n3 n4
24. //          point-to-point |   |          | |
25. //          ================
26. //          LAN 10.1.2.0
27. //forming ring
```

```
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 44. // allocator's configuration; the grid layout will exceed the
// bounding box if more than 18 nodes are provided.
if (nWifi > 18) 47. {
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);
```

```
71. NodeContainer wifiStaNodes;
72. wifiStaNodes.Create (nWifi);


73. NodeContainer wifiApNode = p2pNodes.Get (0);

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

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

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

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

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

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

87. MobilityHelper mobility;

88. mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
i. "MinX", DoubleValue (0.0), ii. "MinY", DoubleValue (0.0),
iii. "DeltaX", DoubleValue (5.0),
iv. "DeltaY", DoubleValue (10.0),
v. "GridWidth", UintegerValue (3),
vi. "LayoutType", StringValue ("RowFirst"));

89. mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
i. "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50))); 90. mobility.Install (wifiStaNodes);

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

93. InternetStackHelper stack;
94. stack.Install (csmaNodes);
95. stack.Install (wifiApNode);
96. stack.Install (wifiStaNodes);

97. Ipv4AddressHelper address;
```

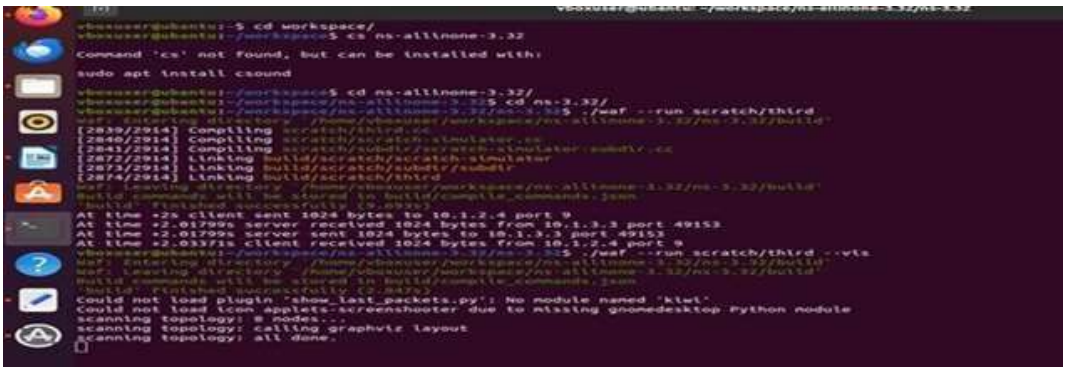
98. address.SetBase ("10.1.1.0", "255.255.255.0");
99. Ipv4InterfaceContainer p2pInterfaces;

100.p2pInterfaces = address.Assign (p2pDevices); 101.address.SetBase ("10.1.2.0", "255.255.255.0");
102.Ipv4InterfaceContainer csmaInterfaces; 103.csmaInterfaces = address.Assign (csmaDevices);

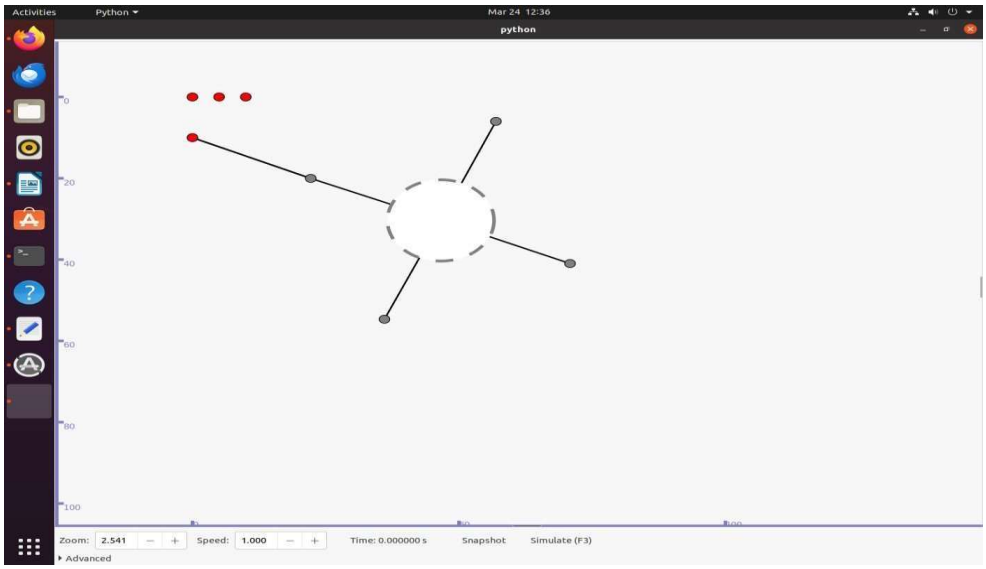
104.address.SetBase ("10.1.3.0", "255.255.255.0");
105.address.Assign (staDevices); 106.address.Assign (apDevices); 107.UdpEchoServerHelper echoServer (9);
108.ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
109.serverApps.Start (Seconds (1.0));
110.serverApps.Stop (Seconds (10.0));
111.UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
112.echoClient.SetAttribute ("MaxPackets", UintegerValue (1)); 113.echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0))); 114.echoClient.SetAttribute ("PacketSize", UintegerValue (1024)); 115.ApplicationContainer clientApps =
116.echoClient.Install (wifiStaNodes.Get (nWifi - 1)); 117.clientApps.Start (Seconds (2.0));
118.clientApps.Stop (Seconds (10.0)); 119.Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
120.Simulator::Stop (Seconds (10.0));
121.if (tracing == true) 122.{
123.pointToPoint.EnablePcapAll ("third"); 124.phy.EnablePcap ("third", apDevices.Get (0));
125.csma.EnablePcap ("third", csmaDevices.Get (0), true); 126. }
127.mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
128.mobility.Install(csmaNodes);
129.AnimationInterface anim("third.xml"); 130.AnimationInterface::SetConstantPosition (csmaNodes.Get(0), 10,25);
131.AnimationInterface::SetConstantPosition (csmaNodes.Get(1), 40,25);
132.anim.EnablePacketMetadata((true));

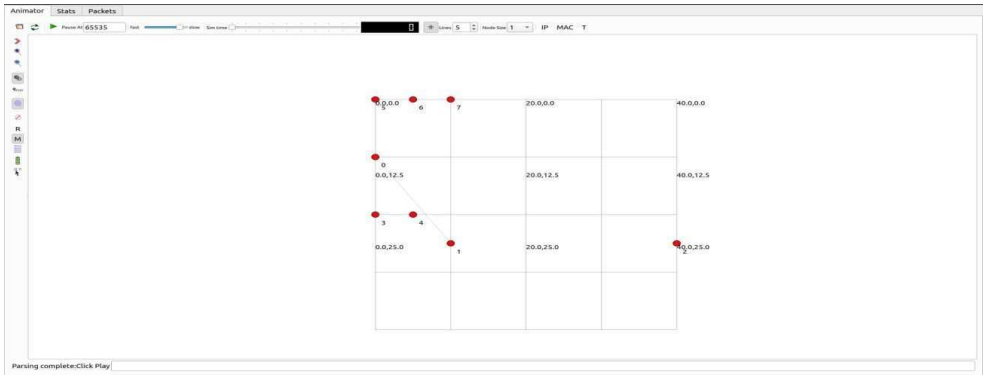
133.Simulator::Run (); 134.Simulator::Destroy (); 135.return 0;
136.}

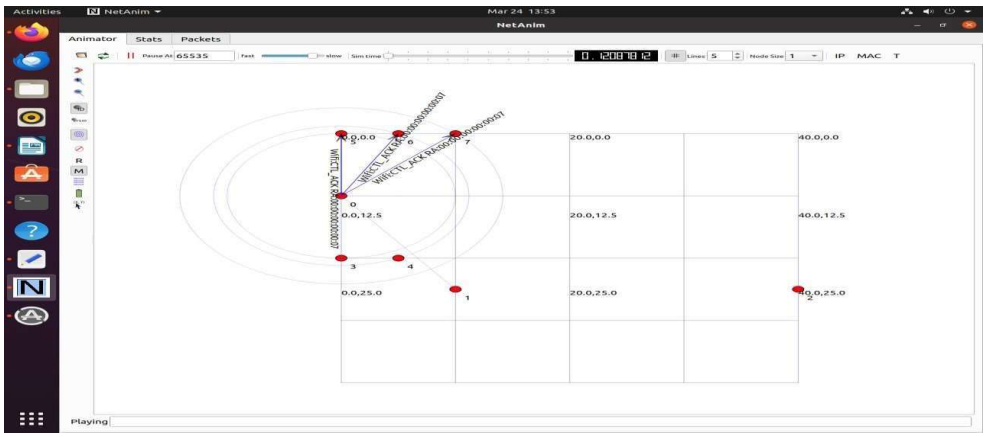
TERMINAL:



OUTPUT:


Activities
Python
Mar 24 12:36
python
Zoom: 2.541
Speed: 1.000
Time: 0.000000 s
Snapshot
Simulate (F3)
Advanced

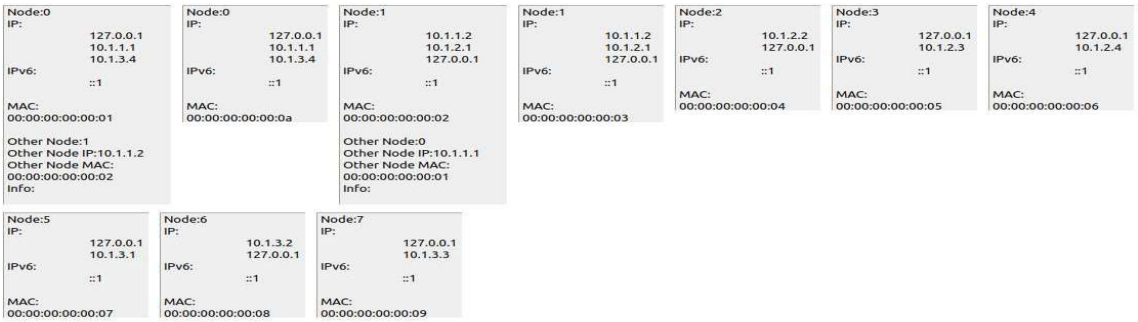

Animator
Stats
Packets
Parsing complete:Click Play


Activities
NetAnim
Mar 24 13:53
NetAnim
Animator
Stats
Packets
Playing

Pratical 8

AIM : Program to simulate Mesh topology.

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
*       Copyright (c) 2008,2009 IITP RAS
*
*       This program is free software; you can redistribute it and/or modify
*       it under the terms of the GNU General Public License version 2 as
*       published by the Free Software Foundation;
*
*       This program is distributed in the hope that it will be useful,
```




```
*       but WITHOUT ANY WARRANTY; without even the implied warranty of * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
*       GNU General Public License for more details.
*
*       You should have received a copy of the GNU General Public License
*       along with this program; if not, write to the Free Software
*       Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*
*       Author: Kirill Andreev <andreev@iitp.ru>
*
*
*       By default this script creates m_xSize * m_ySize square grid topology with * IEEE802.11s stack installed at each node with peering management * and HWMP protocol.
*       The side of the square cell is defined by m_step parameter.
*       When topology is created, UDP ping is installed to opposite corners * by diagonals. packet size of the UDP ping and interval between two * successive packets is configurable.
*
*       m_xSize * step
*       |<          >|
*       step
*       |<--->|
*       * --- * --- * <---Ping sink _
*       | \ | / |     ^
*       | \ | / |     |
```

```
*         * --- *    * m_ySize * step |
*         | / | \ |    |
*         | / | \ |    |
*         * --- * --- *         _
*         ^ Ping source
*
*         See also MeshTest::Configure to read more about configurable * parameters.
*/
#include <iostream> #include <sstream> #include <fstream>
#include "ns3/core-module.h"

#include "ns3/internet-module.h" #include "ns3/network-module.h" #include "ns3/applications-
module.h" #include "ns3/mesh-module.h" #include "ns3/mobility-module.h" #include "ns3/mesh-
helper.h" #include "ns3/yans-wifi-helper.h"

using namespace ns3; NS_LOG_COMPONENT_DEFINE ("TestMeshScript");
class MeshTest
{
public:
MeshTest ();
void Configure (int argc, char ** argv); int Run ();
private:
int                 m_xSize;                    int m_ySize;                    double
                    m_step; double              m_randomStart; double
          m_totalTime; double                   m_packetInterval; uint16_t m_packetSize; uint32_t
m_nIfaces;                  bool m_chan;        bool            m_pcap; bool    m_ascii;std::string
m_stack;                          std::string m_root;
NodeContainer nodes; NetDeviceContainer meshDevices; Ipv4InterfaceContainer interfaces;
MeshHelper mesh; private:         void CreateNodes ();      void InstallInternetStack ();       void
InstallApplication ();
void Report ();
};
MeshTest::MeshTest () : m_xSize (3),       m_ySize
(3),       m_step (100.0),
m_randomStart (0.1),
m_totalTime (100.0),
m_packetInterval (0.1),
m_packetSize (1024),
m_nIfaces (1), m_chan (true), m_pcap (false), m_ascii (false),
m_stack ("ns3::Dot11sStack"), m_root ("ff:ff:ff:ff:ff:ff")

{
}
void
MeshTest::Configure (int argc, char *argv[])
{
CommandLine cmd;
cmd.AddValue ("x-size", "Number of nodes in a row grid", m_xSize); cmd.AddValue ("y-size", "Number
of rows in a grid", m_ySize); cmd.AddValue ("step", "Size of edge in our grid (meters)", m_step); //
Avoid starting all mesh nodes at the same time (beacons may collide)
```

```
cmd.AddValue ("start", "Maximum random start delay for beacon jitter (sec)", m_randomStart);
cmd.AddValue ("time", "Simulation time (sec)", m_totalTime);
cmd.AddValue ("packet-interval", "Interval between packets in UDP ping (sec)", m_packetInterval);
cmd.AddValue ("packet-size", "Size of packets in UDP ping (bytes)", m_packetSize); cmd.AddValue
("interfaces", "Number of radio interfaces used by each mesh point", m_nIfaces); cmd.AddValue
("channels", "Use different frequency channels for different interfaces", m_chan); cmd.AddValue
("pcap", "Enable PCAP traces on interfaces", m_pcap); cmd.AddValue ("ascii", "Enable Ascii traces on
interfaces", m_ascii);
cmd.AddValue ("stack", "Type of protocol stack. ns3::Dot11sStack by default", m_stack);
cmd.AddValue ("root", "Mac address of root mesh point in HWMP", m_root); cmd.Parse (argc, argv);
NS_LOG_DEBUG ("Grid:" << m_xSize << "*" << m_ySize); NS_LOG_DEBUG ("Simulation time: " <<
m_totalTime << " s"); if (m_ascii)
{
PacketMetadata::Enable ();
}
}
void MeshTest::CreateNodes ()
{
/*
* Create m_ySize*m_xSize stations to form a grid topology
*/
nodes.Create (m_ySize*m_xSize);
// Configure YansWifiChannel
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default (); wifiPhy.SetChannel
(wifiChannel.Create ());
/*
*       Create mesh helper and set stack installer to it
*       Stack installer creates all needed protocols and install them to
*       mesh point device
*/
mesh = MeshHelper::Default ();

if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
{
mesh.SetStackInstaller (m_stack, "Root", Mac48AddressValue (Mac48Address (m_root.c_str ())));

} else
{
//If root is not set, we do not use "Root" attribute, because it
//is specified only for 11s mesh.SetStackInstaller (m_stack);
}
if (m_chan)
{
mesh.SetSpreadInterfaceChannels (MeshHelper::SPREAD_CHANNELS);
} else
{
mesh.SetSpreadInterfaceChannels (MeshHelper::ZERO_CHANNEL);
}
mesh.SetMacType ("RandomStart", TimeValue (Seconds (m_randomStart)));
```

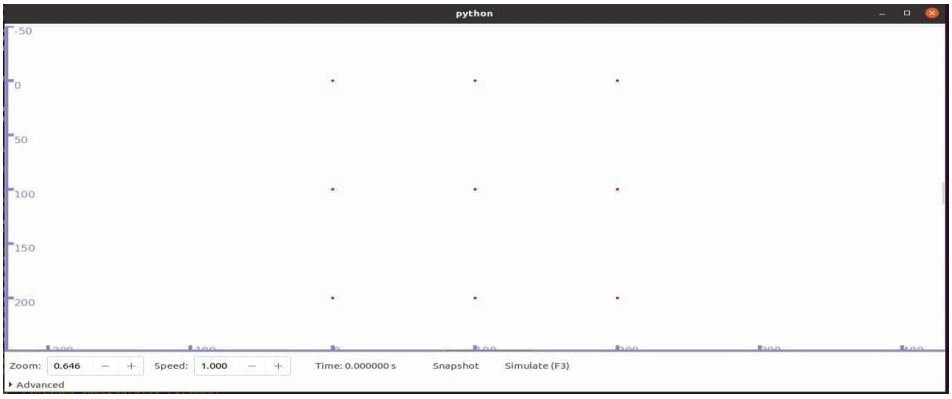
```
// Set number of interfaces - default is single-interface mesh point mesh.SetNumberOfInterfaces
(m_nIfaces);
// Install protocols and return container if MeshPointDevices meshDevices
= mesh.Install (wifiPhy, nodes);
// Setup mobility - static grid topology MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator", "MinX", DoubleValue (0.0),
"MinY", DoubleValue (0.0), "DeltaX", DoubleValue (m_step), "DeltaY", DoubleValue (m_step),
"GridWidth", UintegerValue (m_xSize), "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel"); mobility.Install (nodes);

for (uint32_t i = 0; i < nodes.GetN (); ++i)

if (m_pcap)
wifiPhy.EnablePcapAll (std::string ("mp-")); if (m_ascii)
{
AsciiTraceHelper ascii;
wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("mesh.tr"));
}
}
void MeshTest::InstallInternetStack ()
{
InternetStackHelper internetStack;
internetStack.Install (nodes); Ipv4AddressHelper address; address.SetBase ("10.1.1.0", "255.255.255.0");
interfaces = address.Assign (meshDevices);
}
void MeshTest::InstallApplication ()
{
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (0)); serverApps.Start (Seconds (0.0));
serverApps.Stop (Seconds (m_totalTime));
UdpEchoClientHelper echoClient (interfaces.GetAddress (0), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue ((uint32_t)(m_totalTime*(1/m_packetInterval))));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (m_packetInterval))); echoClient.SetAttribute
("PacketSize", UintegerValue (m_packetSize));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (m_xSize*m_ySize-1)); clientApps.Start
(Seconds (0.0)); clientApps.Stop (Seconds (m_totalTime));
}
{
anim.UpdateNodeDescription (nodes.Get (i), "Node" + std::to_string(i)); // Label the nodes
anim.UpdateNodeColor (nodes.Get (i), 0, 255, 0); // Green color
}
int MeshTest::Run ()
{
CreateNodes (); InstallInternetStack (); InstallApplication ();
Simulator::Schedule (Seconds (m_totalTime), &MeshTest::Report, this); Simulator::Stop (Seconds
(m_totalTime));
Simulator::Run (); Simulator::Destroy (); return 0;
}
void MeshTest::Report ()
{
```

```
unsigned n (0);
for (NetDeviceContainer::Iterator i = meshDevices.Begin (); i != meshDevices.End (); ++i, ++n)
{
std::ostringstream os; os << "mp- report-" << n << ".xml";
std::cerr << "Printing mesh point device #" << n << " diagnostics to " << os.str () << "\n"; std::ofstream
of; of.open (os.str ().c_str ()); if (!of.is_open ())
{
std::cerr << "Error: Can't open file " << os.str () << "\n"; return;
}
mesh.Report (*i, of); of.close ();
} }
int
main (int argc, char *argv[])
{
MeshTest t;
t.Configure (argc, argv); return t.Run ();
}
```

TERMINAL :

```
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/mesh
Waf: Entering directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
[2838/2910] Compiling scratch/mesh.cc
Waf: Leaving directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (5.252s)
Printing mesh point device #0 diagnostics to mp-report-0.xml
Printing mesh point device #1 diagnostics to mp-report-1.xml
Printing mesh point device #2 diagnostics to mp-report-2.xml
Printing mesh point device #3 diagnostics to mp-report-3.xml
Printing mesh point device #4 diagnostics to mp-report-4.xml
Printing mesh point device #5 diagnostics to mp-report-5.xml
Printing mesh point device #6 diagnostics to mp-report-6.xml
Printing mesh point device #7 diagnostics to mp-report-7.xml
Printing mesh point device #8 diagnostics to mp-report-8.xml
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/mesh --vis
Waf: Entering directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.388s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 9 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
```

OUTPUT:



Pratical 09

AIM : Program to simulate UDP server client

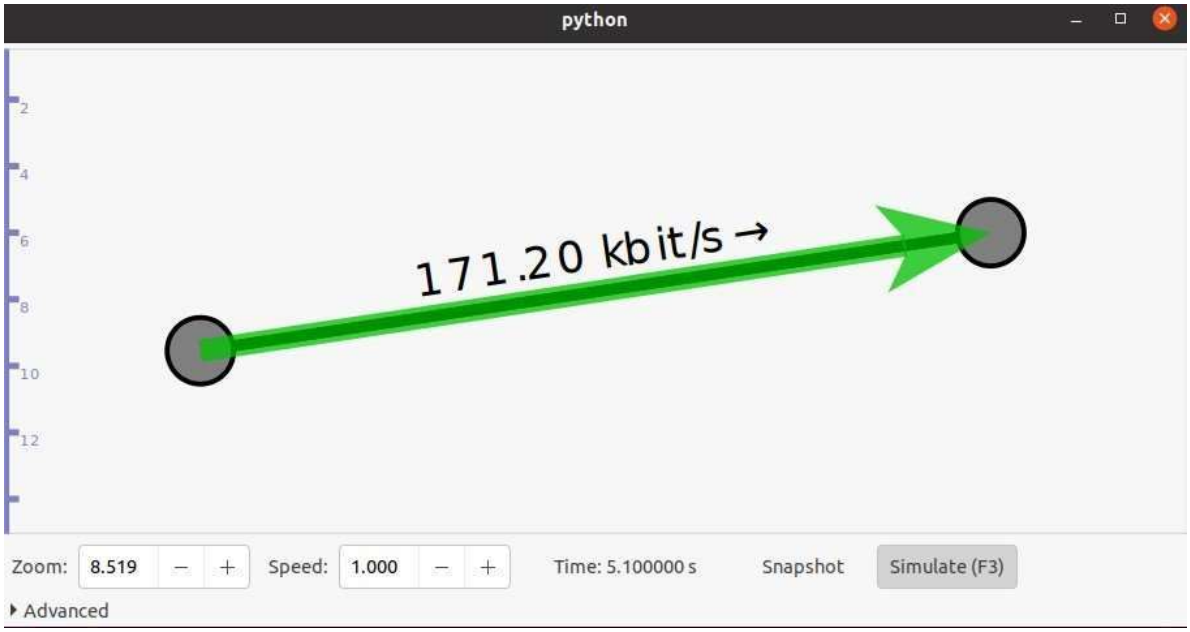
```
// Network topology
//
//       n0     n1
//       |      |
//       =======
//       LAN
//
// - UDP flows from n0 to n1

#include <fstream>
#include "ns3/core-module.h" #include "ns3/csma-module.h"
#include "ns3/applications-module.h" #include "ns3/internet-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("UdpClientServerExample"); int main (int argc, char *argv[])
{
//
// Enable logging for UdpClient and
//
LogComponentEnable ("UdpClient", LOG_LEVEL_INFO); LogComponentEnable ("UdpServer",
LOG_LEVEL_INFO);

bool useV6 = false; Address serverAddress;

CommandLine cmd ( FILE ); cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
cmd.Parse (argc, argv);

//
// Explicitly create the nodes required by the topology (shown above).
//
NS_LOG_INFO ("Create nodes."); NodeContainer n; n.Create (2);

InternetStackHelper internet; internet.Install (n);

NS_LOG_INFO ("Create channels.");
//
// Explicitly create the channels required by the topology (shown above).
//
CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2))); csma.SetDeviceAttribute ("Mtu",
UintegerValue (1400));
NetDeviceContainer d = csma.Install (n);
```

```
//
// We've got the "hardware" in place. Now we need to add IP addresses.
//
NS_LOG_INFO ("Assign IP Addresses."); if (useV6 == false)
{
Ipv4AddressHelper ipv4; ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign (d); serverAddress = Address (i.GetAddress (1));
} else
{
Ipv6AddressHelper ipv6;
ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64)); Ipv6InterfaceContainer i6 = ipv6.Assign (d);
serverAddress = Address(i6.GetAddress (1,1));
}

NS_LOG_INFO ("Create Applications.");
//
// Create one udpServer applications on node one.
//
uint16_t port = 4000; UdpServerHelper server (port);
ApplicationContainer apps = server.Install (n.Get (1)); apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));

//
// Create one UdpClient application to send UDP datagrams from node zero to // node one.
//
uint32_t MaxPacketSize = 1024; Time interPacketInterval = Seconds (0.05); uint32_t maxPacketCount =
320; UdpClientHelper client (serverAddress, port);
client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount)); client.SetAttribute ("Interval",
TimeValue (interPacketInterval)); client.SetAttribute ("PacketSize", UintegerValue (MaxPacketSize));
apps = client.Install (n.Get (0)); apps.Start (Seconds (2.0));
apps.Stop (Seconds (10.0));

//
// Now, do the actual simulation.

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

TERMINAL :

```
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32/scratch$ cd ..
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run udp-client-server --vis
Waf: Entering directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
[2669/2910] Linking build/examples/udp-client-server/ns3.32-udp-client-server-debug
Waf: Leaving directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.525s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 2 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$
```

OUTPUT:



Pratical 10

AIM : Program to simulate DHCP server and clients.

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

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("DhcpExample"); int
main(int argc, char* argv[])
{
CommandLine cmd(  FILE  );

bool verbose = false; bool tracing = false; cmd.AddValue("verbose", "turn on the logs", verbose);
cmd.AddValue("tracing", "turn on the tracing", tracing);

cmd.Parse(argc, argv);
// GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true)); if (verbose)
{
LogComponentEnable("DhcpServer", LOG_LEVEL_ALL); LogComponentEnable("DhcpClient",
LOG_LEVEL_ALL); LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
}

Time stopTime = Seconds(20);

NS_LOG_INFO("Create nodes."); NodeContainer nodes; NodeContainer router; nodes.Create(3);
router.Create(2);

NodeContainer net(nodes, router); NS_LOG_INFO("Create channels.");
CsmaHelper csma; csma.SetChannelAttribute("DataRate",
StringValue("5Mbps")); csma.SetChannelAttribute("Delay",


StringValue("2ms")); csma.SetDeviceAttribute("Mtu", UintegerValue(1500));
NetDeviceContainer devNet = csma.Install(net);

NodeContainer p2pNodes; p2pNodes.Add(net.Get(4)); p2pNodes.Create(1);

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

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

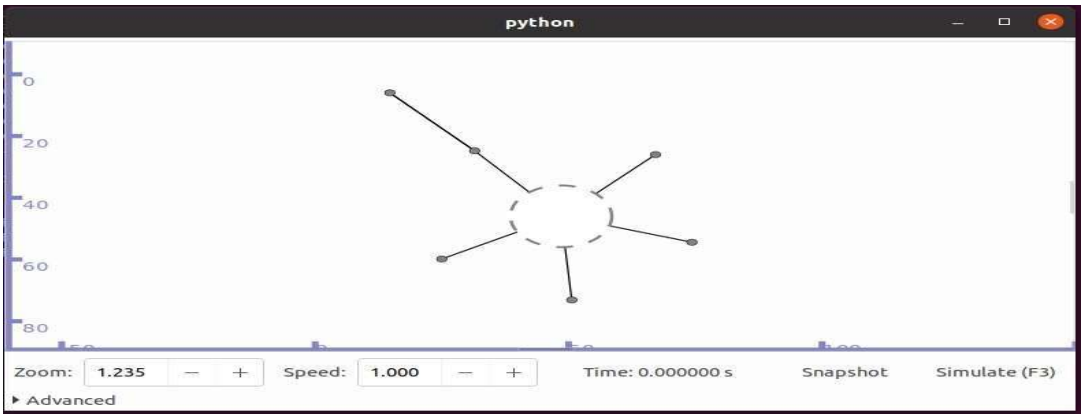
```
InternetStackHelper tcpip; tcpip.Install(nodes); tcpip.Install(router); tcpip.Install(p2pNodes.Get(1));

Ipv4AddressHelper address;
address.SetBase("172.30.1.0", "255.255.255.0"); Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign(p2pDevices);

// manually add a routing entry because we don't want to add a dynamic routing Ipv4StaticRoutingHelper
ipv4RoutingHelper;
Ptr<Ipv4> ipv4Ptr = p2pNodes.Get(1)->GetObject<Ipv4>();
Ptr<Ipv4StaticRouting> staticRoutingA = ipv4RoutingHelper.GetStaticRouting(ipv4Ptr); staticRoutingA-
>AddNetworkRouteTo(Ipv4Address("172.30.0.0"),
Ipv4Mask("/24"), Ipv4Address("172.30.1.1"), 1);

NS_LOG_INFO("Setup the IP addresses and create DHCP applications."); DhcpHelper dhcpHelper;

// The router must have a fixed IP. Ipv4InterfaceContainer fixedNodes =
dhcpHelper.InstallFixedAddress(devNet.Get(4), Ipv4Address("172.30.0.17"), Ipv4Mask("/24")); // Not
really necessary, IP forwarding is enabled by default in IPv4.
fixedNodes.Get(0).first->SetAttribute("IpForward", BooleanValue(true));

// DHCP server
ApplicationContainer dhcpServerApp = dhcpHelper.InstallDhcpServer(devNet.Get(3),
Ipv4Address("172.30.0.12"),
Ipv4Address("172.30.0.0"),
Ipv4Mask("/24"), Ipv4Address("172.30.0.10"), Ipv4Address("172.30.0.15"),
Ipv4Address("172.30.0.17"));

// This is just to show how it can be done. DynamicCast<DhcpServer>(dhcpServerApp.Get(0))
->AddStaticDhcpEntry(devNet.Get(2)->GetAddress(), Ipv4Address("172.30.0.14"));

dhcpServerApp.Start(Seconds(0)); dhcpServerApp.Stop(stopTime);

// DHCP clients
NetDeviceContainer dhcpClientNetDevs; dhcpClientNetDevs.Add(devNet.Get(0));
dhcpClientNetDevs.Add(devNet.Get(1)); dhcpClientNetDevs.Add(devNet.Get(2));

ApplicationContainer dhcpClients = dhcpHelper.InstallDhcpClient(dhcpClientNetDevs);
dhcpClients.Start(Seconds(1));
dhcpClients.Stop(stopTime); UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps = echoServer.Install(p2pNodes.Get(1)); serverApps.Start(Seconds(0));
serverApps.Stop(stopTime);

UdpEchoClientHelper echoClient(p2pInterfaces.GetAddress(1), 9);
echoClient.SetAttribute("MaxPackets", UintegerValue(100)); echoClient.SetAttribute("Interval",
TimeValue(Seconds(1))); echoClient.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps = echoClient.Install(nodes.Get(1)); clientApps.Start(Seconds(10));
clientApps.Stop(stopTime);

Simulator::Stop(stopTime + Seconds(10)); if (tracing)
```

```
{
csma.EnablePcapAll("dhcp-csma"); pointToPoint.EnablePcapAll("dhcp-p2p");
}

NS_LOG_INFO("Run Simulation."); Simulator::Run(); Simulator::Destroy();
NS_LOG_INFO("Done.");

return 0;
}
```

Terminal:

```
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run DHCP-example --vis
Waf: Entering directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
[2669/2912] Linking build/examples/udp-client-server/ns3.32-udp-client-server-debug
[2837/2912] Compiling scratch/DHCP-example.cc
[2838/2912] Compiling scratch/udp-client-server.cc
[2840/2912] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2870/2912] Linking build/scratch/subdir/subdir
[2871/2912] Linking build/scratch/udp-client-server
[2872/2912] Linking build/scratch/DHCP-example
Waf: Leaving directory `/home/ubantu/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (12.306s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 6 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$
```

OUTPUT:



Practical 11

AIM: Program to simulate FTP using TCP.
CODE:
ftp-server.h
#indef FTP_SERVER_H #define FTP_SERVER_H #include “ns3/application.h” #include “ns3/address.h” #include “ns3/ptr.h” #include “ns3/socket.h”
#include “ns3/traced-callback.h” namespace ns3 { class FtpServer :
public Application { public: static TypeId GetTypeId (void); FtpServer (); virtual ~FtpServer ();
protected:
virtual void StartApplication (void); virtual void StopApplication (void); private:
void HandleRead (Ptr<Socket> socket); void SendFile (Ptr<Socket> socket);
Ptr<Socket> m_socket; Address m_local;
};
} // namespace ns3
#endif // FTP_SERVER_H

ftp-server.cc #include “ftp-server.h” #include “ns3/log.h”
#include “ns3/simulator.h”
#include “ns3/inet-socket-address.h” #include “ns3/uinteger.h” namespace ns3 {
NS_LOG_COMPONENT_DEFINE (“FtpServer”); NS_OBJECT_ENSURE_REGISTERED (FtpServer);
TypeId FtpServer::GetTypeId (void) {
static TypeId tid = TypeId (“ns3::FtpServer”) .SetParent<Application> ()
.SetGroupName (“Applications”)
.AddConstructor<FtpServer> ()
.AddAttribute (“Local”, “The Address on which to Bind the rx socket.”, AddressValue (),
MakeAddressAccessor (&FtpServer::m_local),
MakeAddressChecker ());

return tid;
}
FtpServer::FtpServer () { NS_LOG_FUNCTION (this);
}
FtpServer::~FtpServer () { NS_LOG_FUNCTION (this);
}
void FtpServer::StartApplication () { NS_LOG_FUNCTION (this);
if (m_socket == 0) {
m_socket = Socket::CreateSocket (GetNode (), TcpSocketFactory::GetTypeId ()); m_socket->Bind (m_local); m_socket->Listen ();
m_socket->SetRecvCallback (MakeCallback (&FtpServer::HandleRead, this));
}
}
void FtpServer::StopApplication () { NS_LOG_FUNCTION (this);
if (m_socket) { m_socket->Close ();
m_socket->SetRecvCallback (MakeNullCallback<void, Ptr<Socket> > ());
}
}

```
void FtpServer::HandleRead (Ptr<Socket> socket) { NS_LOG_FUNCTION (this << socket);
Ptr<Packet> packet; Address from;
while ((packet = socket->RecvFrom (from))) {
NS_LOG_INFO (“Received request from ” << InetSocketAddress::ConvertFrom (from).GetIpv4 ());
SendFile (socket);
}
}
void FtpServer::SendFile (Ptr<Socket> socket) { NS_LOG_FUNCTION (this << socket);
Ptr<Packet> packet = Create<Packet> (1024); // Simulate sending a 1024-byte file socket->Send (packet);
NS_LOG_INFO (“Sent file to client”);
}
} // namespace ns3

ftp-client.h
#ifndef FTP_CLIENT_H

#define FTP_CLIENT_H #include “ns3/application.h”
#include “ns3/address.h” #include “ns3/ptr.h” #include “ns3/socket.h”
#include “ns3/traced-callback.h” namespace ns3 { class FtpClient :
public Application { public: static TypeId GetTypeId (void); FtpClient (); virtual ~FtpClient (); protected:
virtual void StartApplication (void); virtual void StopApplication (void); private: void SendRequest ();
void HandleRead (Ptr<Socket> socket); Ptr<Socket> m_socket;
Address m_peer; EventId m_sendEvent;
};
} // namespace ns3
#endif // FTP_CLIENT_H

ftp-client.cc #include “ftp-client.h” #include “ns3/log.h”
#include “ns3/simulator.h”
#include “ns3/inet-socket-address.h” #include “ns3/uinteger.h” namespace ns3 {
NS_LOG_COMPONENT_DEFINE (“FtpClient”); NS_OBJECT_ENSURE_REGISTERED (FtpClient);
TypeId FtpClient::GetTypeId (void) {
static TypeId tid = TypeId (“ns3::FtpClient”)
.SetParent<Application> () .SetGroupName (“Applications”)
.AddConstructor<FtpClient> ()
.AddAttribute (“Remote”, “The Address of the FTP server.”, AddressValue (),
MakeAddressAccessor (&FtpClient::m_peer), MakeAddressChecker ());
return tid;
}
FtpClient::FtpClient () { NS_LOG_FUNCTION (this);
}
FtpClient::~FtpClient () {


NS_LOG_FUNCTION (this);
}
void FtpClient::StartApplication () { NS_LOG_FUNCTION (this);
if (m_socket == 0) {
m_socket = Socket::CreateSocket (GetNode (), TcpSocketFactory::GetTypeId ()); m_socket->Connect
(m_peer); m_socket->SetRecvCallback (MakeCallback (&FtpClient::HandleRead, this)); m_sendEvent =
```

```
Simulator::Schedule (Seconds (1.0), &FtpClient::SendRequest, this);
}
}
void FtpClient::StopApplication () { NS_LOG_FUNCTION (this);
if (m_socket) { m_socket->Close ();
}
Simulator::Cancel (m_sendEvent);
}
void FtpClient::SendRequest () { NS_LOG_FUNCTION (this);
Ptr<Packet> packet = Create<Packet> (10); // Simulate sending a request packet m_socket->Send
(packet);
NS_LOG_INFO ("Sent request to server");
}
void FtpClient::HandleRead (Ptr<Socket> socket) { NS_LOG_FUNCTION (this << socket);
Ptr<Packet> packet; Address from;
while ((packet = socket->RecvFrom (from))) {
NS_LOG_INFO ("Received file from " << InetSocketAddress::ConvertFrom (from).GetIpv4 ()); //
Handle the received file here
}
}
} // namespace ns3
```

4. Create FTP Helper Classes

Create helper classes to facilitate the use and testing of FTP. ftp-helper.h #ifndef FTP_HELPER_H #define FTP_HELPER_H #include "ns3/application-container.h" #include "ns3/node-container.h" #include "ns3/ftp-server.h"

```
#include "ns3/ftp-client.h" namespace ns3 { class FtpServerHelper { public:
FtpServerHelper (Address address); ApplicationContainer Install (NodeContainer c); private:
Ptr<Application> InstallPriv (Ptr<Node> node);
Address m_address;
};
class FtpClientHelper { public:
FtpClientHelper (Address address); ApplicationContainer Install (NodeContainer c); private:
Ptr<Application> InstallPriv (Ptr<Node> node);
Address m_address;
};
} // namespace ns3
#endif // FTP_HELPER_H ftp-helper.cc #include
"ftp-helper.h"
#include "ns3/uinteger.h"
#include "ns3/names.h" namespace ns3 {
FtpServerHelper::FtpServerHelper (Address address)
: m_address (address) {
}
ApplicationContainer FtpServerHelper::Install (NodeContainer c) { ApplicationContainer apps;
for (NodeContainer::Iterator i = c.Begin (); i != c.End (); ++i) { Ptr<Node> node = *i;
Ptr<Application> app = InstallPriv (node); apps.Add (app);
}
```

return apps;
}
Ptr<Application> FtpServerHelper::InstallPriv (Ptr<Node> node) { Ptr<FtpServer> server = CreateObject<FtpServer> (); server->SetAttribute (“Local”, AddressValue (m_address)); node->AddApplication (server); return server;
}
FtpClientHelper::FtpClientHelper (Address address)
: m_address (address) {
}
ApplicationContainer FtpClientHelper::Install (NodeContainer c) { ApplicationContainer apps; for (NodeContainer::Iterator i = c.Begin (); i != c.End (); ++i) { Ptr<Node> node = *i;

Ptr<Application> app = InstallPriv (node); apps.Add (app);
}
return apps;
}
Ptr<Application> FtpClientHelper::InstallPriv (Ptr<Node> node) { Ptr<FtpClient> client = CreateObject<FtpClient> (); client->SetAttribute (“Remote”, AddressValue (m_address)); node->AddApplication (client); return client;
}
} // namespace ns3

5. Integrate FTP with ns-3 Build System
Update the wscript file in the src directory to include your new FTP module. src/ftp/wscript def build(bld):
module = bld.create_ns3_module(‘ftp’, [‘core’, ‘network’, ‘internet’, ‘applications’]) module.source = [‘model/ftp-server.cc’,
‘model/ftp-client.cc’, ‘helper/ftp-helper.cc’,
]
headers = bld(features=’ns3header’) headers.module = ‘ftp’ headers.source = [‘model/ftp-server.h’, ‘model/ftp-client.h’, ‘helper/ftp-helper.h’,
]
Build and Test
Rebuild ns-3 to include your new FTP module.
./waf configure
./waf build
Create test scripts to verify the implementation of your FTP protocol. scratch/test-ftp.cc #include “ns3/core-module.h”
#include “ns3/network-module.h” #include “ns3/internet-module.h” #include “ns3/point-to-point-module.h” #include “ns3/ftp-helper.h”
using namespace ns3; int main
(int argc, char *argv[]) { CommandLine cmd; cmd.Parse (argc, argv);
NodeContainer nodes; nodes.Create (2);
PointToPointHelper pointToPoint; pointToPoint.SetDeviceAttribute (“DataRate”, StringValue (“5Mbps”));

pointToPoint.SetChannelAttribute (“Delay”, StringValue (“2ms”)); NetDeviceContainer devices; devices = pointToPoint.Install (nodes);
InternetStackHelper stack; stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase (“10.1.1.0”, “255.255.255.0”);
Ipv4InterfaceContainer interfaces = address.Assign (devices); FtpServerHelper ftpServer (InetSocketAddress (Ipv4Address::GetAny (), 21)); ApplicationContainer serverApps = ftpServer.Install (nodes.Get (1)); serverApps.Start (Seconds (1.0)); serverApps.Stop (Seconds (10.0)); FtpClientHelper ftpClient (InetSocketAddress (interfaces.GetAddress (1), 21)); ApplicationContainer clientApps = ftpClient.Install (nodes.Get (0)); clientApps.Start (Seconds (2.0)); clientApps.Stop (Seconds (10.0)); Simulator::Run ();
Simulator::Destroy (); return 0;
}

TERMINAL:

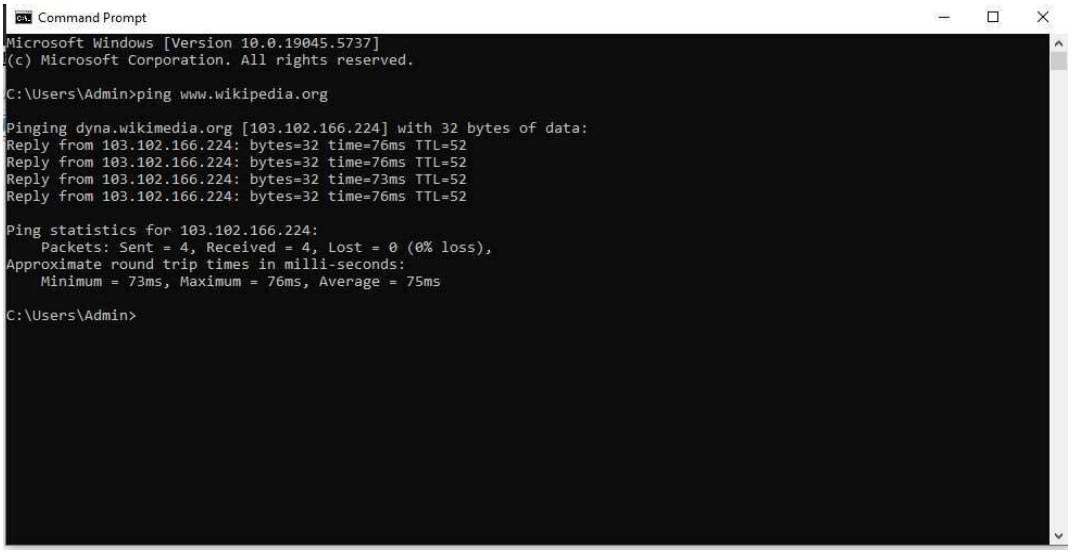
```
admin24@admin24-virtual-machine:~$ cd workspace/
admin24@admin24-virtual-machine:~/workspace$ cd ns-allinone-3.32/
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32$ cd ns-3.32/
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32$ cd src/
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32/src$ mkdir ftp
mkdir: cannot create directory ‘ftp’: File exists
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32/src$ cd ftp
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32/src/ftp$ ls
helper  model  test  wscript
```

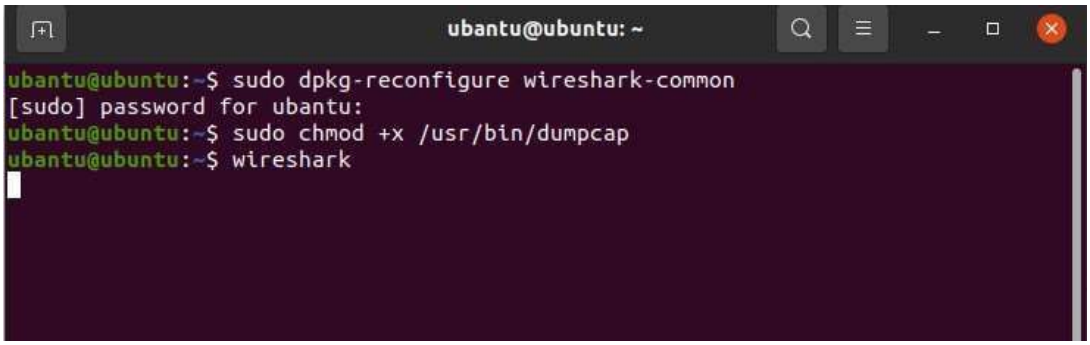
```
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf configure
Setting top to                           : /home/admin24/workspace/ns-allinone-3.32/ns-3.32
Setting out to                           : /home/admin24/workspace/ns-allinone-3.32/ns-3.32/build
Checking for 'gcc' (C compiler)          : /usr/bin/gcc
Checking for cc version                  : 9.4.0
Checking for 'g++' (C++ compiler)        : /usr/bin/g++
Checking for compilation flag -Wl,--soname=foo support : ok
Checking for compilation flag -std=c++11 support       : ok
Checking boost includes                  : headers not found, please provide a --boost-includes argument (see help)
Checking boost includes                  : headers not found, please provide a --boost-includes argument (see help)
Checking for program 'python'            : /usr/bin/python3
```

Pratical 12

AIM : Exercises for analysing the network protocols using Wireshark . Capture the packets while browsing the any web site and analyse the header fields of various protocols.

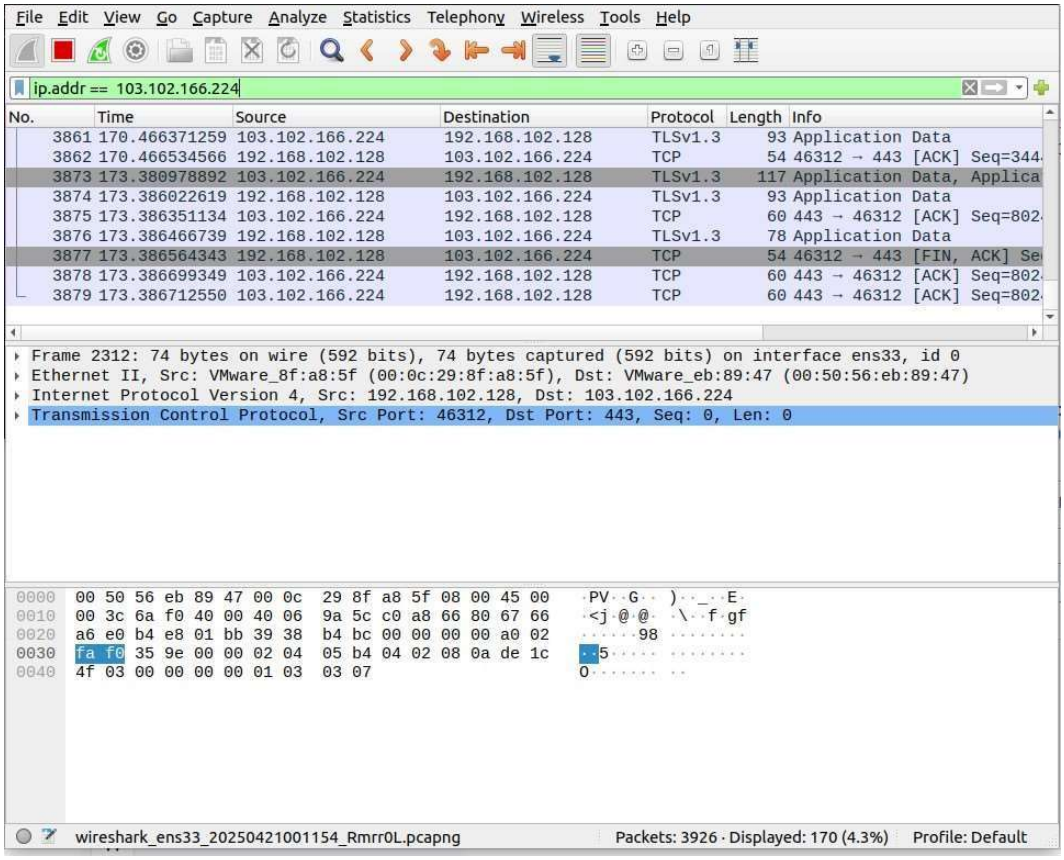
TERMINAL:





OUTPUT:


ip.addr == 103.102.166.224
https://www.wikipedia.org


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ip.addr == 103.102.166.224
No. Time Source Destination Protocol Length Info
3861 170.466371259 103.102.166.224 192.168.102.128 TLSv1.3 93 Application Data
3862 170.466534566 192.168.102.128 103.102.166.224 TCP 54 46312 → 443 [ACK] Seq=344
3873 173.380978892 103.102.166.224 192.168.102.128 TLSv1.3 117 Application Data, Applica
3874 173.386022619 192.168.102.128 103.102.166.224 TLSv1.3 93 Application Data
3875 173.386351134 103.102.166.224 192.168.102.128 TCP 60 443 → 46312 [ACK] Seq=802
3876 173.386466739 192.168.102.128 103.102.166.224 TLSv1.3 78 Application Data
3877 173.386564343 192.168.102.128 103.102.166.224 TCP 54 46312 → 443 [FIN, ACK] Se
3878 173.386699349 103.102.166.224 192.168.102.128 TCP 60 443 → 46312 [ACK] Seq=802
3879 173.386712550 103.102.166.224 192.168.102.128 TCP 60 443 → 46312 [ACK] Seq=802
Frame 2312: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface ens33, id 0
Ethernet II, Src: VMware_8f:a8:5f (00:0c:29:8f:a8:5f), Dst: VMware_eb:89:47 (00:50:56:eb:89:47)
Internet Protocol Version 4, Src: 192.168.102.128, Dst: 103.102.166.224
Transmission Control Protocol, Src Port: 46312, Dst Port: 443, Seq: 0, Len: 0
0000 00 50 56 eb 89 47 00 0c 29 8f a8 5f 08 00 45 00
0010 00 3c 6a f0 40 00 40 06 9a 5c c0 a8 66 80 67 66
0020 a6 e0 b4 e8 01 bb 39 38 b4 bc 00 00 00 00 a0 02
0030 fa f0 35 9e 00 00 02 04 05 b4 04 02 08 0a de 1c
0040 4f 03 00 00 00 00 01 03 03 07
wireshark_ens33_20250421001154_Rmrr0L.pcapng
Packets: 3926 · Displayed: 170 (4.3%)
Profile: Default

Practical 13

AIM: Evaluate the network performance using metrics: throughput, delay, response time, packet loss, dropped packets etc. (Any Topology).

CODE:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */ /*
*               This program is free software; you can redistribute it and/or modify
*               it under the terms of the GNU General Public License version 2 as
*               published by the Free Software Foundation;
*
*               This program is distributed in the hope that it will be useful,
*               but WITHOUT ANY WARRANTY; without even the implied warranty of *
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
*               GNU General Public License for more details.
*
*               You should have received a copy of the GNU General Public License
*               along with this program; if not, write to the Free Software
*               Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/
#include <fstream>
#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/stats-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SeventhScriptExample");

// ===========================================================================
//
//               node 0      node 1
// +----------------+          +          +
// |  ns-3 TCP   |   |  ns-3 TCP         |
// +----------------+          +          +
// |  10.1.1.1     | |  10.1.1.2|
// +----------------+          +          +
// | point-to-point |  | point-to-point |
// +----------------+          +          +
//               |   |
//               +  +
//               5 Mbps, 2 ms
//
//
// We want to look at changes in the ns-3 TCP congestion window. We need
// to crank up a flow and hook the Congestion Window attribute on the socket

// of the sender. Normally one would use an on-off application to generate a
```

```
// flow, but this has a couple of problems. First, the socket of the on-off
// application is not created until Application Start time, so we wouldn't be
// able to hook the socket (now) at configuration time. Second, even if we
// could arrange a call after start time, the socket is not public so we // couldn't get at it.
//
// So, we can cook up a simple version of the on-off application that does what
// we want. On the plus side we don't need all of the complexity of the on-off
// application. On the minus side, we don't have a helper, so we have to get // a little more involved in the details, but this is trivial.
//
// So first, we create a socket and do the trace connect on it; then we pass // this socket into the constructor of our simple application which we then // install in the source node.
//
// NOTE: If this example gets modified, do not forget to update the .png figure
// in src/stats/docs/seventh-packet-byte-count.png
// ===========================================================================
//
class MyApp : public Application
{
public:
MyApp ();
virtual ~MyApp ();

/**
*  Register this type.
*  \return The TypeId.
*/
static TypeId GetTypeId (void);
void Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t nPackets, DataRate dataRate);

private:
virtual void StartApplication (void); virtual void StopApplication (void);
void ScheduleTx (void); void SendPacket (void);
Ptr<Socket>  m_socket; Address  m_peer;
uint32_tm_packetSize;
uint32_tm_nPackets;
DataRate  m_dataRate;
EventId  m_sendEvent; bool  m_running; uint32_t  m_packetsSent;
};

MyApp::MyApp ()
: m_socket (0), m_peer (), m_packetSize (0),
m_nPackets (0),
m_dataRate (0), m_sendEvent (), m_running (false), m_packetsSent (0)
{
}

MyApp::~MyApp ()
{
```

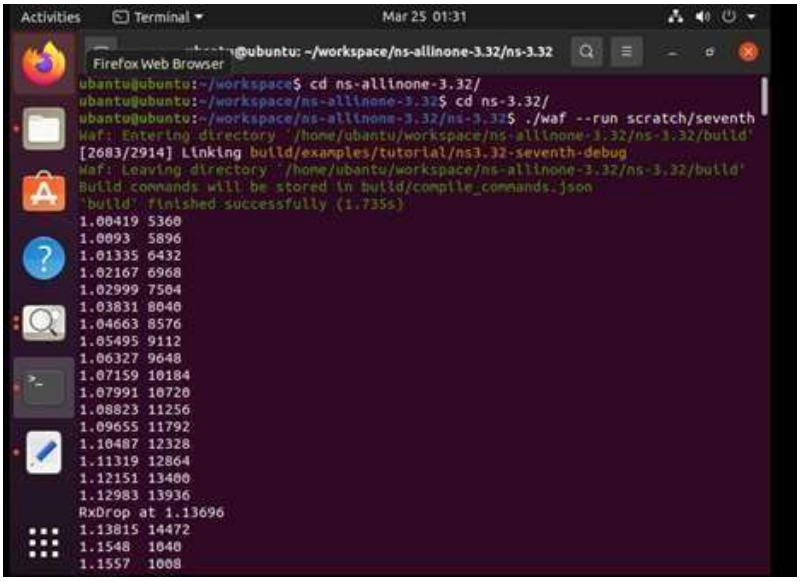
```
m_socket = 0;
}
/* static */
TypeId MyApp::GetTypeId (void)
{
static TypeId tid = TypeId ("MyApp")
.SetParent<Application> ()
.SetGroupName ("Tutorial")
.AddConstructor<MyApp> ()
;
return tid;
}
void
MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t nPackets, DataRate
dataRate)
{
m_socket = socket; m_peer
= address; m_packetSize = packetSize; m_nPackets = nPackets;
m_dataRate = dataRate;
}
void
MyApp::StartApplication (void)
{
m_running = true; m_packetsSent
= 0;
if (InetSocketAddress::IsMatchingType (m_peer))
{
m_socket->Bind ();
} else
{
m_socket->Bind6 ();

m_socket->Connect (m_peer);

SendPacket ();
}
void
MyApp::StopApplication (void)
{
m_running = false;

if (m_sendEvent.IsRunning ())
{
Simulator::Cancel (m_sendEvent);
}
if (m_socket)
{
m_socket->Close ();
}
}
```

```
void
MyApp::SendPacket (void)
{
Ptr<Packet> packet = Create<Packet> (m_packetSize); m_socket->Send (packet);

if (++m_packetsSent < m_nPackets)
{
ScheduleTx ();
}
}
void
MyApp::ScheduleTx (void)
{
if (m_running)
{
Time tNext (Seconds (m_packetSize * 8 / static_cast<double> (m_dataRate.GetBitRate ()))); m_sendEvent
= Simulator::Schedule (tNext, &MyApp::SendPacket, this);
}
}
static void
CwndChange (Ptr<OutputStreamWrapper> stream, uint32_t oldCwnd, uint32_t newCwnd)
{
NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << "\t" << newCwnd);
*stream->GetStream () << Simulator::Now ().GetSeconds () << "\t" << oldCwnd << "\t" << newCwnd <<
std::endl;

static void
RxDrop (Ptr<PcapFileWrapper> file, Ptr<const Packet> p)
{
NS_LOG_UNCOND ("RxDrop at " << Simulator::Now ().GetSeconds ()); file->Write (Simulator::Now (),
p);
}
int
main (int argc, char *argv[])
{
bool useV6 = false;

CommandLine cmd ( FILE ); cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
cmd.Parse (argc, argv);

NodeContainer nodes; nodes.Create (2);

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

NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);

Ptr<RateErrorModel> em = CreateObject<RateErrorModel> (); em->SetAttribute ("ErrorRate",
DoubleValue (0.00001));
```

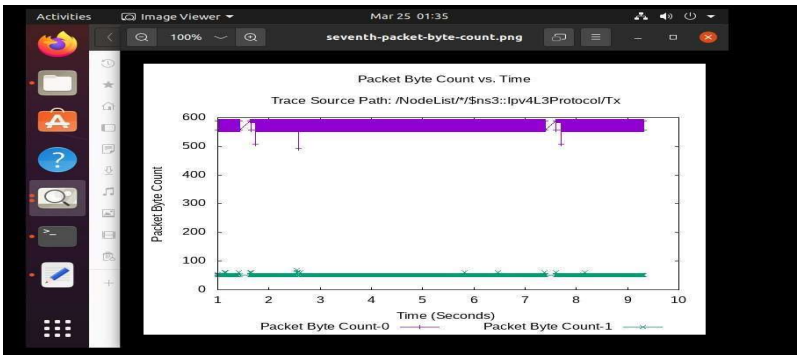
```
devices.Get (1)->SetAttribute ("ReceiveErrorModel", PointerValue (em));

InternetStackHelper stack; stack.Install (nodes);

uint16_t sinkPort = 8080; Address sinkAddress; Address anyAddress; std::string probeType; std::string
tracePath;
if (useV6 == false)
{
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices); sinkAddress = InetSocketAddress
(interfaces.GetAddress (1), sinkPort); anyAddress = InetSocketAddress (Ipv4Address::GetAny (),
sinkPort); probeType = "ns3::Ipv4PacketProbe"; tracePath = "/NodeList/*/$ns3::Ipv4L3Protocol/Tx";
} else
{
Ipv6AddressHelper address; address.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));
Ipv6InterfaceContainer interfaces = address.Assign (devices); sinkAddress = Inet6SocketAddress
(interfaces.GetAddress (1,1), sinkPort); anyAddress = Inet6SocketAddress (Ipv6Address::GetAny (),
sinkPort); probeType = "ns3::Ipv6PacketProbe";
tracePath = "/NodeList/*/$ns3::Ipv6L3Protocol/Tx";
}
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", anyAddress); ApplicationContainer
sinkApps = packetSinkHelper.Install (nodes.Get (1)); sinkApps.Start (Seconds (0.));
sinkApps.Stop (Seconds (20.));
Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (nodes.Get (0), TcpSocketFactory::GetTypeId ());
Ptr<MyApp> app = CreateObject<MyApp> ();
app->Setup (ns3TcpSocket, sinkAddress, 1040, 1000, DataRate ("1Mbps"));
nodes.Get (0)->AddApplication (app); app->SetStartTime (Seconds (1.)); app->SetStopTime (Seconds
(20.));

AsciiTraceHelper asciiTraceHelper;
Ptr<OutputStreamWrapper> stream = asciiTraceHelper.CreateFileStream ("seventh.cwnd");
ns3TcpSocket-
>TraceConnectWithoutContext ("CongestionWindow", MakeBoundCallback (&CwndChange, stream));

PcapHelper pcapHelper;
Ptr<PcapFileWrapper> file = pcapHelper.CreateFile ("seventh.pcap", std::ios::out,
PcapHelper::DLT_PPP); devices.Get (1)->TraceConnectWithoutContext ("PhyRxDrop",
MakeBoundCallback (&RxDrop, file));

// Use GnuplotHelper to plot the packet byte count over time GnuplotHelper plotHelper;
// Configure the plot. The first argument is the file name prefix
// for the output files generated. The second, third, and fourth // arguments are, respectively, the plot title,
x-axis, and y-axis labels plotHelper.ConfigurePlot ("seventh-packet-byte-count",
"Packet Byte Count vs. Time", "Time (Seconds)",
"Packet Byte Count");

// Specify the probe type, trace source path (in configuration namespace), and
// probe output trace source ("OutputBytes") to plot. The fourth argument
// specifies the name of the data series label on the plot. The last
```

```
// argument formats the plot by specifying where the key should be placed. plotHelper.PlotProbe
(probeType,
tracePath, "OutputBytes", "Packet Byte Count",
GnuplotAggregator::KEY_BELOW);

// Use FileHelper to write out the packet byte count over time FileHelper fileHelper;

// Configure the file to be written, and the formatting of output data. fileHelper.ConfigureFile ("seventh-
packet-byte-count", FileAggregator::FORMATTED);

// Set the labels for this formatted output file.
fileHelper.Set2dFormat ("Time (Seconds) = %.3e\tPacket Byte Count = %.0f");

// Specify the probe type, trace source path (in configuration namespace), and
// probe output trace source ("OutputBytes") to write. fileHelper.WriteProbe (probeType,
tracePath, "OutputBytes");
Simulator::Stop (Seconds (20)); Simulator::Run (); Simulator::Destroy ();

return 0;
}
```

TERMINAL:



OUTPUT:

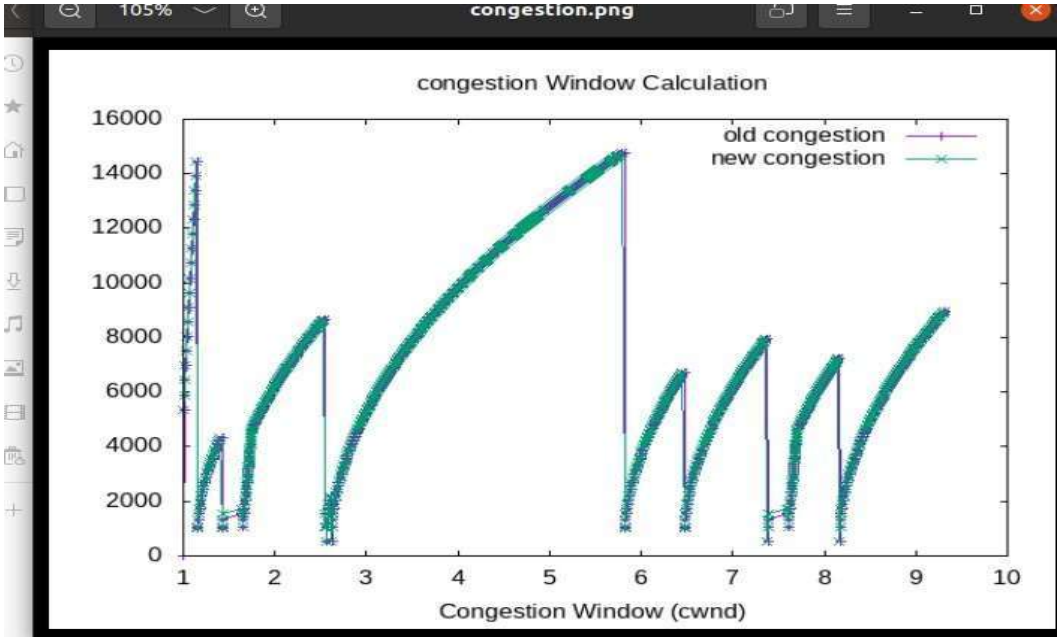


SAVE THE FILE AS CONGESTION PNG IN THE NS-3.32 FOLDER.

```
1 set terminal png
2 set output "congestion.png"
3 set title "congestion Window Calculation"
4 set xlabel "Congestion Window (cwnd)"
5 plot "seventh.cwnd" using 1:2 with linespoints title "old congesion", "seventh.cwnd" using 1:3 with linespoints title "new congesion"
```

Terminal:

```
ubantu@ubuntu:~/workspace/ns-allinone-3.32/ns-3.32$ gnuplot congestion.plt
```



PROJECT

AIM : To simulate and analyze the behavior of dynamic global routing in a mixed point-to-point and CSMA network topology using NS-3, including automatic rerouting of traffic based on interface events, and visualize network act.

dynamic-global-routing.cc file

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */ /*
*       This program is free software; you can redistribute it and/or modify
*       it under the terms of the GNU General Public License version 2 as
*       published by the Free Software Foundation;
*
*       This program is distributed in the hope that it will be useful,
*       but WITHOUT ANY WARRANTY; without even the implied warranty of * MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. See the
*       GNU General Public License for more details.
*
*       You should have received a copy of the GNU General Public License
*       along with this program; if not, write to the Free Software
*       Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*
*       Contributed by: Luis Cortes (cortes@gatech.edu)
*/


// This script exercises global routing code in a mixed point-to-point
// and csma/cd environment. We bring up and down interfaces and observe
// the effect on global routing. We explicitly enable the attribute
// to respond to interface events, so that routes are recomputed // automatically.
//
// Network topology
//
// n0
//       \ p-p
//       \       (shared csma/cd)
//       n2      n3
//       /       |       |
//       / p-p   n4      n5      n6
// n1    p-p
// |     |
//       //
p-p
//
// - at time 1 CBR/UDP flow from n1 to n6's IP address on the n5/n6 link
// - at time 10, start similar flow from n1 to n6's address on the n1/n6 link
//
```

```
// Order of events

// At pre-simulation time, configure global routes. Shortest path from
// n1 to n6 is via the direct point-to-point link
// At time 1s, start CBR traffic flow from n1 to n6
// At time 2s, set the n1 point-to-point interface to down. Packets
//         will be diverted to the n1-n2-n5-n6 path
// At time 4s, re-enable the n1/n6 interface to up. n1-n6 route restored.
// At time 6s, set the n6-n1 point-to-point Ipv4 interface to down (note, this
//         keeps the point-to-point link "up" from n1's perspective). Traffic will
//         flow through the path n1-n2-n5-n6
// At time 8s, bring the interface back up. Path n1-n6 is restored // At time 10s, stop the first flow.
// At time 11s, start a new flow, but to n6's other IP address (the one
//         on the n1/n6 p2p link)
// At time 12s, bring the n1 interface down between n1 and n6. Packets
//         will be diverted to the alternate path
// At time 14s, re-enable the n1/n6 interface to up. This will change
//         routing back to n1-n6 since the interface up notification will cause
//         a new local interface route, at higher priority than global routing // At time 16s, stop the second flow.
// - Tracing of queues and packet receptions to file "dynamic-global-routing.tr" #include <iostream>
#include <fstream>
#include <string> #include <cassert>

#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" // <<=== ADD THIS

using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("DynamicGlobalRoutingExample"); int
main (int argc, char *argv[])
{
// The below value configures the default behavior of global routing.
// By default, it is disabled. To respond to interface events, set to true
Config::SetDefault ("ns3::Ipv4GlobalRouting::RespondToInterfaceEvents", BooleanValue (true));


// Allow the user to override any of the defaults and the above
// Bind ()s at run-time, via command-line arguments CommandLine cmd ( FILE );


cmd.Parse (argc, argv);

NS_LOG_INFO ("Create nodes."); NodeContainer c; c.Create (7);
NodeContainer n0n2 = NodeContainer (c.Get (0), c.Get (2)); NodeContainer n1n2 = NodeContainer (c.Get (1),
c.Get (2)); NodeContainer n5n6 = NodeContainer (c.Get (5), c.Get (6)); NodeContainer n1n6 = NodeContainer
(c.Get (1), c.Get (6));
NodeContainer n2345 = NodeContainer (c.Get (2), c.Get (3), c.Get (4), c.Get (5));
```

```
InternetStackHelper internet; internet.Install (c);

// 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"));
NetDeviceContainer d0d2 = p2p.Install (n0n2); NetDeviceContainer d1d6 = p2p.Install (n1n6);

NetDeviceContainer d1d2 = p2p.Install (n1n2);

p2p.SetDeviceAttribute ("DataRate", StringValue ("1500kbps")); p2p.SetChannelAttribute ("Delay", StringValue ("10ms")); NetDeviceContainer d5d6 = p2p.Install (n5n6);

// We create the channels first without any IP addressing information CsmaHelper csma; csma.SetChannelAttribute ("DataRate", StringValue ("5Mbps")); csma.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer d2345 = csma.Install (n2345);

// Later, we add IP addresses. NS_LOG_INFO ("Assign IP Addresses."); Ipv4AddressHelper ipv4; ipv4.SetBase ("10.1.1.0", "255.255.255.0"); ipv4.Assign (d0d2);

ipv4.SetBase ("10.1.2.0", "255.255.255.0");
ipv4.Assign (d1d2);

ipv4.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer i5i6 = ipv4.Assign (d5d6);


ipv4.SetBase ("10.250.1.0", "255.255.255.0");
ipv4.Assign (d2345);

ipv4.SetBase ("172.16.1.0", "255.255.255.0");
Ipv4InterfaceContainer i1i6 = ipv4.Assign (d1d6);

// Create router nodes, initialize routing database and set up the routing
// tables in the nodes. Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

// Create the OnOff application to send UDP datagrams of size
// 210 bytes at a rate of 448 Kb/s NS_LOG_INFO ("Create Applications."); uint16_t port = 9; // Discard port (RFC 863) OnOffHelper onoff ("ns3::UdpSocketFactory",
InetSocketAddress (i5i6.GetAddress (1), port)); onoff.SetConstantRate (DataRate ("2kbps")); onoff.SetAttribute ("PacketSize", UintegerValue (50));

ApplicationContainer apps = onoff.Install (c.Get (1)); apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));

// Create a second OnOff application to send UDP datagrams of size
// 210 bytes at a rate of 448 Kb/s
OnOffHelper onoff2 ("ns3::UdpSocketFactory", InetSocketAddress (i1i6.GetAddress (1), port));
```

onoff2.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
onoff2.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));
onoff2.SetAttribute ("DataRate", StringValue ("2kbps")); onoff2.SetAttribute ("PacketSize", UintegerValue (50));

ApplicationContainer apps2 = onoff2.Install (c.Get (1)); apps2.Start (Seconds (11.0));
apps2.Stop (Seconds (16.0));

// Create an optional packet sink to receive these packets PacketSinkHelper sink ("ns3::UdpSocketFactory", Address (InetSocketAddress (Ipv4Address::GetAny (), port))); apps = sink.Install (c.Get (6)); apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));

PacketSinkHelper sink2 ("ns3::UdpSocketFactory",
Address (InetSocketAddress (Ipv4Address::GetAny (), port))); apps2 = sink2.Install (c.Get (6)); apps2.Start (Seconds (11.0));
apps2.Stop (Seconds (16.0));

AsciiTraceHelper ascii;

Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("dynamic-global-routing.tr"); p2p.EnableAsciiAll (stream); csma.EnableAsciiAll (stream);
internet.EnableAsciiIpv4All (stream); p2p.EnablePcapAll ("dynamic-global-routing");
csma.EnablePcapAll ("dynamic-global-routing", false); Ptr<Node> n1 = c.Get (1);
Ptr<Ipv4> ipv41 = n1->GetObject<Ipv4> ();
// The first ifIndex is 0 for loopback, then the first p2p is numbered 1,
// then the next p2p is numbered 2 uint32_t ipv4ifIndex1 = 2;

Simulator::Schedule (Seconds (2),&Ipv4::SetDown,ipv41, ipv4ifIndex1); Simulator::Schedule (Seconds (4),&Ipv4::SetUp,ipv41, ipv4ifIndex1);

Ptr<Node> n6 = c.Get (6);
Ptr<Ipv4> ipv46 = n6->GetObject<Ipv4> ();
// The first ifIndex is 0 for loopback, then the first p2p is numbered 1,
// then the next p2p is numbered 2 uint32_t ipv4ifIndex6 = 2;
Simulator::Schedule (Seconds (6),&Ipv4::SetDown,ipv46, ipv4ifIndex6); Simulator::Schedule (Seconds (8),&Ipv4::SetUp,ipv46, ipv4ifIndex6);

Simulator::Schedule (Seconds (12),&Ipv4::SetDown,ipv41, ipv4ifIndex1); Simulator::Schedule (Seconds (14),&Ipv4::SetUp,ipv41, ipv4ifIndex1);

// Trace routing tables Ipv4GlobalRoutingHelper g;
Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ("dynamic- globalrouting.routes", std::ios::out);
g.PrintRoutingTableAllAt (Seconds (12), routingStream);

// --- Add Animation Code ---

```
AnimationInterface anim ("dynamic-global-routing.xml");

// Optional: Set fixed positions for better visualization in NetAnim anim.SetConstantPosition(c.Get(0), 10, 30);
anim.SetConstantPosition(c.Get(1),
10, 10); anim.SetConstantPosition(c.Get(2), 30, 20);
anim.SetConstantPosition(c.Get(3), 50, 30); anim.SetConstantPosition(c.Get(4),
50, 10); anim.SetConstantPosition(c.Get(5), 70, 20);
anim.SetConstantPosition(c.Get(6), 90, 20);
// --- End Animation Code ---

NS_LOG_INFO ("Run Simulation."); Simulator::Run (); Simulator::Destroy ();


NS_LOG_INFO ("Done.");
      }
```

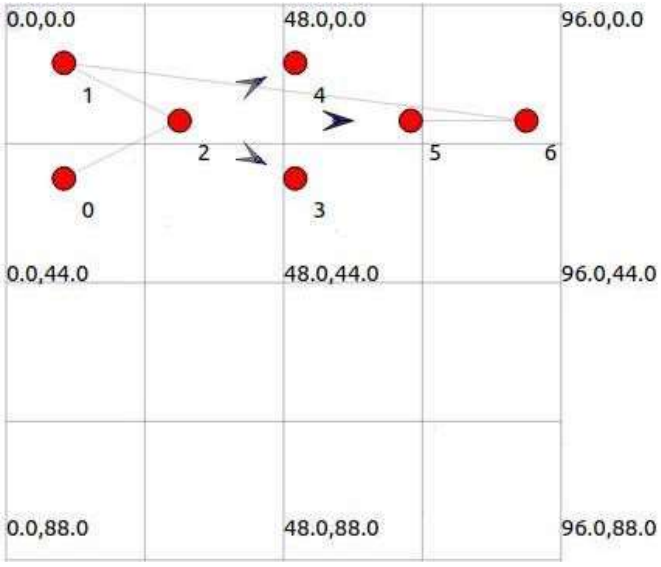
TERMINAL:

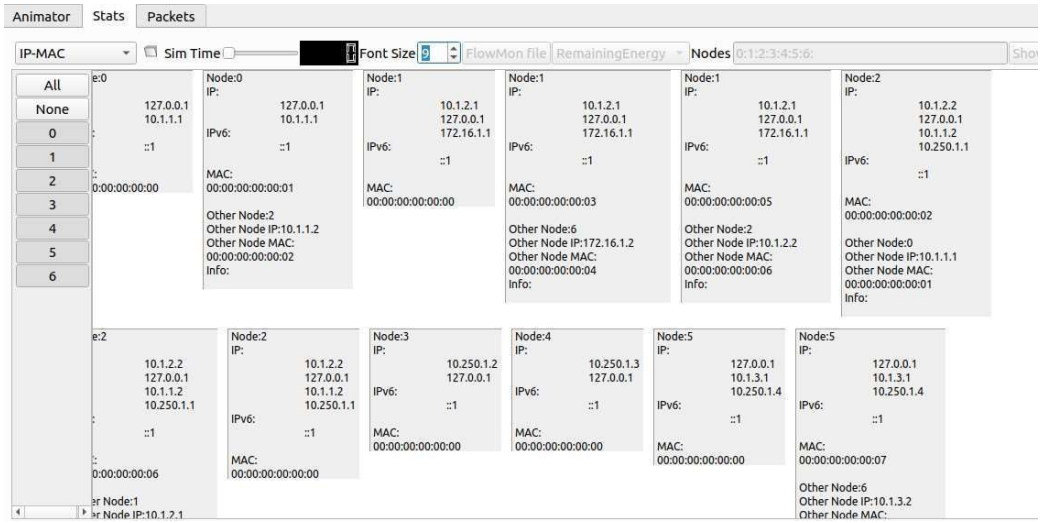
```
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/dynamic-global-routing
Waf: Entering directory `/home/admin24/workspace/ns-allinone-3.32/ns-3.32/build'
[1874/1914] Compiling scratch/dynamic-global-routing.cc
[1875/1914] Linking build/scratch/dynamic-global-routing
Waf: Leaving directory `/home/admin24/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (15.984s)
```

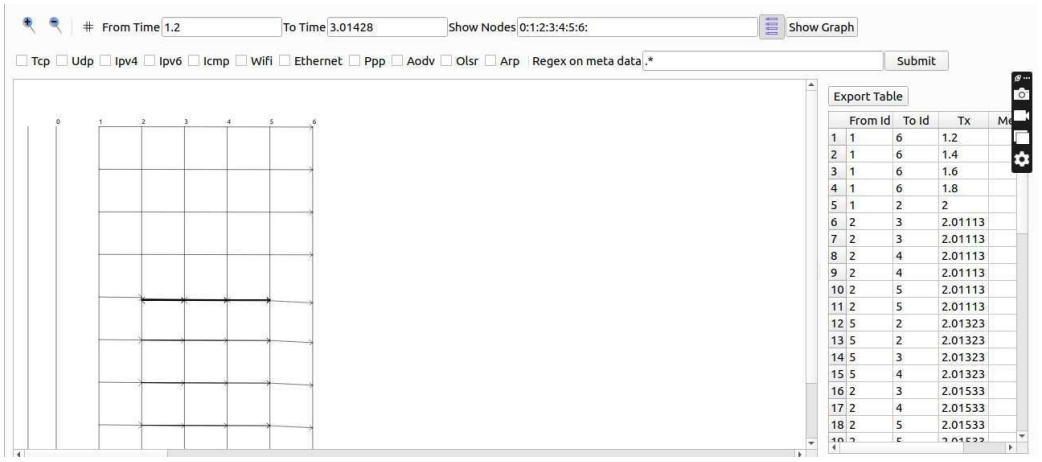
```
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/ns-3.32$ cd ..
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32$ cd netanim-3.108/
admin24@admin24-virtual-machine:~/workspace/ns-allinone-3.32/netanim-3.108$ ./NetAnim
```

OUTPUT:

GRAPGH




Animator
Stats
Packets
IP-MAC
Sim Time
Font Size
FlowMon file
RemainingEnergy
Nodes
All
None
0
1
2
3
4
5
6
Node:0
IP:
127.0.0.1
10.1.1.1
IPv6:
::1
MAC:
00:00:00:00:00:01
Other Node:2
Other Node IP:10.1.1.2
Other Node MAC:
00:00:00:00:00:02
Info:
Node:1
IP:
10.1.2.1
127.0.0.1
172.16.1.1
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:1
IP:
10.1.2.1
127.0.0.1
172.16.1.1
IPv6:
::1
MAC:
00:00:00:00:00:03
Other Node:6
Other Node IP:172.16.1.2
Other Node MAC:
00:00:00:00:00:04
Info:
Node:1
IP:
10.1.2.1
127.0.0.1
172.16.1.1
IPv6:
::1
MAC:
00:00:00:00:00:05
Other Node:2
Other Node IP:10.1.2.2
Other Node MAC:
00:00:00:00:00:06
Info:
Node:2
IP:
10.1.2.2
127.0.0.1
10.1.1.2
10.250.1.1
IPv6:
::1
MAC:
00:00:00:00:00:02
Other Node:0
Other Node IP:10.1.1.1
Other Node MAC:
00:00:00:00:00:01
Info:
Node:2
IP:
10.1.2.2
127.0.0.1
10.1.1.2
10.250.1.1
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:3
IP:
10.250.1.2
127.0.0.1
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:4
IP:
10.250.1.3
127.0.0.1
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:5
IP:
127.0.0.1
10.1.3.1
10.250.1.4
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:5
IP:
127.0.0.1
10.1.3.1
10.250.1.4
IPv6:
::1
MAC:
00:00:00:00:00:07
Other Node:6
Other Node IP:10.1.3.2


From Time 1.2
To Time 3.01428
Show Nodes 0:1:2:3:4:5:6:
Show Graph
Tcp
Udp
Ipv4
Ipv6
Icmp
Wifi
Ethernet
Ppp
Aodv
Olsr
Arp
Regex on meta data .*
Submit
Export Table
From Id
To Id
Tx
1 1 6 1.2
2 1 6 1.4
3 1 6 1.6
4 1 6 1.8
5 1 2 2
6 2 3 2.01113
7 2 3 2.01113
8 2 4 2.01113
9 2 4 2.01113
10 2 5 2.01113
11 2 5 2.01113
12 5 2 2.01323
13 5 2 2.01323
14 5 3 2.01323
15 5 4 2.01323
16 2 3 2.01533
17 2 4 2.01533
18 2 5 2.01533