

## NS LAB 4 SEPTEMBER | Lab Assignment 4

Name: Maloth Aditya  
Roll No: 120CS0124

### Question:

1. Set constant position for point-to-point nodes and as well as CSMA nodes.
2. Plot a graph by using Gnuplot via, .tr, considering throughput and goodput as parameters for your graph generation.
3. Display the result in NetAnim, Wireshark and TraceMetrics.

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

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

## NS LAB 4 SEPTEMBER | Lab Assignment 4

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");

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

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

    cmd.Parse (argc,argv);

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

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

    NodeContainer p2pNodes;
    p2pNodes.Create (2);

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

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

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

## NS LAB 4 SEPTEMBER | Lab Assignment 4

```
CsmaHelper csma;  
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));  
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));  
  
NetDeviceContainer csmaDevices;  
csmaDevices = csma.Install (csmaNodes);  
  
NodeContainer wifiStaNodes;  
wifiStaNodes.Create (nWifi);  
NodeContainer wifiApNode = p2pNodes.Get (0);  
  
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();  
YansWifiPhyHelper phy;  
phy.SetChannel (channel.Create ());  
  
WifiHelper wifi;  
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");  
  
WifiMacHelper mac;  
Ssid ssid = Ssid ("ns-3-ssid");  
mac.SetType ("ns3::StaWifiMac",  
             "Ssid", SsidValue (ssid),  
             "ActiveProbing", BooleanValue (false));  
  
NetDeviceContainer staDevices;  
staDevices = wifi.Install (phy, mac, wifiStaNodes);  
  
mac.SetType ("ns3::ApWifiMac",  
             "Ssid", SsidValue (ssid));  
  
NetDeviceContainer apDevices;  
apDevices = wifi.Install (phy, mac, wifiApNode);  
  
MobilityHelper mobility;  
  
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",  
                               "MinX", DoubleValue (0.0),  
                               "MinY", DoubleValue (0.0),  
                               "DeltaX", DoubleValue (5.0),  
                               "DeltaY", DoubleValue (10.0),  
                               "GridWidth", UIntegerValue (3),  
                               "LayoutType", StringValue ("RowFirst"));  
  
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",  
                           "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));  
mobility.Install (wifiStaNodes);  
  
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
```

## NS LAB 4 SEPTEMBER | Lab Assignment 4

```
mobility.Install (wifiApNode);
```

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

```
Ipv4AddressHelper address;
```

```
address.SetBase ("10.1.1.0", "255.255.255.0");  
Ipv4InterfaceContainer p2pInterfaces;  
p2pInterfaces = address.Assign (p2pDevices);
```

```
address.SetBase ("10.1.2.0", "255.255.255.0");  
Ipv4InterfaceContainer csmaInterfaces;  
csmaInterfaces = address.Assign (csmaDevices);
```

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

```
UdpEchoServerHelper echoServer (9);
```

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

```
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);  
echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));  
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));  
echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));
```

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

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

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

```
//Configure node positions  
AnimationInterface anim("lab4.xml");  
anim.SetConstantPosition(p2pNodes.Get(0), 12.0,12.0);  
anim.SetConstantPosition(p2pNodes.Get(1), 24.0,12.0);  
anim.SetConstantPosition(csmaNodes.Get(0), 24.0,24.0);  
anim.SetConstantPosition(csmaNodes.Get(1), 12.0,24.0);  
anim.SetConstantPosition(csmaNodes.Get(2), 12.0, 36.0);
```

## NS LAB 4 SEPTEMBER | Lab Assignment 4

```
anim.SetConstantPosition(csmaNodes.Get(3), 24.0,36.0);
```

```
AsciiTraceHelper ascii;  
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("lab4-point-to-point.tr"));  
csma.EnableAsciiAll(ascii.CreateFileStream("lab4-csma.tr"));
```

```
csma.EnablePcapAll("lab4-pcap",false);
```

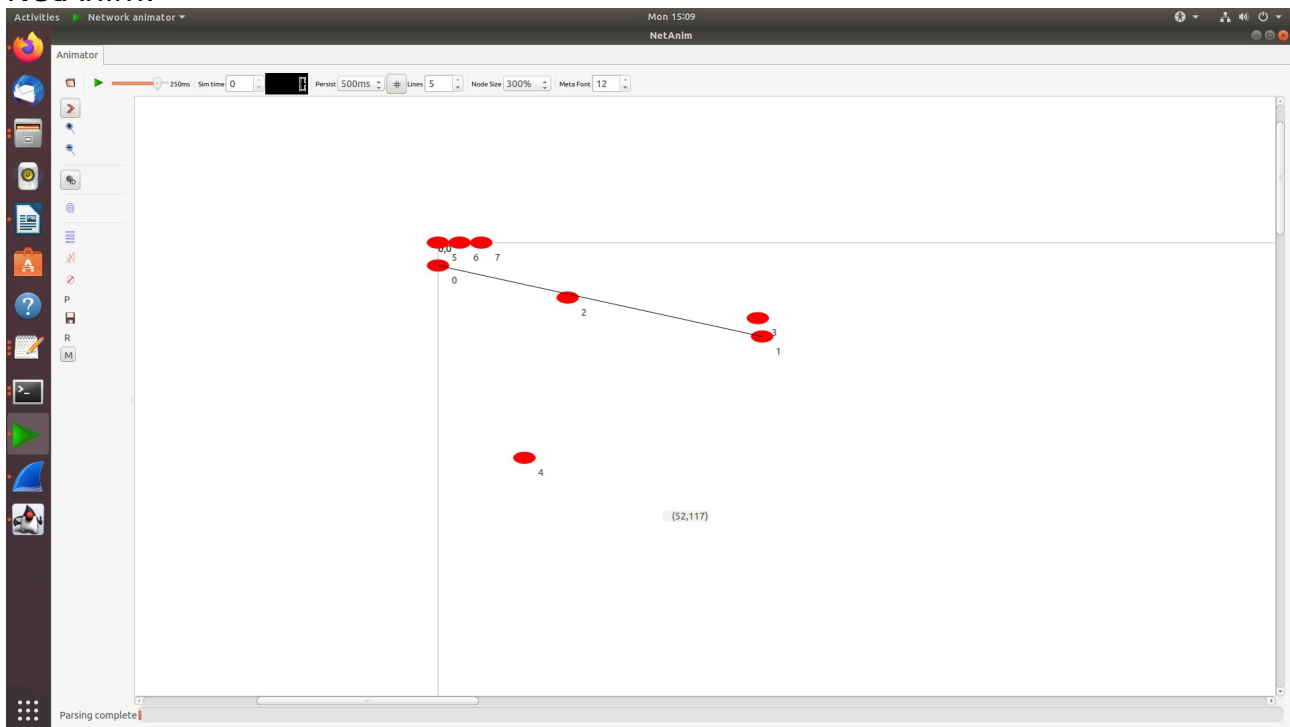
```
if (tracing)  
{  
    phy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);  
    pointToPoint.EnablePcapAll ("third");  
    phy.EnablePcap ("third", apDevices.Get (0));  
    csma.EnablePcap ("third", csmaDevices.Get (0), true);  
}
```

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

```
}
```

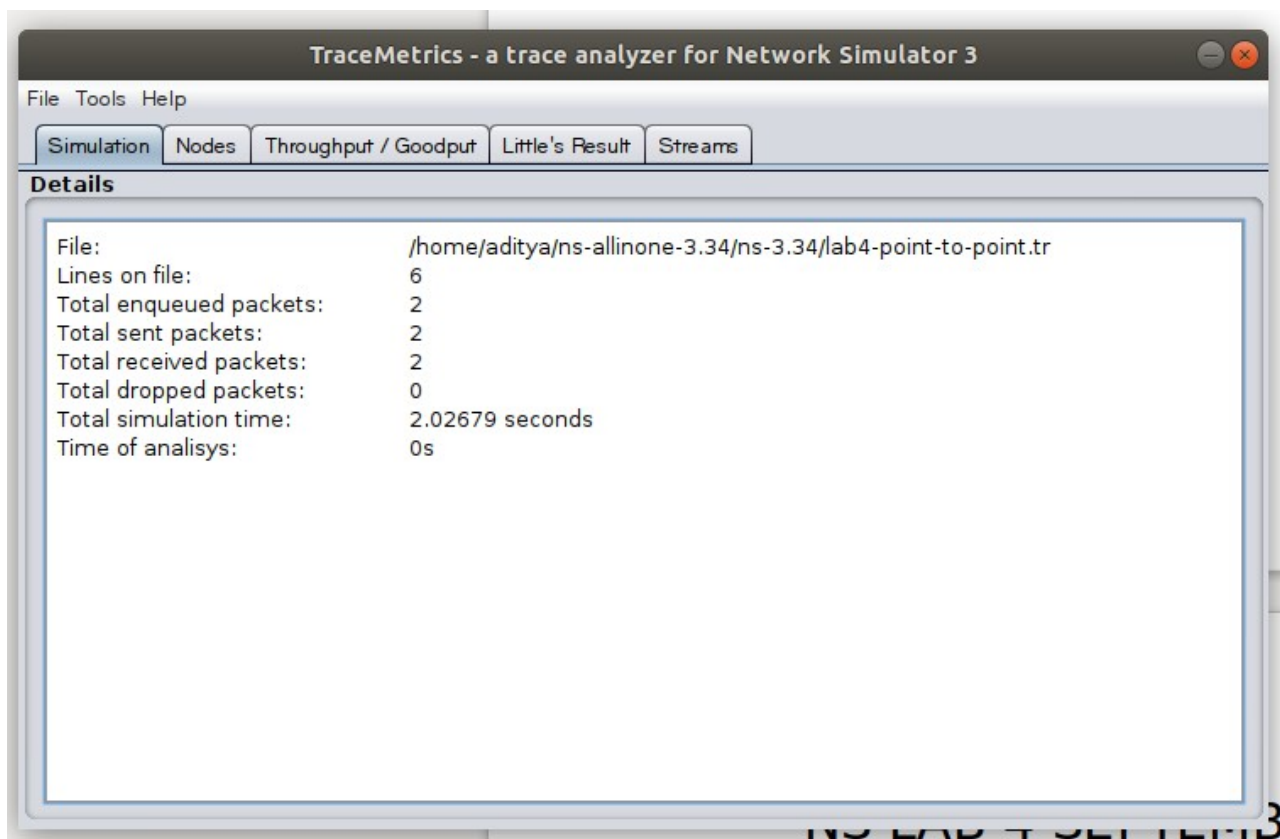
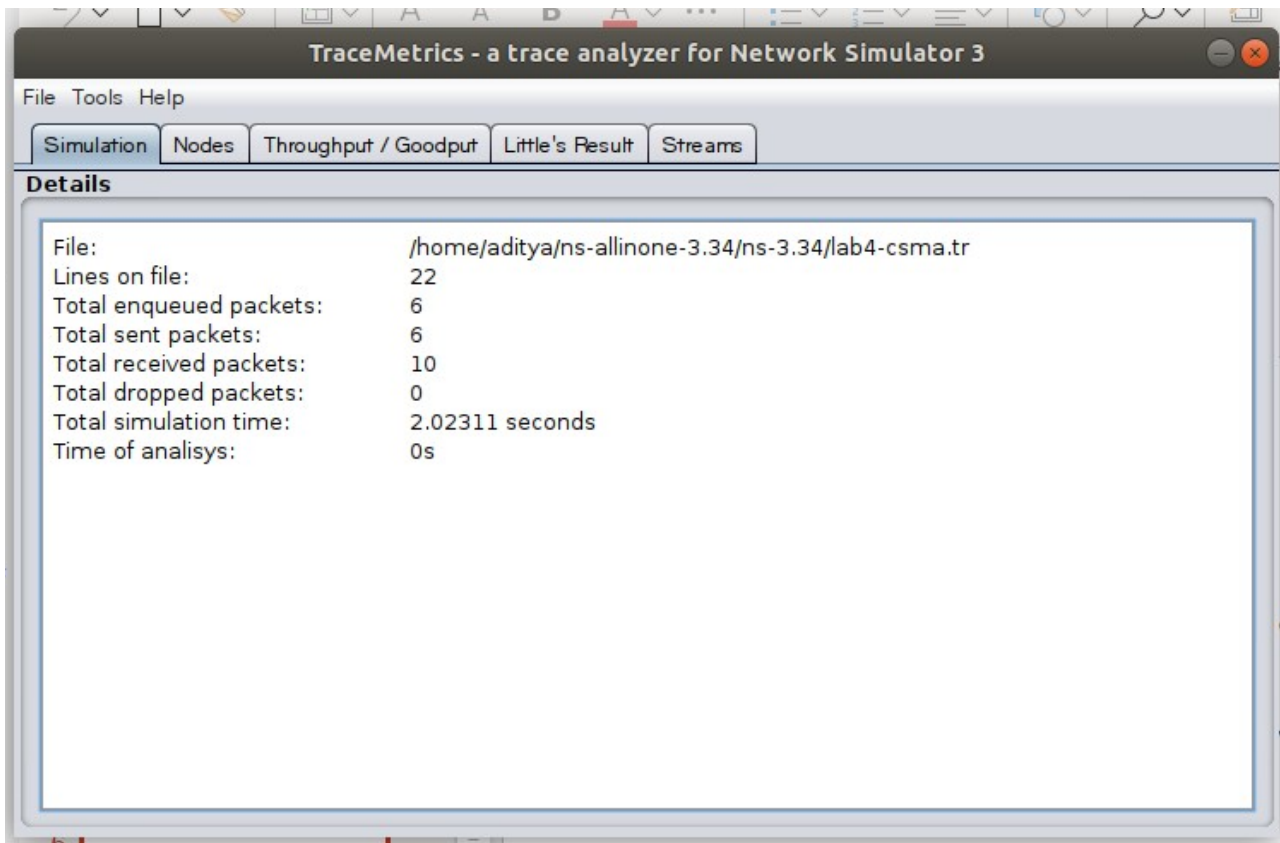
### OUTPUT:

#### NetAnim:



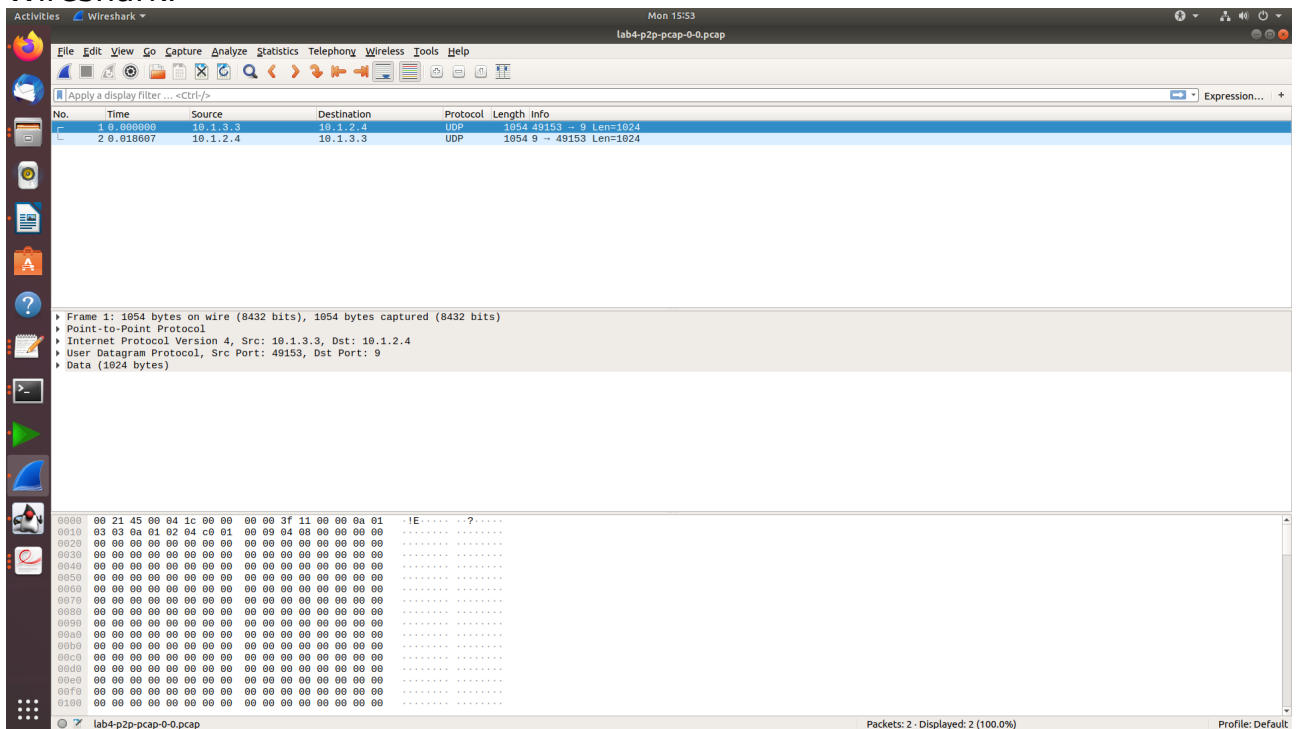
#### TraceMetrics:

## NS LAB 4 SEPTEMBER | Lab Assignment 4



# NS LAB 4 SEPTEMBER | Lab Assignment 4

## Wireshark:



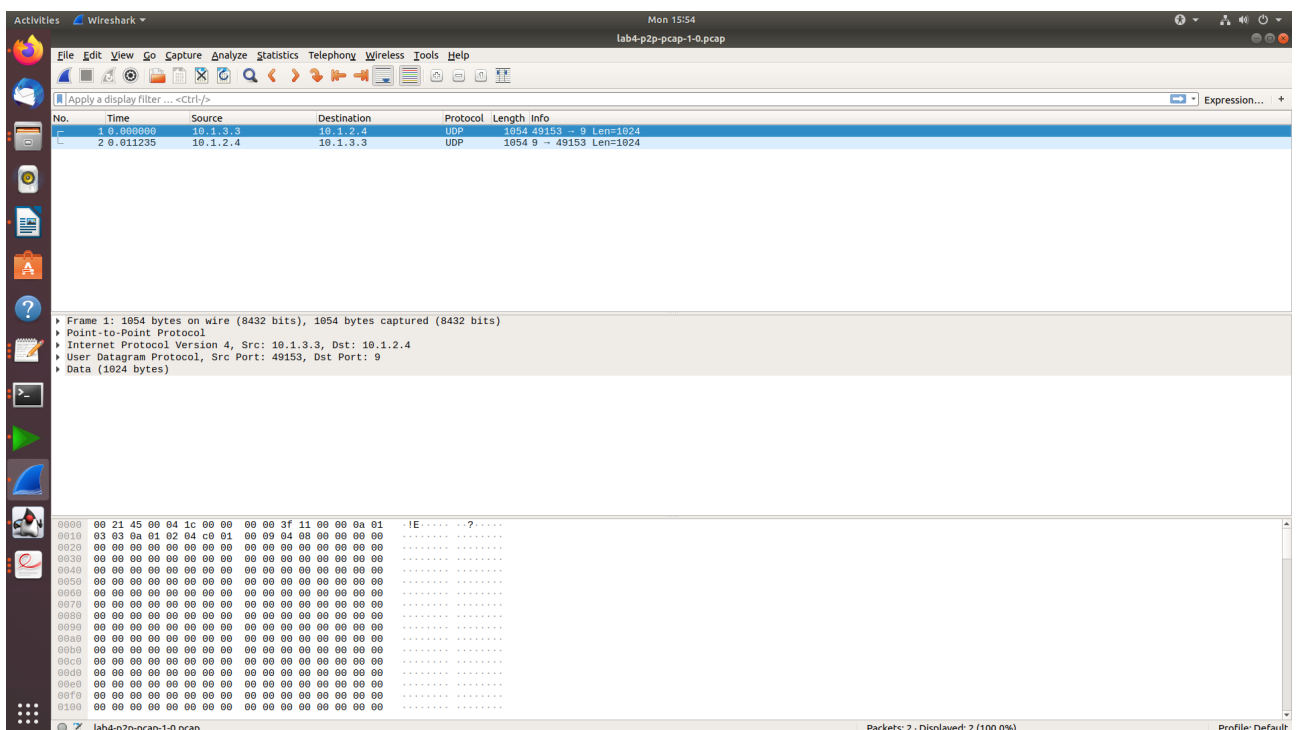
Wireshark interface showing packet capture data for lab4-p2p-pcap-0-0.pcap. The packet list displays two packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.3.3	10.1.2.4	UDP	1054	49153 → 9 Len=1024
2	0.018007	10.1.2.4	10.1.3.3	UDP	1054	9 → 49153 Len=1024

The packet details pane for packet 1 shows:

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

The packet bytes pane shows the raw data in hexadecimal and ASCII format.



Wireshark interface showing packet capture data for lab4-p2p-pcap-1-0.pcap. The packet list displays two packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.3.3	10.1.2.4	UDP	1054	49153 → 9 Len=1024
2	0.011235	10.1.2.4	10.1.3.3	UDP	1054	9 → 49153 Len=1024

The packet details pane for packet 1 shows:

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

The packet bytes pane shows the raw data in hexadecimal and ASCII format.

# NS LAB 4 SEPTEMBER | Lab Assignment 4

Wireshark interface showing packet capture data for lab4-pcap-1-1.pcap. The packet list shows several ARP and UDP packets. The packet details pane highlights Ethernet II, Src: 00:00:00:00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff), and Address Resolution Protocol (request). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00:00:00:03	Broadcast	ARP	64	Who has 10.1.2.4? Tell 10.1.2.1 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
2	0.000024	00:00:00:00:00:06	00:00:00:00:00:03	ARP	64	10.1.2.4 is at 00:00:00:00:00:06 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
3	0.000024	10.1.3.3	10.1.2.4	UDP	1076	49153 -> 9 Len=1024 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
4	0.005129	00:00:00:00:00:06	Broadcast	ARP	64	Who has 10.1.2.1? Tell 10.1.2.4 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5	0.005129	00:00:00:00:00:03	00:00:00:00:00:06	ARP	64	10.1.2.1 is at 00:00:00:00:00:03 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
6	0.005235	10.1.2.4	10.1.3.3	UDP	1076	9 -> 49153 Len=1024 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]

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

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

Packets: 6 - Displayed: 6 (100.0%) Profile: Default

Wireshark interface showing packet capture data for lab4-pcap-2-0.pcap. The packet list shows two ARP packets. The packet details pane highlights Ethernet II, Src: 00:00:00:00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff), and Address Resolution Protocol (request). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00:00:00:03	Broadcast	ARP	64	Who has 10.1.2.4? Tell 10.1.2.1 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
2	0.005117	00:00:00:00:00:06	Broadcast	ARP	64	Who has 10.1.2.1? Tell 10.1.2.4 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]

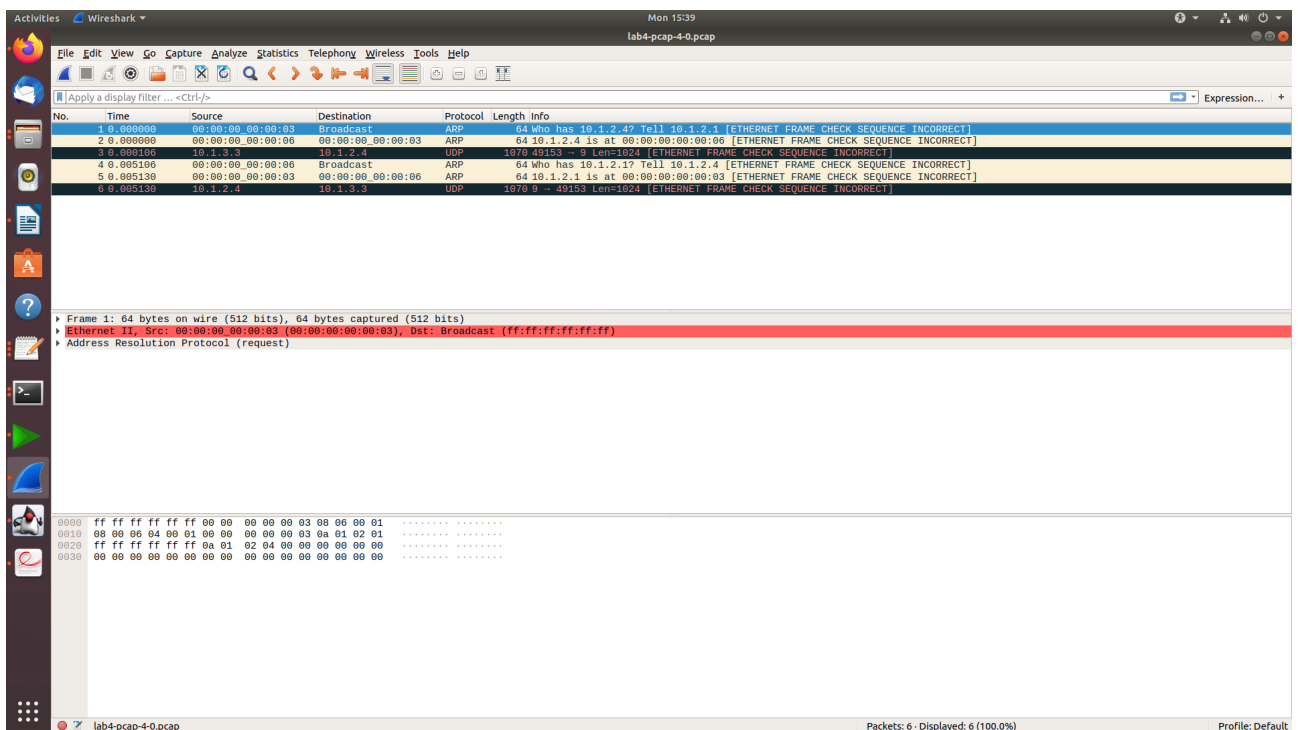
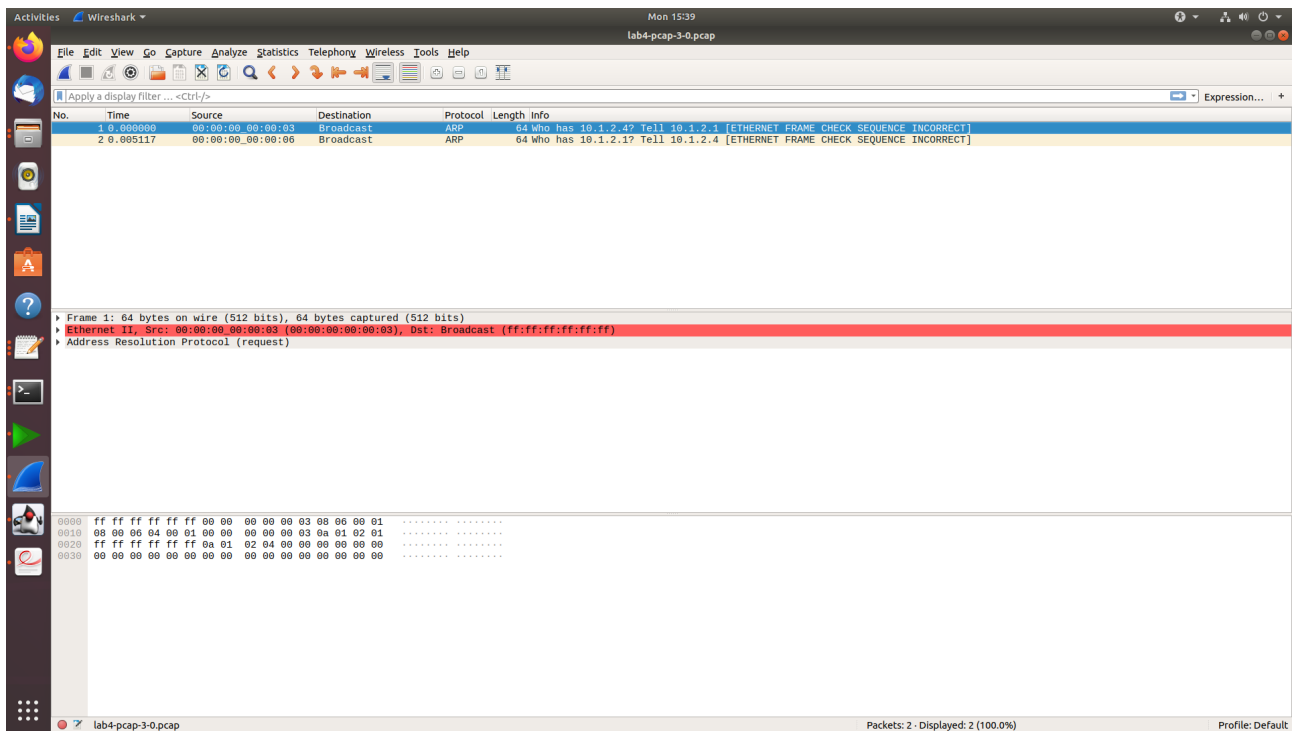
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)  
Ethernet II, Src: 00:00:00:00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
Address Resolution Protocol (request)

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

Packets: 2 - Displayed: 2 (100.0%) Profile: Default



# NS LAB 4 SEPTEMBER | Lab Assignment 4



Gnuplot:

# NS LAB 4 SEPTEMBER | Lab Assignment 4

