Northeastern University

EECE5155: Wireless Sensor Networks and the Internet of Things

Lab Assignment 1 Report

Authors:

Group #14
Alston Liu
Robert Doss

Instructor: Prof. Leonardo Bonati

Date: January 31st, 2025

Task 1

Experimental setup

Since we have no prior experience with ns-3 and wireshark, this is to familiarize ourselves to the library APIs and information being sent to and from the echo servers. In this task, we had to simulate 2 nodes with a network interface at each node. We modified the *first.cc* from the tutorial directory to simulate our own network. In this network, we are required to have 2 nodes.
In our code, we create a NodeContainer and create 2 nodes using the *create* function:

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

Afterwards, we changed the point-to-point link with a data rate of 10 Mbps and a delay of 2ms. We had to set the device attributes and channel attributes and initialize them to the other objects.

```
PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
```

We also need to set the IP Addresses for the point-to-point link using the IPv4 Address 192.168.2.0\24:

```
Ipv4AddressHelper address;

address.SetBase("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign(devices);
```

For our application, we would need something that simulates the data being sent. This is where our echo client comes. We changed the port number to 63 and set the packet size to 256 Bytes:

```
UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 63); // changed port
echoClient.SetAttribute("MaxPackets", UintegerValue(1));
echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient.SetAttribute("PacketSize", UintegerValue(256)); // changed packet size
```

Finally, we enable packet tracing for Wireshark to see:

```
// Enable PCAP tracing
pointToPoint.EnablePcapAll("first",true);
```

Results

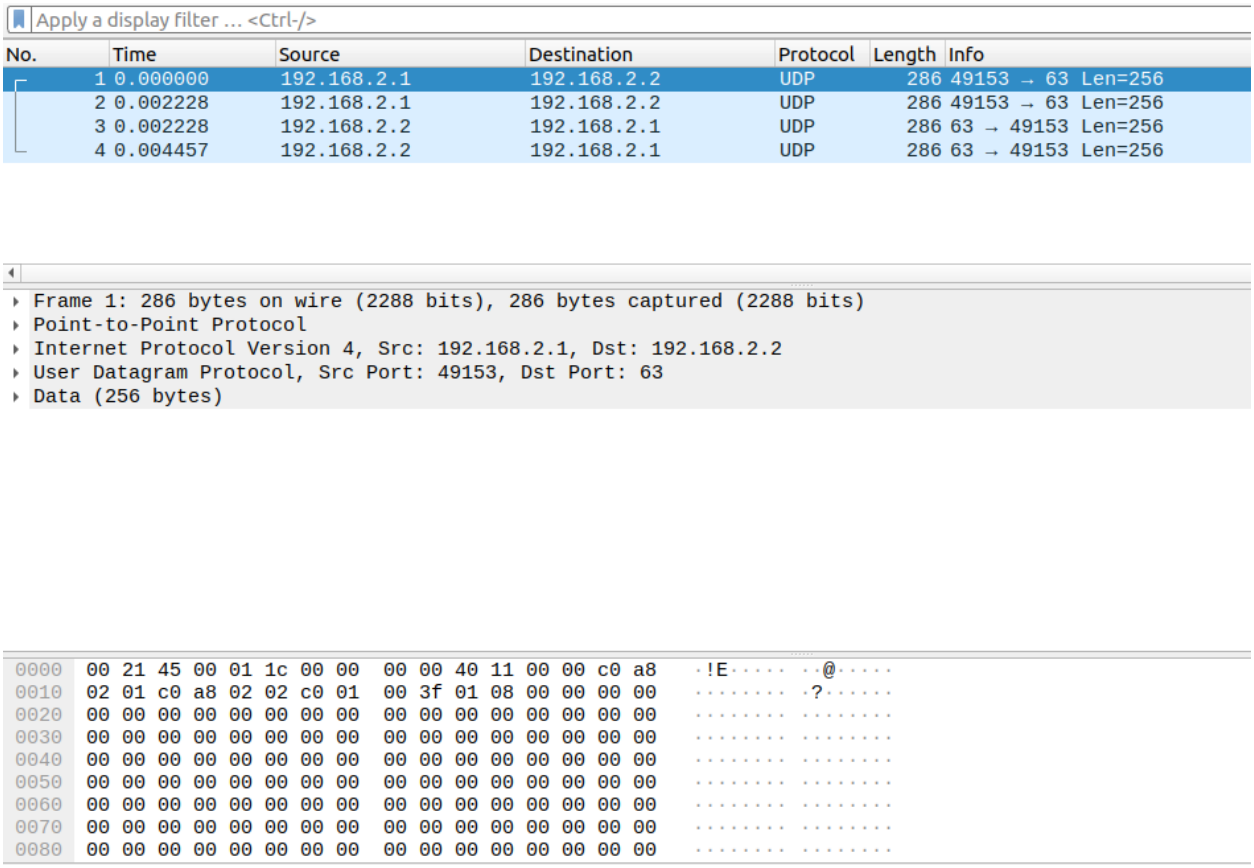
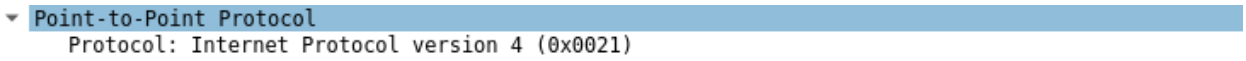


Figure 1: First packet being sent.

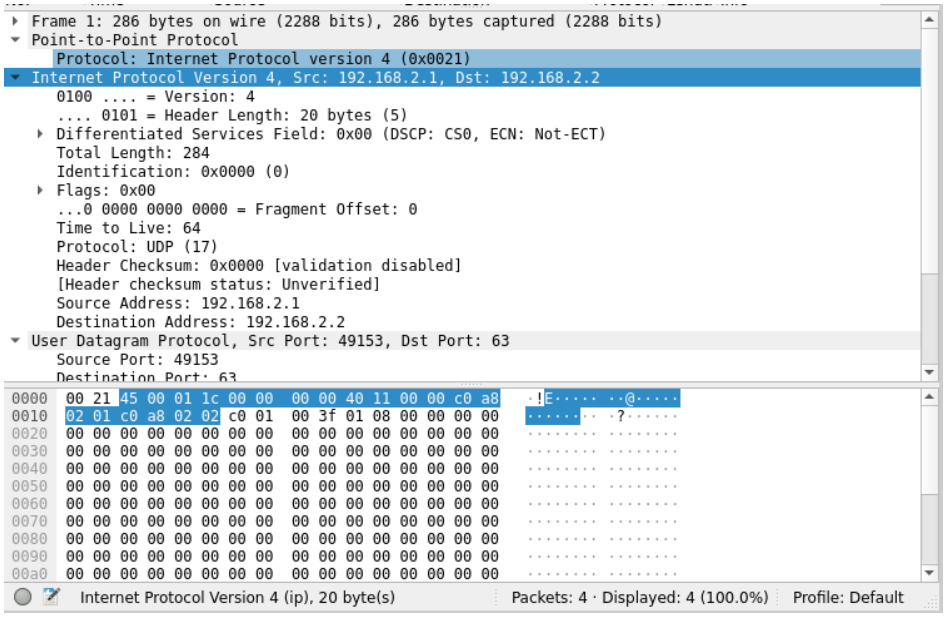
As we send the packets, we can see the UDP echo server sending data packets back and forth, with a propagation delay of 2 ms. In the bottom of the screen, we can see the data that's being sent from client and server. Since we see the first packet, it has a length of 256 Bytes being sent to the IP Address 192.168.2.2 from 192.168.2.1.

In the middle section and last section of the window, we can see the packet information. The first part shows the summary of the packet capture. The next section shows the point-to-point information and the protocol it uses:

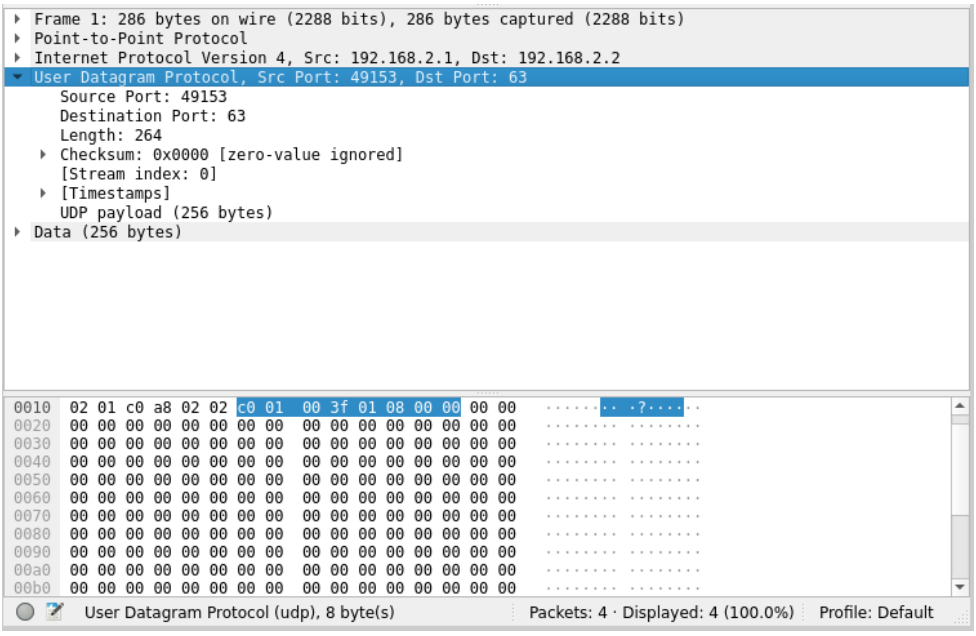


The point-to-point protocol uses IPv4 which are the addresses we use. In the first 2 Bytes in the last section we can see 0x0021 displayed. When we click on the bytes, it also highlights the information Wireshark finds.

Afterwards, we see the IPv4 source and destination addresses. If we expand this section, we can see a lot more information like length, protocol, flags, etc. Here's a screenshot of all the information we see in this section:



We can see certain parts of the bytes being sent and coordinate what information it corresponds to. The next section is the user datagram protocol which shows the source and destination ports and the length of this information and the data. We can see that this part of the data takes around 8 Bytes of information. The rest are the 256 Bytes being sent from our application.



The second packet is the same information but reverse where the source information is flipped into the destination section.

Learnt Lessons

In the first task, we learned about how to set up an UDP echo server using ns3 with 2 nodes that have speeds of 10 Mbps and 2ms delay. We learned how we can set up the addresses in a certain range or subnet. We observed that The UDP echo server communicates to and from Port 63 and 49153 carrying a total of 286 Bytes per packet with 30 Bytes of overhead. The data size is 256 Bytes but an additional 30 Bytes are added for the metadata, source, and destination information. We also learned how to use Wireshark to peer into the data that is being sent from the client and servers. We had to configure the pcap file to visualize our data in Wireshark.

Task 2

Experimental Setup

In this task, we need to set up 2 CMSA link. The network would contain 3 nodes in the shared bus operating under CSMA, 3 nodes in the second bus operating under CSMA, and 2 nodes in the Point-to-Point Link. We had to simulate a Point-to-Point link between node 2 and 3 while also simulating the CMSA links. The applications running in the network will be the UDP Echo Server at Node 1 and 5.

To create the nodes for the CSMA we do this:

```
uint32_t nCsma = 3;
// first bus
NodeContainer csmaBusA;
csmaBusA.Create(nCsma);

// second bus
NodeContainer csmaBusB;
csmaBusB.Create(nCsma);

// Bus 1 setup
CsmaHelper csmaA;
csmaA.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csmaA.SetChannelAttribute("Delay", TimeValue(MicroSeconds(10)));
NetDeviceContainer csmaADevices;
csmaADevices = csmaA.Install(csmaBusA);

// Bus 2 setup
CsmaHelper csmaB;
csmaB.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csmaB.SetChannelAttribute("Delay", TimeValue(MicroSeconds(10)));
NetDeviceContainer csmaBDevices;
```

```
csmaBDevices = csmaB.Install(csmaBusB);
```

This sets up the CSMA nodes with a rate of 100 Mbps and a delay of 10 µs. For each helper object, we assigned 3 nodes for each node in our simulation on both sides. Next we needed to setup the Point-to-Point link:

```
// Point-to-point setup
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install(p2pNodes);
```

Here we changed the data rate for the point-to-point link to 10 Mbps with a delay of 2ms. Now we currently have the CSMA bus 1 & 2 along with the point-2-point link object. Next, we setup the IP addresses for each.

```
// assign IP for bus 1
Ipv4AddressHelper address;
address.SetBase("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer csmaAInterfaces;
csmaAInterfaces = address.Assign(csmaADevices);

// assign IP for bus 2
address.SetBase("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaBInterfaces;
csmaBInterfaces = address.Assign(csmaBDevices);

// assign IP for point-to-point
address.SetBase("192.168.3.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign(p2pDevices);
```

We set the first CSMA bus 1 addresses to 192.168.1.x Subnet and bus 2 addresses to 192.168.2.x Subnet. Finally, we assigned the 192.168.3.x subnet to the point-to-point link. This establishes the connection between the CSMA buses to communicate with each other through this link. To send this data we had to setup the UDP echo servers. However, we would need to choose the node it is outputting from:

```
UdpEchoServerHelper echoServer(21);
ApplicationContainer serverApps = echoServer.Install(csmaBusA.Get(1));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient(csmaAInterfaces.GetAddress(1), 21);
echoClient.SetAttribute("MaxPackets", UintegerValue(2));
```

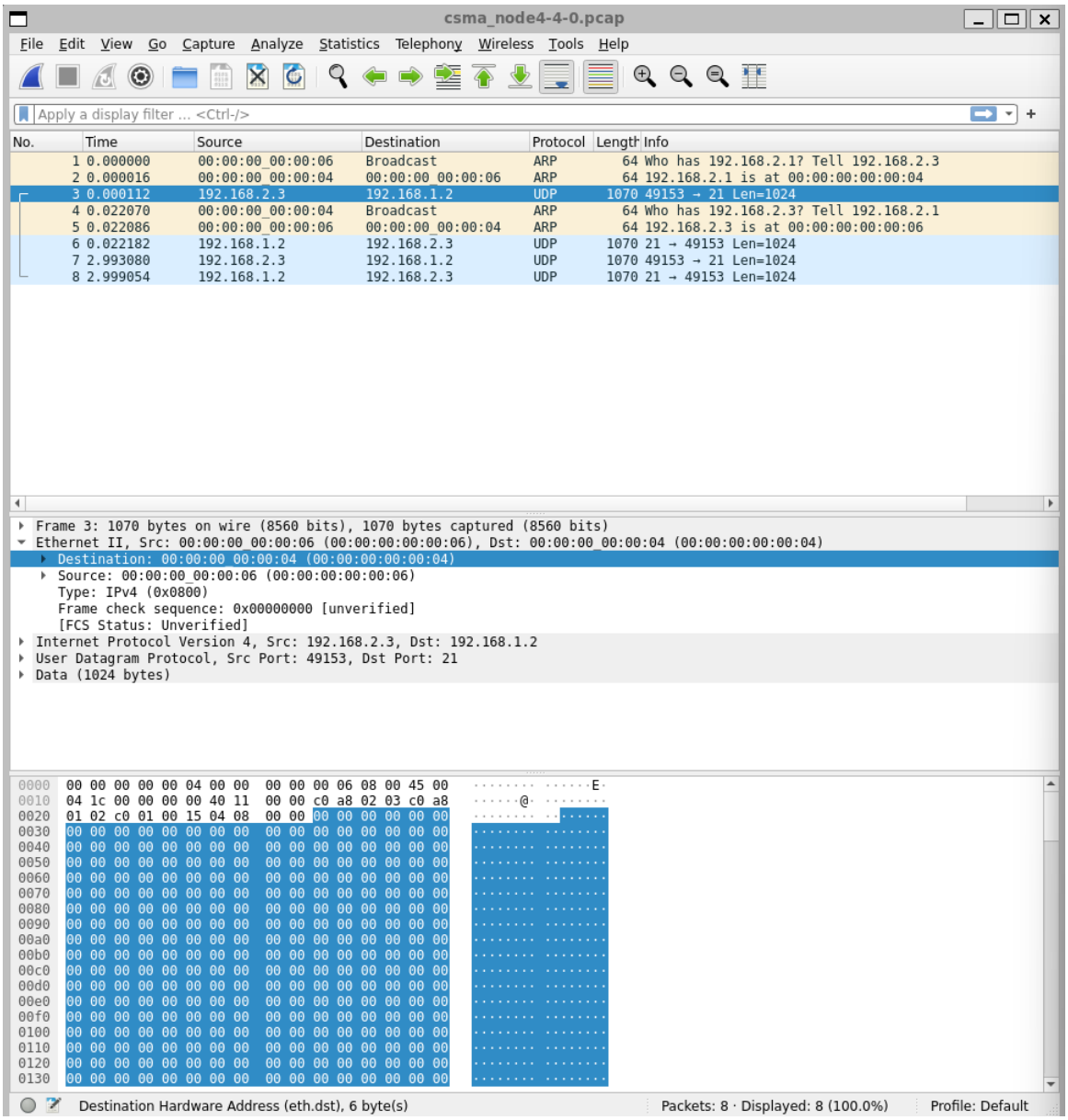
```
echoClient.SetAttribute("Interval", TimeValue(Seconds(3.0)));
echoClient.SetAttribute("PacketSize", UintegerValue(1024));
ApplicationContainer clientApps = echoClient.Install(csmaBusB.Get(2));
clientApps.Start(Seconds(4.0));
clientApps.Stop(Seconds(10.0));
```

This section of the code sets up the UDP echo server on Node 1 & Node 5. We set up the echo server to be communicating through port 21 with the packet data size being 1 KB and an interval of 3 seconds. We also set a limit to how many packets it can send and receive. However, to see the packets, we setup the pcap to be listening on Node 2 and 4.

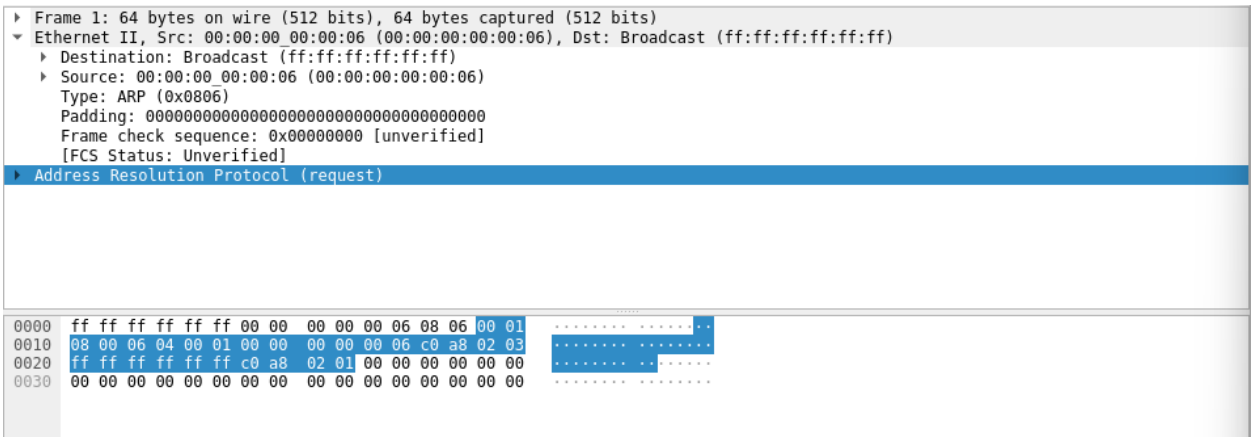
```
pointToPoint.EnablePcap("p2p_node2", p2pDevices.Get(0), true);
csmaB.EnablePcap("csma_node4", csmaBDevices.Get(1), true);
```

Results

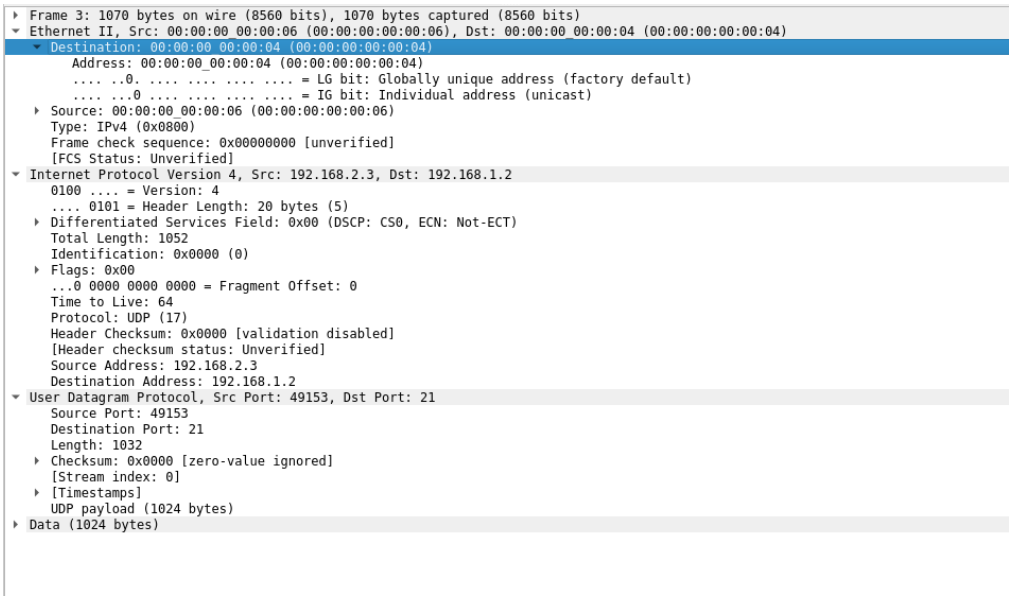
As we finish setting up our environment, we ran the simulation and created two files to analyze the nodes and the p2p link. We will first look at the CSMA nodes and how they are communicating. We recognize some lines similar to Task 1 where it sends an echo packet to the P2P link. However, we also see packet numbers 1,2,4, & 5 are different packets in both size and protocol. This leads us to believe that this is an internal packet for CSMA nodes:



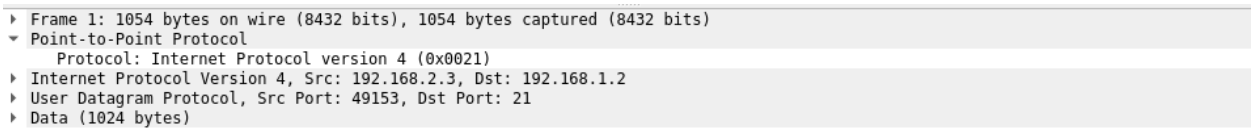
As we look closer into the data of the packet, we can see that the protocol it uses is ARP (Address Resolution Protocol). In this case, it is trying to find the address 192.168.2.1 and the source address would be 192.168.2.3. It sends these requests to other devices on the same subnet until it finds that specific address. We noticed that the source is **00:00:00:00:00:06** which is a weird address but found out that this is a MAC Address that is the method of identifying devices for the ARP protocol.



As we look at Packet 3, we can see that it sends data from 192.168.2.3 to the 192.168.1.2 address. This means that CSMA A and CSMA B are communicating and sending the correct number of packets and data. We see that there's 1070 bytes of data being sent and the data size is 1024. In this case, we noticed there's more overhead bytes on these packets compared to the 30 bytes in the previous task. Ethernet II packet is the addition to the packet compared to the first task.



However, when we see the P2P link analysis, we see that only around 1054 bytes of data were captured and sent.



This would be similar to the 30 bytes of overhead data is the same as the first task. Around 16 bytes of data were not used from the previous. I think this is used for the CMSA link to identify which nodes to send the packet from but from the perspective of the P2P link, it's all the same.

Learnt Lesson

In the second task, we modified the second.cc file from the ns-3.42 tutorials to satisfy the requirements of our system. The original system consisted of a point to point link between a single node and a CSMA bus. Our modified script simulates a system containing two CSMA buses connected via a point-to-point link. Completing this task allowed us to learn about communication between CSMA buses in ns-3.