

## EECE5155: Wireless Sensor Networks and the Internet of Things Laboratory Assignment 1 Report

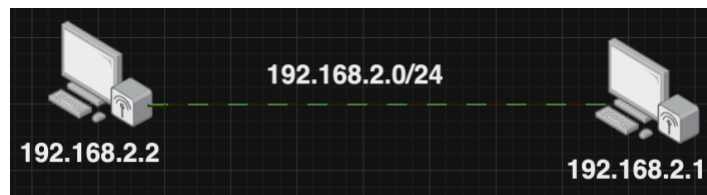
### Author:

- Soniya Nitin Kadam
- Group number on Canvas: 27

Date: 09/29/2023

### Task 1

#### 1. Experimental setup



##### a) 2 nodes, 1 network interface at each node.

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

In both first.cc and Assignment1 Task1 we are supposed to create 2 nodes. Both connected to each other in point-to-point link. Point-to-point are links which have a dedicated connection with each other so our nodes do not have access to this link. *NodeContainer* here is a class which allows us to store the node objects, and *nodes.create(2)*, creates 2 nodes in this point-to-point link.

##### b) Point to Point link with:

- Data rate: 10Mbps
- Delay: 2ms

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

In the sample code, the Data rate is 5Mbps and Delay is 2ms, in this code we have changed the data rate to 10Mbps and kept the same Delay. We are able to make these changes and add these specification to the link by using *.SetChannelAttribute* method.

### c) IP Address Assignment – 192.168.2.0/24

```
Ipv4AddressHelper address;  
address.SetBase("192.168.2.0", "255.255.255.0");
```

The `address.SetBase` method allows us to set the network address & the network mask of the link. In the example the IP address of the link is 10.1.1.0/24, so we have changed the IP address to 192.168.2.0/24. The /24 in this case means 255.255.255.0 which indicates that the IP address belongs to class C.

### d) Application

- UDP Echo server on port 63
- Packet Size – 256 bytes

```
UdpEchoServerHelper echoServer(63);  
  
ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));  
serverApps.Start(Seconds(1.0));  
serverApps.Stop(Seconds(10.0));  
  
UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 63);  
echoClient.SetAttribute("MaxPackets", UintegerValue(1));  
echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));  
echoClient.SetAttribute("PacketSize", UintegerValue(256));
```

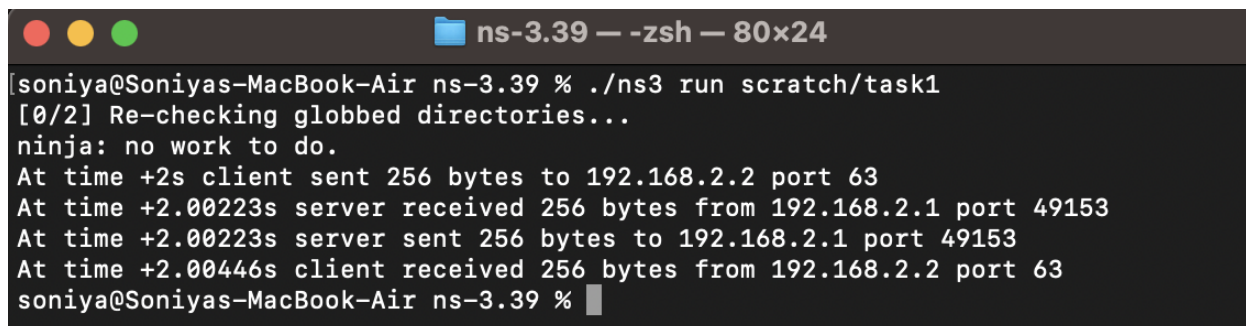
The line `UdpEchoServerHelper echoServer(63)`, creates a `UdpEchoServerHelper` object and assigns it to the variable `echoServer`. The `UdpEchoServerHelper` class is used to create and install UDP echo server applications.

The 63 in the constructor of the `UdpEchoServerHelper` object specifies the port number that the echo server will listen on. The first parameter to the `UdpEchoClientHelper` constructor is the IP address of the UDP echo server. The second parameter is the port number that the echo server is listening on.

The `echoClient.SetAttribute()` calls are used to set the following attributes of the UDP echo client application:

- MaxPackets: The maximum number of packets to send. In this code we are sending only one packet.
- Interval: The time interval between sending packets. In this code the time interval is set to only one second.
- PacketSize: The size of each packet to send. In this code the packet size is set to 256 bytes which in the tutorial file was set to 1024 bytes.

## 2. Results



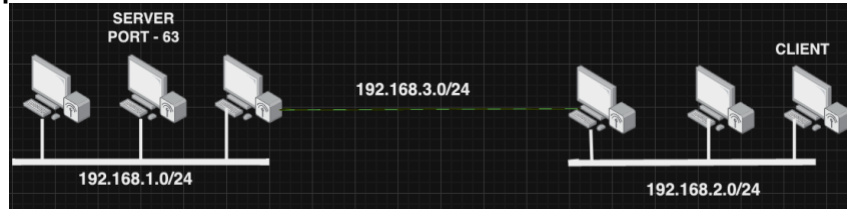
```
ns-3.39 --zsh-- 80x24  
[soniya@Soniya-MacBook-Air ns-3.39 % ./ns3 run scratch/task1  
[0/2] Re-checking globbed directories...  
ninja: no work to do.  
At time +2s client sent 256 bytes to 192.168.2.2 port 63  
At time +2.00223s server received 256 bytes from 192.168.2.1 port 49153  
At time +2.00223s server sent 256 bytes to 192.168.2.1 port 49153  
At time +2.00446s client received 256 bytes from 192.168.2.2 port 63  
soniya@Soniya-MacBook-Air ns-3.39 %
```

The image above is the output of task one after making the changes given to in assignment. We can see that at time 2sec the client sends a packet of 256 bytes to the IP address 192.168.2.2 at port 63, which is the IP address of the server. After the packet is received by the server, the server sends back a packet of 256 bytes to 192.168.2.1 at port 49153 which we can analyze to be the IP address of the client. This procedure takes place twice. In the tutorial first.cc the bytes that are sent to the client are 1024 to client port 9 with IP address range 10.1.1.0/24



## Task 2

### 1. Experimental setup



```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE("Assignment1Task2");

int
main(int argc, char* argv[])
{
    bool verbose = true;
    uint32_t nCsmal = 2;
    uint32_t nCsma2 = 2;

    CommandLine cmd(__FILE__);
    cmd.AddValue("nCsmal", "Number of \"extra\" CSMA nodes/devices", nCsmal);
    cmd.AddValue("nCsma2", "Number of \"extra\" CSMA nodes/devices", nCsma2);
    cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);

    cmd.Parse(argc, argv);

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

    nCsmal = nCsmal == 0 ? 1 : nCsmal;
    nCsma2 = nCsma2 == 0 ? 1 : nCsma2;

    NodeContainer p2pNodes;
    p2pNodes.Create(2);

    NodeContainer csmaNodes2;
    csmaNodes2.Add(p2pNodes.Get(1));
    csmaNodes2.Create(nCsma2);

    NodeContainer csmaNodes1;
    csmaNodes1.Add(p2pNodes.Get(0));
    csmaNodes1.Create(nCsmal);
```

The lines `uint32_t nCsmal = 2` & `uint32_t nCsma2 = 2`, create variables `nCsmal` & `nCsma2` both with value 2. Further we create a point-to-point link with 2 nodes using the command `p2pNodes.Create(2);`, here (2) symbolizes the number of nodes. Once all 3 links are created separately, we add the `nCsmal` & `nCsma2` links to each of the nodes to create an entire link. This is accomplished by `csmaNodes2.Add(p2pNodes.Get(1)); csmaNodes2.Create(nCsma2);` & similarly `csmaNodes1.Add(p2pNodes.Get(0)); csmaNodes1.Create(nCsmal);`.

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

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

CsmaHelper ncsma1;
ncsma1.SetChannelAttribute("DataRate", StringValue("100Mbps"));
ncsma1.SetChannelAttribute("Delay", TimeValue(MicroSeconds(10)));

CsmaHelper ncsma2;
ncsma2.SetChannelAttribute("DataRate", StringValue("100Mbps"));
ncsma2.SetChannelAttribute("Delay", TimeValue(MicroSeconds(10)));
```

In the original second.cc there 2 specifications of the link attributes given one for the CSMA link and one for Point to Point, in this case we have 3 links for which we must define the attributes. Initially in the sample code the point-to-point link had data rate as 5Mbps which is changed here to be 10Mbps while the delaying both the cases remains the same. Similarly, for CSMA link, the CSMA link in the sample had data rate as 100Mbps and delay as 6560 nano seconds, in this case the data rate for both CSMA links remains the same while the delay is changed to 10 microseconds. All this is accomplished by using *SetChannelAttribute* which allows us to set various channel parameters.

```
Ipv4AddressHelper address;
address.SetBase("192.168.3.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign(p2pDevices);

address.SetBase("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces1;
csmaInterfaces1 = address.Assign(csmaDevices1);

address.SetBase("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces2;
csmaInterfaces2 = address.Assign(csmaDevices2);
```

Here, we are assigning IP address to the point-to-point link and the 2 CSMA links. For the CSMA link 1 we have assigned 192.168.1.0/24, for CSMA link 2 the IP address assigned is 192.168.2.0/24 and for point to point link the IP address assigned is 192.168.3.0/24. This is done by *address.SetBase* which allows us to give an IP address and assigned it to a particular link.

```
UdpEchoServerHelper echoServer(21);

ApplicationContainer serverApps = echoServer.Install(csmaNodes1.Get(1));
serverApps.Start(Seconds(4.0));
serverApps.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient(csmaInterfaces1.GetAddress(1), 21);
echoClient.SetAttribute("MaxPackets", UintegerValue(2));
echoClient.SetAttribute("Interval", TimeValue(Seconds(3.0)));
echoClient.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps = echoClient.Install(csmaNodes2.Get(nCsmas2));
clientApps.Start(Seconds(4.0));
clientApps.Stop(Seconds(10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables();

nCsmas1.EnablePcap("serverside", csmaDevices1.Get(0), true);
nCsmas2.EnablePcap("clientside", csmaDevices2.Get(1), true);

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

We are supposed to echo the server at port number 21, therefore we add the link *UdpEchoServerHelper echoServer(21);*. We are able to start and stop the client and server using *serverApps.Start(Seconds(4.0));*, *serverApps.Stop(Seconds(10.0));* for the server & *clientApps.Start(Seconds(4.0));*, *clientApps.Stop(Seconds(10.0));*, for the client, we are doing this according to the specification given to us to send the first packet at 4 seconds. Further we have set attributes for the packets that need to be sent, we are sending 2 packets, we have used *echoClient.SetAttribute* to set *MaxPackets* which will define the number of packets to be sent i.e. 2 in this case, the *Interval* to set the interval at which both the packets to be sent which is set at 3 because we have to send one at 4 sec and other at 7 sec and lastly *PacketSize* which is set to 1024 bytes which tells us each packet will be of 1024 byte *nCsmas1.EnablePcap("serverside", csmaDevice1.Get(0), true);* is used to generate a Pcap (Wireshark) compatible file from the CSMA link 1 with index (0) which is 192.168.1.1 i.e. Node2. Similarly, the next line *nCsmas2.EnablePcap("clientside", csmaDevice2.Get(1), true);* is

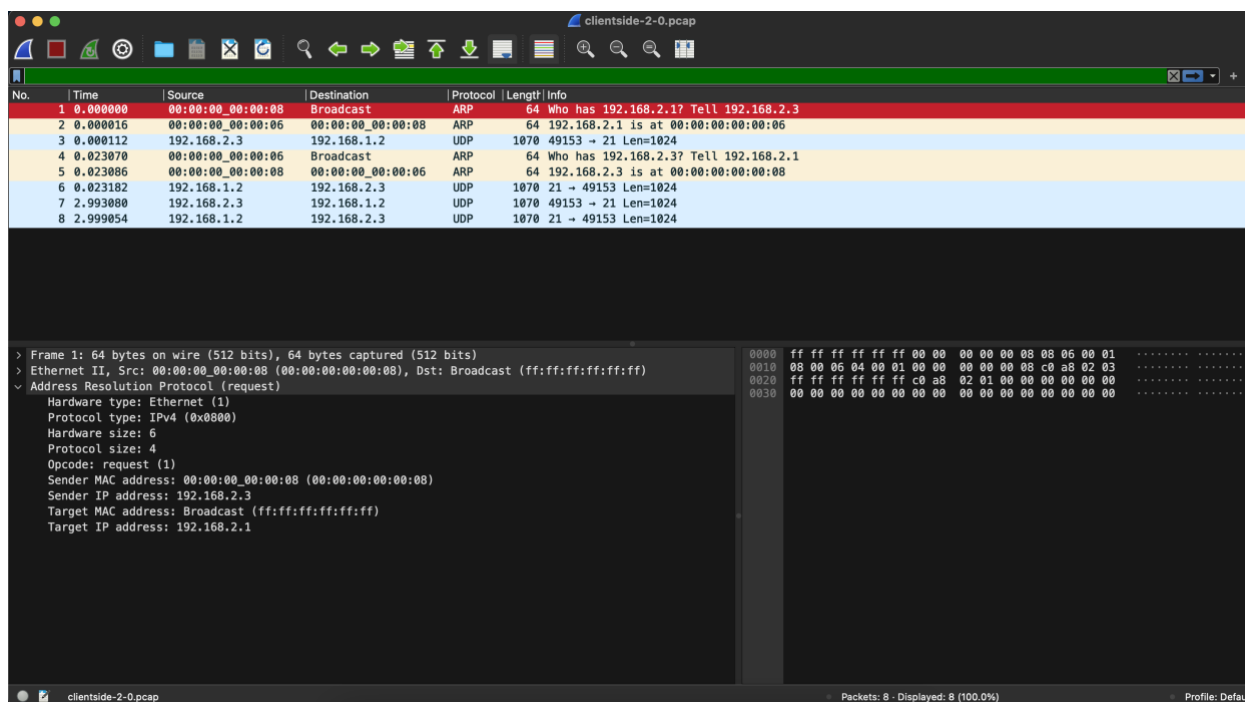


used to generate a Pcap (Wireshark) compatible file from the CSMA link 2 with index (1) which is 192.168.2.2 i.e., Node4.

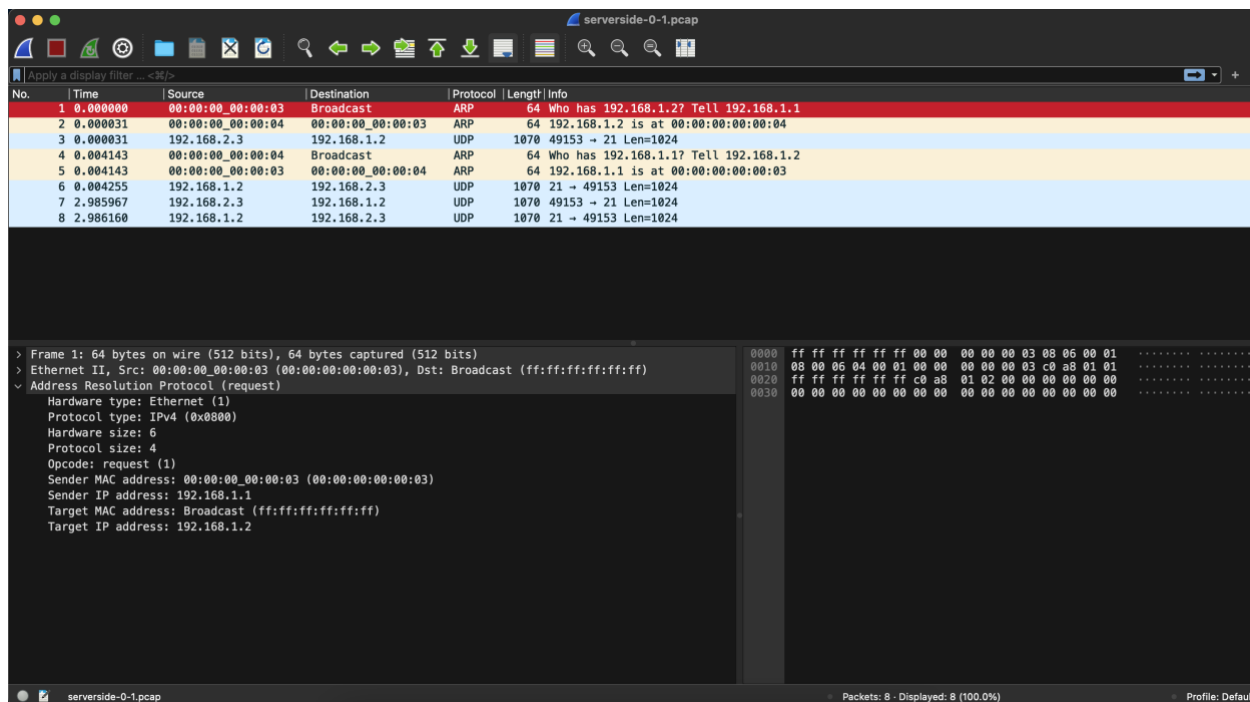
## 2. Results

```
ns-3.39 — -zsh — 80x24
soniya@Soniyas-MacBook-Air ns-3.39 % ./ns3 run scratch/task2
[0/2] Re-checking globbed directories...
ninja: no work to do.
At time +4s client sent 1024 bytes to 192.168.1.2 port 21
At time +4.0171s server received 1024 bytes from 192.168.2.3 port 49153
At time +4.0171s server sent 1024 bytes to 192.168.2.3 port 49153
At time +4.0302s client received 1024 bytes from 192.168.1.2 port 21
At time +7s client sent 1024 bytes to 192.168.1.2 port 21
At time +7.00303s server received 1024 bytes from 192.168.2.3 port 49153
At time +7.00303s server sent 1024 bytes to 192.168.2.3 port 49153
At time +7.00607s client received 1024 bytes from 192.168.1.2 port 21
soniya@Soniyas-MacBook-Air ns-3.39 %
```

Since we are sending 2 packets one at 4 seconds and other at 7 second, we can see that in front of time there are time stamps which clearly mention 4s and 7s. In both the time stamps, the client is sending to the server with IP address 192.168.1.2 at port 21 and sender is receiving the packet from the client with IP address 192.168.2.3 port 49153, all of this is clearly visible in the output. From both ends we are sending and receiving the data packet of size 1204 bytes.



This Wireshark screen shot is from the client side. In the first line we can see that the destination shows as Broadcast with protocol as ARP, so what happens is the client is trying to find route towards the server and send a broadcast message to find which node has the closest connection towards the server. In our case 192.168.2.3 is the client and 192.168.2.1 is the node (3) connected to a point-to-point network with 192.168.1.1 which is in the servers CSMA link. This is done using the ARP protocol which is used to map IP address to MAC addresses. Therefore 192.168.2.3 (client) sends the packet to 192.168.1.2 (server) on port 21 using this information. The packet size as we can see is 1024 bytes.



This Wireshark screen shot is from the server side. In the fourth line we can see that the destination shows as Broadcast with protocol as ARP, so what happens is the server is now trying to find route towards the client and send a broadcast message to find which node has the closest connection towards the client. In our case 192.168.1.2 is the server and 192.168.1.1 is the node (2) connected to a point-to-point network with 192.168.2.1 which is in the client CSMA link. This is done using the ARP protocol which is used to map IP address to MAC addresses. Therefore 192.168.1.2 (server) from port 21 sends the packet to 192.168.2.3 (client) on port 49153 using this information. The packet size as we can see in 1024 bytes.

Once the connection is establishing the transfer of data is taking place in UDP protocol.

### 3. Learnt Lessons

We have learned how to create a network using point to point and CSMA links and to create a network with more the 2 connecting nodes. We have also analyzed the working of different protocols and how the assigning of IP address and communication between nodes takes place. We have studied about new protocols like UDP & ARP and how they are used in this type of communication systems.