Northeastern University

EECE5155: Wireless Sensor Networks and the Internet of Things
Laboratory Assignment 2 Report

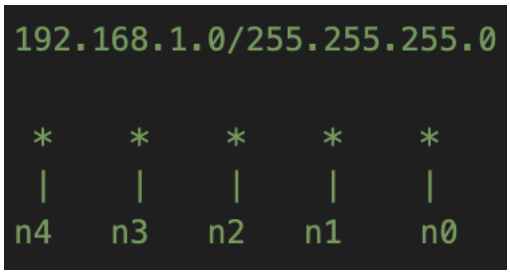
Author:
- Soniya Nitin Kadam
- Group number on Canvas: 27

Date: 10/29/2023

Task 1

1. Experimental setup

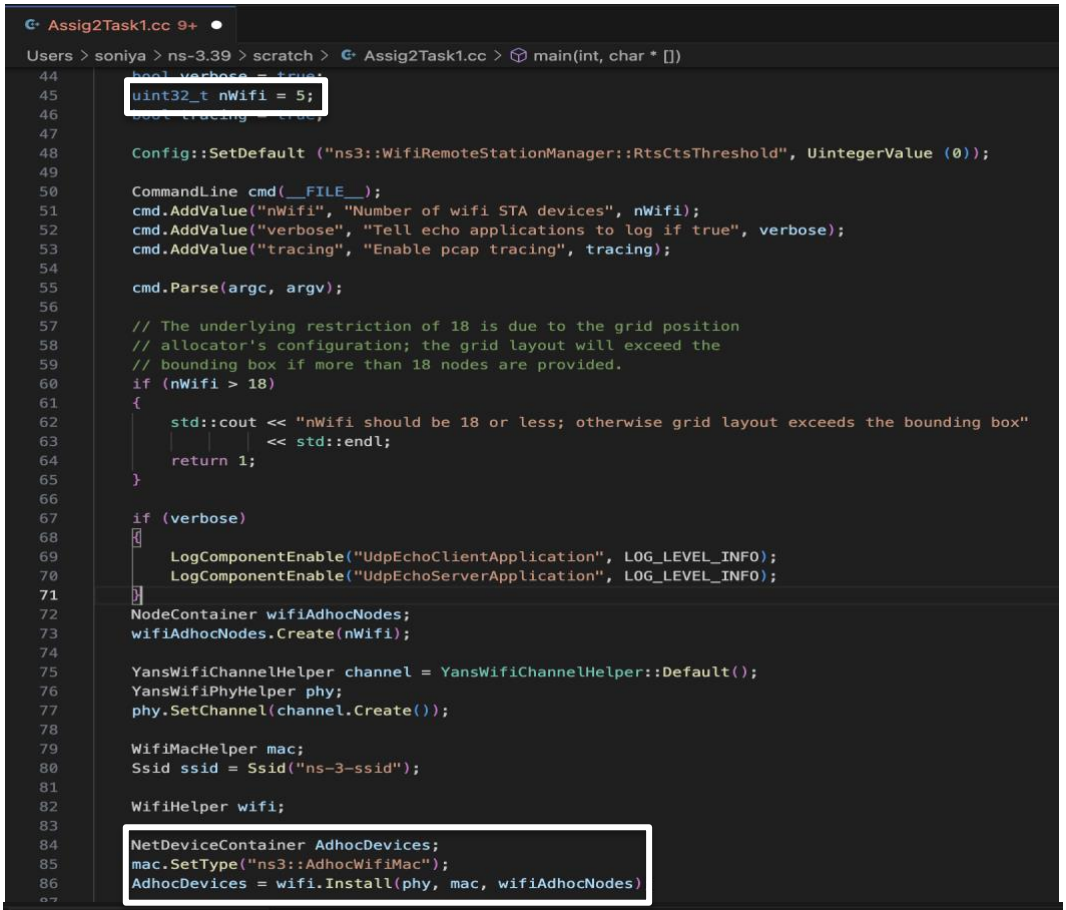
In this lab task 1, we have made 5 Wi-Fi nodes.



a) The IP addresses assigned for the network is 192.168.1.0 / 255.255.255.0. The nodes assign IP address according to node number. Here the node vise IP addresses are –
   - 192.168.1.1 – Node 0
   - 192.168.1.2 – Node 1
   - 192.168.1.3 – Node 2
   - 192.168.1.4 – Node 3
   - 192.168.1.5 – Node 4

b) We have removed the Point-to-point connection, the Access Point and the CSMA link because we only require the 5 WIFI nodes.
c) Rather than assigning one node as Client we have 2 nodes (node 3 & node 4), sending data to one Server at node 0 at port number 20.
d) We are echoing the Server at 4 instances, 2 instances by Node 3 with IP 192.168.1.4 at 1 second and 2 second mark and the size of the packet is 512 bytes. 2 instances by Node 4 with IP 192.168.1.5 at 2 second and 4 second mark and the size of the packet is 512 bytes.
e) The dimensions of the rectangle have also changed to -90,90,-90,90
f) We are creating PCAP file only at Node 1.
g) We have forced RTC/CTS to the code and verified the output.

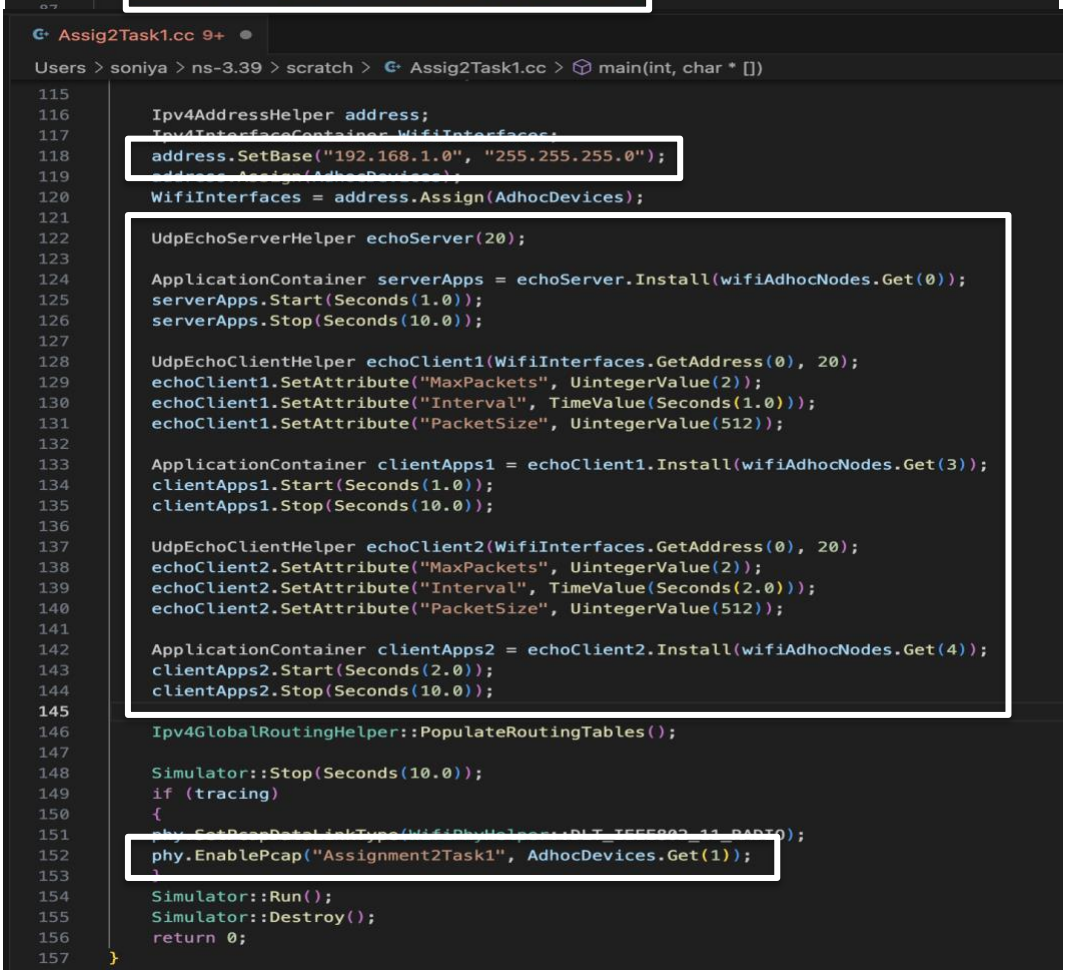
COMMENTS ARE ADDED IN THE CODE



Assig2Task1.cc 9+

Users > soniya > ns-3.39 > scratch > Assig2Task1.cc > main(int, char * [])

```
44      bool verbose = true;
45      uint32_t nWifi = 5;
46      bool tracing = true;
47
48      Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", UintegerValue (0));
49
50      CommandLine cmd(__FILE__);
51      cmd.AddValue("nWifi", "Number of wifi STA devices", nWifi);
52      cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);
53      cmd.AddValue("tracing", "Enable pcap tracing", tracing);
54
55      cmd.Parse(argc, argv);
56
57      // The underlying restriction of 18 is due to the grid position
58      // allocator's configuration; the grid layout will exceed the
59      // bounding box if more than 18 nodes are provided.
60      if (nWifi > 18)
61      {
62          std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding box"
63                    << std::endl;
64          return 1;
65      }
66
67      if (verbose)
68      {
69          LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
70          LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
71      }
72      NodeContainer wifiAdhocNodes;
73      wifiAdhocNodes.Create(nWifi);
74
75      YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
76      YansWifiPhyHelper phy;
77      phy.SetChannel(channel.Create());
78
79      WifiMacHelper mac;
80      Ssid ssid = Ssid("ns-3-ssid");
81
82      WifiHelper wifi;
83
84      NetDeviceContainer AdhocDevices;
85      mac.SetType("ns3::AdhocWifiMac");
86      AdhocDevices = wifi.Install(phy, mac, wifiAdhocNodes)
```



Assig2Task1.cc 9+

Users > soniya > ns-3.39 > scratch > Assig2Task1.cc > main(int, char * [])

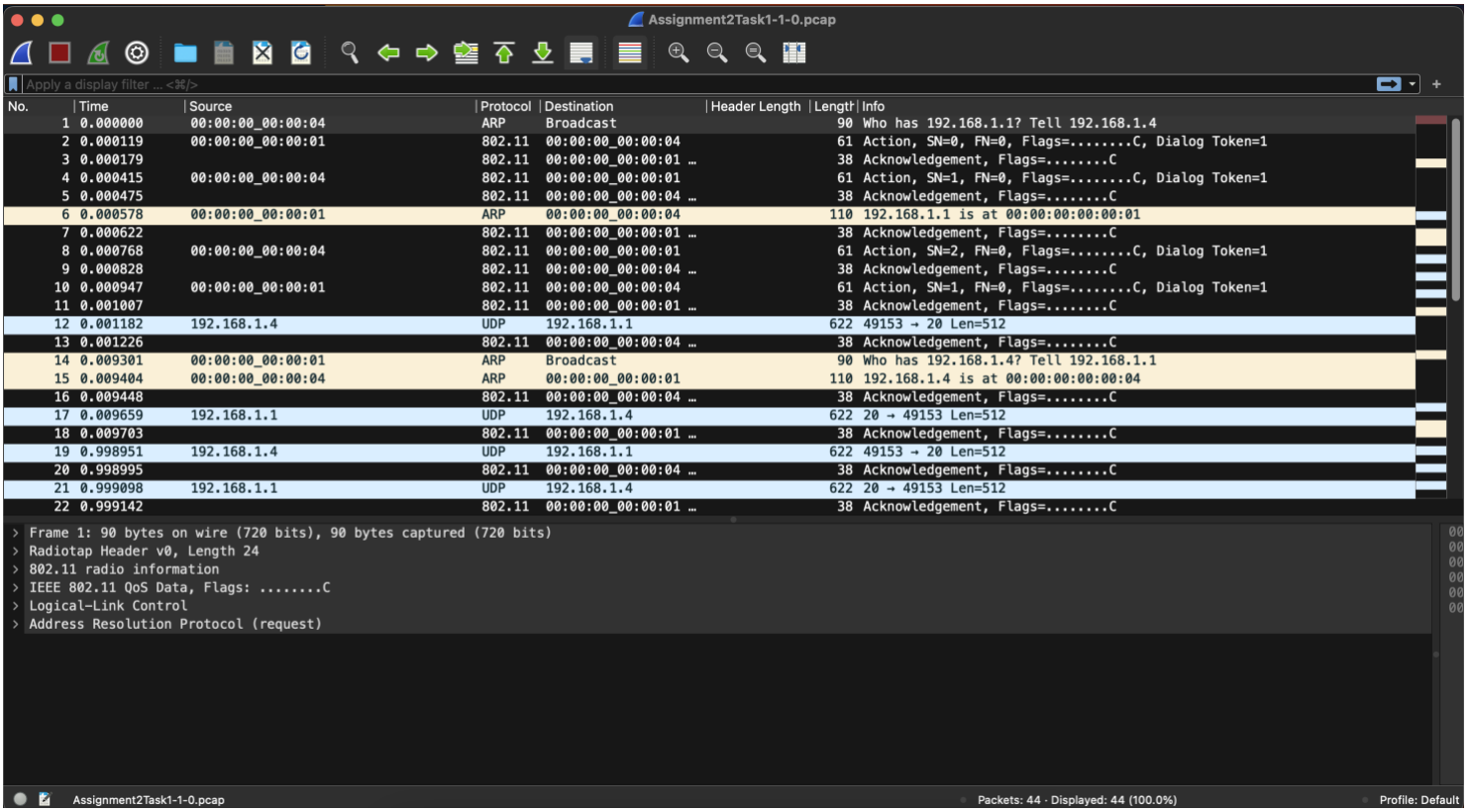
```
115
116     Ipv4AddressHelper address;
117     Ipv4InterfaceContainer WifiInterfaces;
118     address.SetBase("192.168.1.0", "255.255.255.0");
119     address.Assign(AdhocDevices);
120     WifiInterfaces = address.Assign(AdhocDevices);
121
122     UdpEchoServerHelper echoServer(20);
123
124     ApplicationContainer serverApps = echoServer.Install(wifiAdhocNodes.Get(0));
125     serverApps.Start(Seconds(1.0));
126     serverApps.Stop(Seconds(10.0));
127
128     UdpEchoClientHelper echoClient1(WifiInterfaces.GetAddress(0), 20);
129     echoClient1.SetAttribute("MaxPackets", UintegerValue(2));
130     echoClient1.SetAttribute("Interval", TimeValue(Seconds(1.0)));
131     echoClient1.SetAttribute("PacketSize", UintegerValue(512));
132
133     ApplicationContainer clientApps1 = echoClient1.Install(wifiAdhocNodes.Get(3));
134     clientApps1.Start(Seconds(1.0));
135     clientApps1.Stop(Seconds(10.0));
136
137     UdpEchoClientHelper echoClient2(WifiInterfaces.GetAddress(0), 20);
138     echoClient2.SetAttribute("MaxPackets", UintegerValue(2));
139     echoClient2.SetAttribute("Interval", TimeValue(Seconds(2.0)));
140     echoClient2.SetAttribute("PacketSize", UintegerValue(512));
141
142     ApplicationContainer clientApps2 = echoClient2.Install(wifiAdhocNodes.Get(4));
143     clientApps2.Start(Seconds(2.0));
144     clientApps2.Stop(Seconds(10.0));
145
146     Ipv4GlobalRoutingHelper::PopulateRoutingTables();
147
148     Simulator::Stop(Seconds(10.0));
149     if (tracing)
150     {
151         phy.SetPcapDataLinkType(WifiPhyHelper::DLT_IEEE802_11_RADIO);
152         phy.EnablePcap("Assignment2Task1", AdhocDevices.Get(1));
153     }
154     Simulator::Run();
155     Simulator::Destroy();
156     return 0;
157 }
```

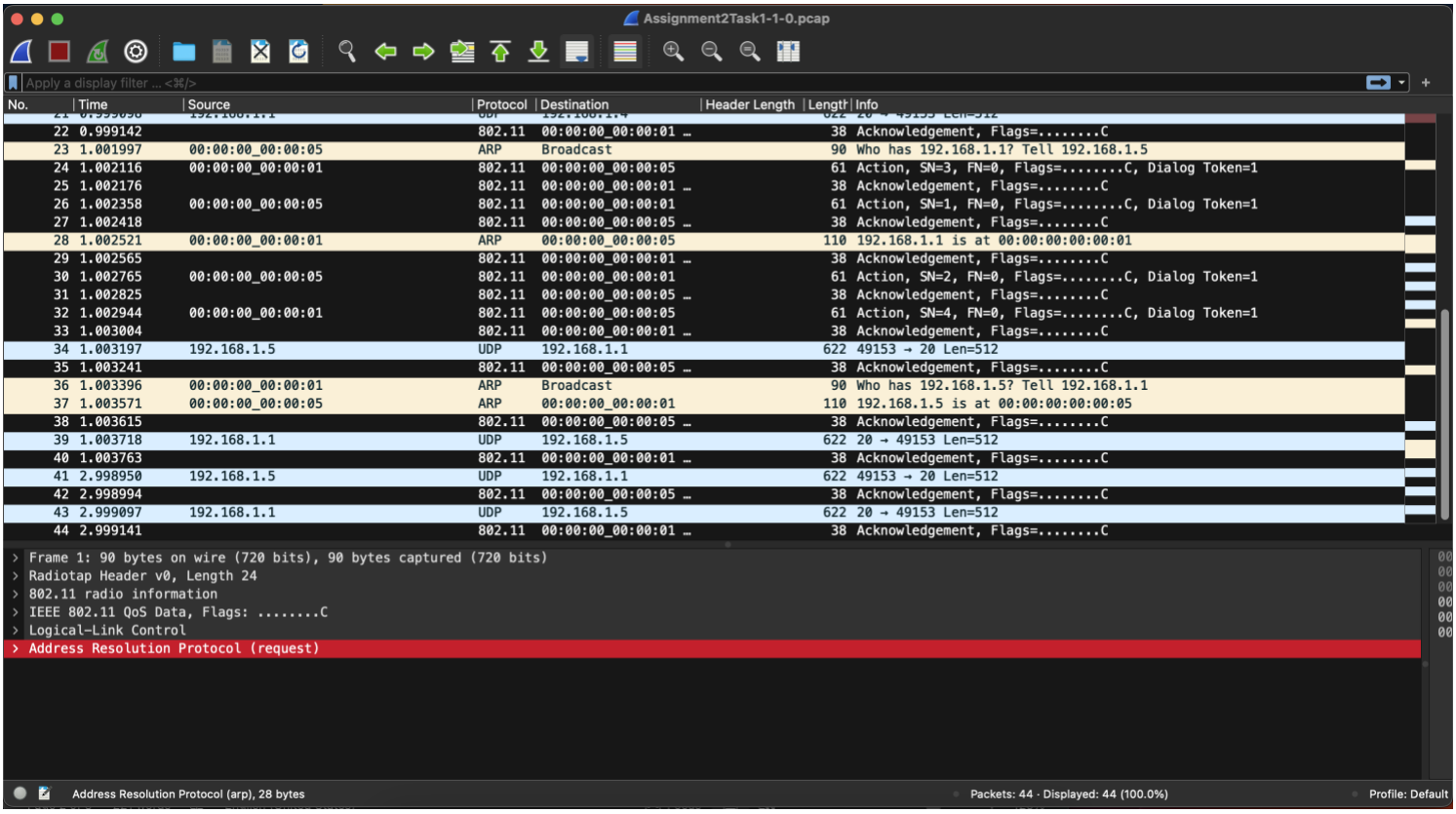
2. Results

Terminal Output: Here we can see that both the clients at port 3 with IP address 192.168.1.4 & port 4 with IP address 192.168.1.5 are communicating with the server which is port 0 with IP address 192.168.1.1.

```
ns-3.39 — -zsh — 80×24
soniya@Soniyas-MacBook-Air ns-3.39 % ./ns3 run scratch/Assig2Task1.cc
[0/2] Re-checking globbed directories...
ninja: no work to do.
At time +1s client sent 512 bytes to 192.168.1.1 port 20
At time +1.00291s server received 512 bytes from 192.168.1.4 port 49153
At time +1.00291s server sent 512 bytes to 192.168.1.4 port 49153
At time +1.01156s client received 512 bytes from 192.168.1.1 port 20
At time +2s client sent 512 bytes to 192.168.1.1 port 20
At time +2s client sent 512 bytes to 192.168.1.1 port 20
At time +2.00066s server received 512 bytes from 192.168.1.4 port 49153
At time +2.00066s server sent 512 bytes to 192.168.1.4 port 49153
At time +2.00126s client received 512 bytes from 192.168.1.1 port 20
At time +2.00493s server received 512 bytes from 192.168.1.5 port 49153
At time +2.00493s server sent 512 bytes to 192.168.1.5 port 49153
At time +2.00562s client received 512 bytes from 192.168.1.1 port 20
At time +4s client sent 512 bytes to 192.168.1.1 port 20
At time +4.00015s server received 512 bytes from 192.168.1.5 port 49153
At time +4.00015s server sent 512 bytes to 192.168.1.5 port 49153
At time +4.00039s client received 512 bytes from 192.168.1.1 port 20
soniya@Soniyas-MacBook-Air ns-3.39 %
```

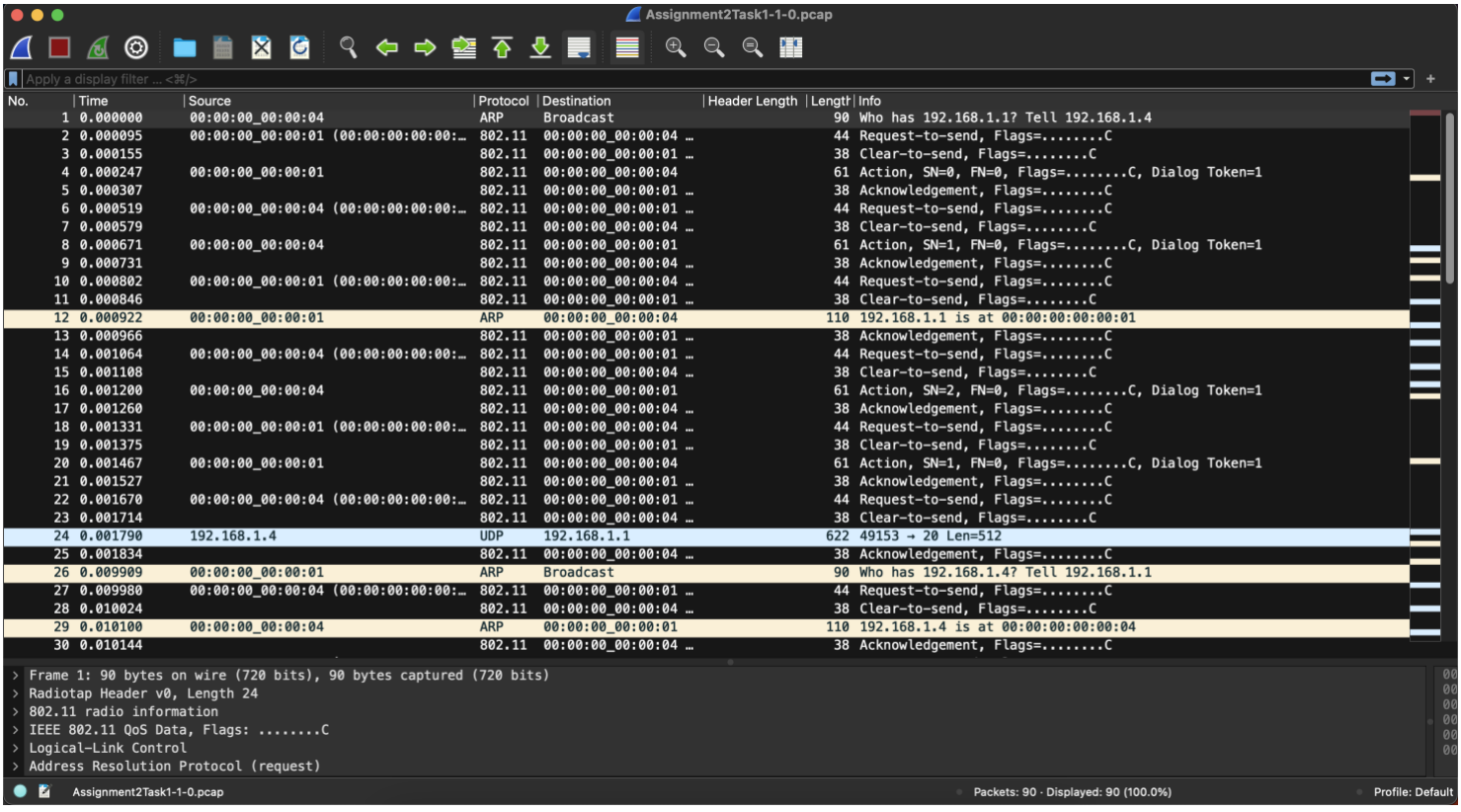
Without RTS/CTS

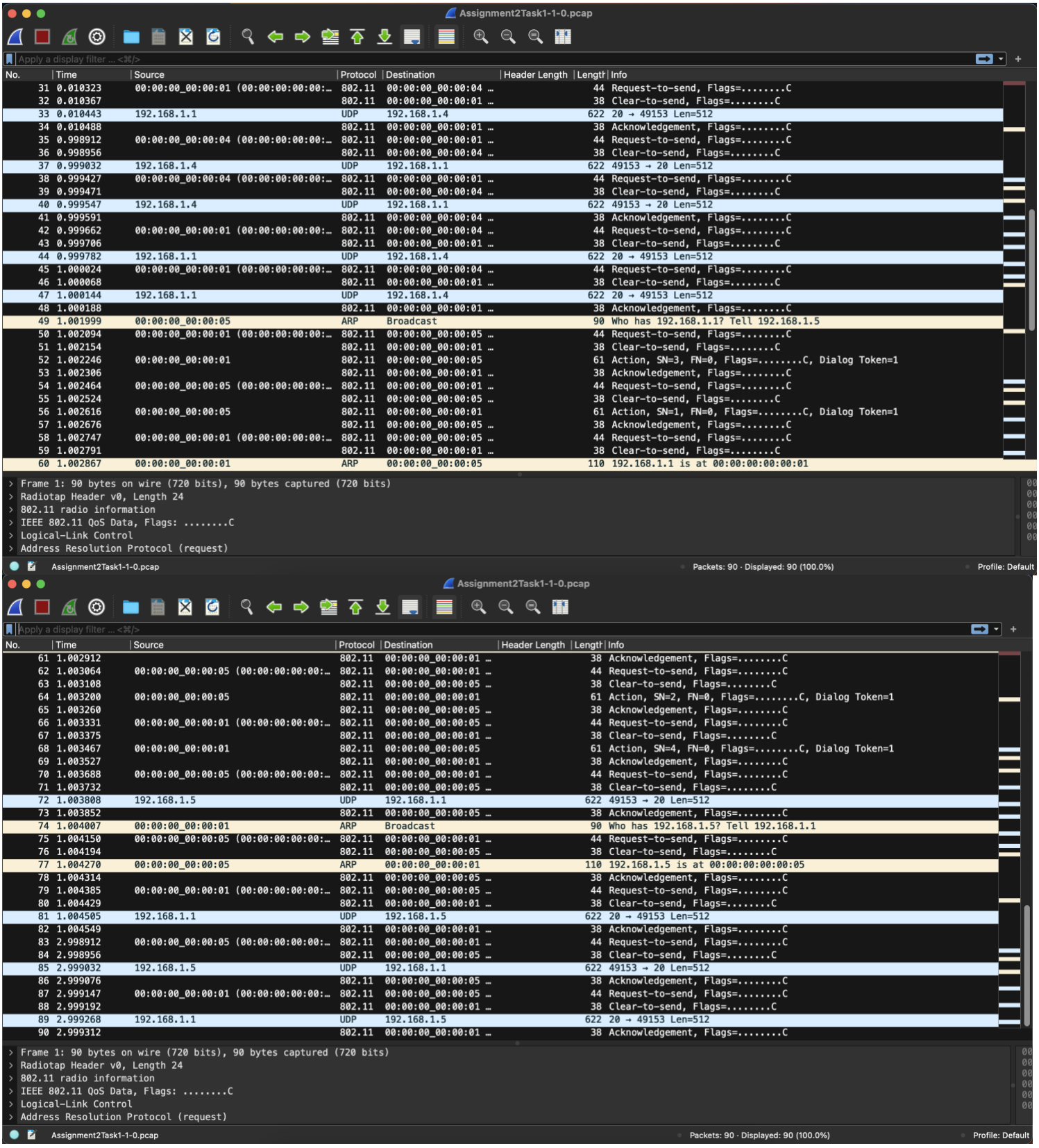




With RTS/CTS

After Adding - Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", UintegerValue (0));




Assignment2Task1-1-0.pcap
Apply a display filter ... <⌘/>
No. Time Source Protocol Destination Header Length Length Info
31 0.010323 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:04 ... 44 Request-to-send, Flags=........C
32 0.010367 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
33 0.010443 192.168.1.1 UDP 192.168.1.4 622 20 → 49153 Len=512
34 0.010488 802.11 00:00:00_00:00:01 ... 38 Acknowledgement, Flags=........C
35 0.998912 00:00:00_00:00:04 (00:00:00:00:00:... 802.11 00:00:00_00:00:01 ... 44 Request-to-send, Flags=........C
36 0.998956 802.11 00:00:00_00:00:04 ... 38 Clear-to-send, Flags=........C
37 0.999032 192.168.1.4 UDP 192.168.1.1 622 49153 → 20 Len=512
38 0.999427 00:00:00_00:00:04 (00:00:00:00:00:... 802.11 00:00:00_00:00:01 ... 44 Request-to-send, Flags=........C
39 0.999471 802.11 00:00:00_00:00:04 ... 38 Clear-to-send, Flags=........C
40 0.999547 192.168.1.4 UDP 192.168.1.1 622 49153 → 20 Len=512
41 0.999591 802.11 00:00:00_00:00:04 ... 38 Acknowledgement, Flags=........C
42 0.999662 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:04 ... 44 Request-to-send, Flags=........C
43 0.999706 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
44 0.999782 192.168.1.1 UDP 192.168.1.4 622 20 → 49153 Len=512
45 1.000024 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:04 ... 44 Request-to-send, Flags=........C
46 1.000068 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
47 1.000144 192.168.1.1 UDP 192.168.1.4 622 20 → 49153 Len=512
48 1.000188 802.11 00:00:00_00:00:01 ... 38 Acknowledgement, Flags=........C
49 1.001999 00:00:00_00:00:05 ARP Broadcast 90 Who has 192.168.1.1? Tell 192.168.1.5
50 1.002094 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:05 ... 44 Request-to-send, Flags=........C
51 1.002154 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
52 1.002246 00:00:00_00:00:01 802.11 00:00:00_00:00:05 61 Action, SN=3, FN=0, Flags=........C, Dialog Token=1
53 1.002306 802.11 00:00:00_00:00:01 ... 38 Acknowledgement, Flags=........C
54 1.002464 00:00:00_00:00:05 (00:00:00:00:00:... 802.11 00:00:00_00:00:01 ... 44 Request-to-send, Flags=........C
55 1.002524 802.11 00:00:00_00:00:05 ... 38 Clear-to-send, Flags=........C
56 1.002616 00:00:00_00:00:05 802.11 00:00:00_00:00:01 61 Action, SN=1, FN=0, Flags=........C, Dialog Token=1
57 1.002676 802.11 00:00:00_00:00:05 ... 38 Acknowledgement, Flags=........C
58 1.002747 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:05 ... 44 Request-to-send, Flags=........C
59 1.002791 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
60 1.002867 00:00:00_00:00:01 ARP 00:00:00_00:00:05 110 192.168.1.1 is at 00:00:00:00:00:01
> Frame 1: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
> Radiotap Header v0, Length 24
> 802.11 radio information
> IEEE 802.11 QoS Data, Flags: C
> Logical-Link Control
> Address Resolution Protocol (request)
Assignment2Task1-1-0.pcap
Packets: 90 · Displayed: 90 (100.0%)
Profile: Default

Assignment2Task1-1-0.pcap
Apply a display filter ... <⌘/>
No. Time Source Protocol Destination Header Length Length Info
61 1.002912 802.11 00:00:00_00:00:01 ... 38 Acknowledgement, Flags=........C
62 1.003064 00:00:00_00:00:05 (00:00:00:00:00:... 802.11 00:00:00_00:00:01 ... 44 Request-to-send, Flags=........C
63 1.003108 802.11 00:00:00_00:00:05 ... 38 Clear-to-send, Flags=........C
64 1.003200 00:00:00_00:00:05 802.11 00:00:00_00:00:01 61 Action, SN=2, FN=0, Flags=........C, Dialog Token=1
65 1.003260 802.11 00:00:00_00:00:05 ... 38 Acknowledgement, Flags=........C
66 1.003331 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:05 ... 44 Request-to-send, Flags=........C
67 1.003375 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
68 1.003467 00:00:00_00:00:01 802.11 00:00:00_00:00:05 61 Action, SN=4, FN=0, Flags=........C, Dialog Token=1
69 1.003527 802.11 00:00:00_00:00:01 ... 38 Acknowledgement, Flags=........C
70 1.003688 00:00:00_00:00:05 (00:00:00:00:00:... 802.11 00:00:00_00:00:01 ... 44 Request-to-send, Flags=........C
71 1.003732 802.11 00:00:00_00:00:05 ... 38 Clear-to-send, Flags=........C
72 1.003808 192.168.1.5 UDP 192.168.1.1 622 49153 → 20 Len=512
73 1.003852 802.11 00:00:00_00:00:05 ... 38 Acknowledgement, Flags=........C
74 1.004007 00:00:00_00:00:01 ARP Broadcast 90 Who has 192.168.1.5? Tell 192.168.1.1
75 1.004150 00:00:00_00:00:05 (00:00:00:00:00:... 802.11 00:00:00_00:00:01 ... 44 Request-to-send, Flags=........C
76 1.004194 802.11 00:00:00_00:00:05 ... 38 Clear-to-send, Flags=........C
77 1.004270 00:00:00_00:00:05 ARP 00:00:00_00:00:01 110 192.168.1.5 is at 00:00:00:00:00:05
78 1.004314 802.11 00:00:00_00:00:05 ... 38 Acknowledgement, Flags=........C
79 1.004385 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:05 ... 44 Request-to-send, Flags=........C
80 1.004429 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
81 1.004505 192.168.1.1 UDP 192.168.1.5 622 20 → 49153 Len=512
82 1.004549 802.11 00:00:00_00:00:01 ... 38 Acknowledgement, Flags=........C
83 2.998912 00:00:00_00:00:05 (00:00:00:00:00:... 802.11 00:00:00_00:00:01 ... 44 Request-to-send, Flags=........C
84 2.998956 802.11 00:00:00_00:00:05 ... 38 Clear-to-send, Flags=........C
85 2.999032 192.168.1.5 UDP 192.168.1.1 622 49153 → 20 Len=512
86 2.999076 802.11 00:00:00_00:00:05 ... 38 Acknowledgement, Flags=........C
87 2.999147 00:00:00_00:00:01 (00:00:00:00:00:... 802.11 00:00:00_00:00:05 ... 44 Request-to-send, Flags=........C
88 2.999192 802.11 00:00:00_00:00:01 ... 38 Clear-to-send, Flags=........C
89 2.999268 192.168.1.1 UDP 192.168.1.5 622 20 → 49153 Len=512
90 2.999312 802.11 00:00:00_00:00:01 ... 38 Acknowledgement, Flags=........C
> Frame 1: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
> Radiotap Header v0, Length 24
> 802.11 radio information
> IEEE 802.11 QoS Data, Flags: C
> Logical-Link Control
> Address Resolution Protocol (request)
Assignment2Task1-1-0.pcap
Packets: 90 · Displayed: 90 (100.0%)
Profile: Default

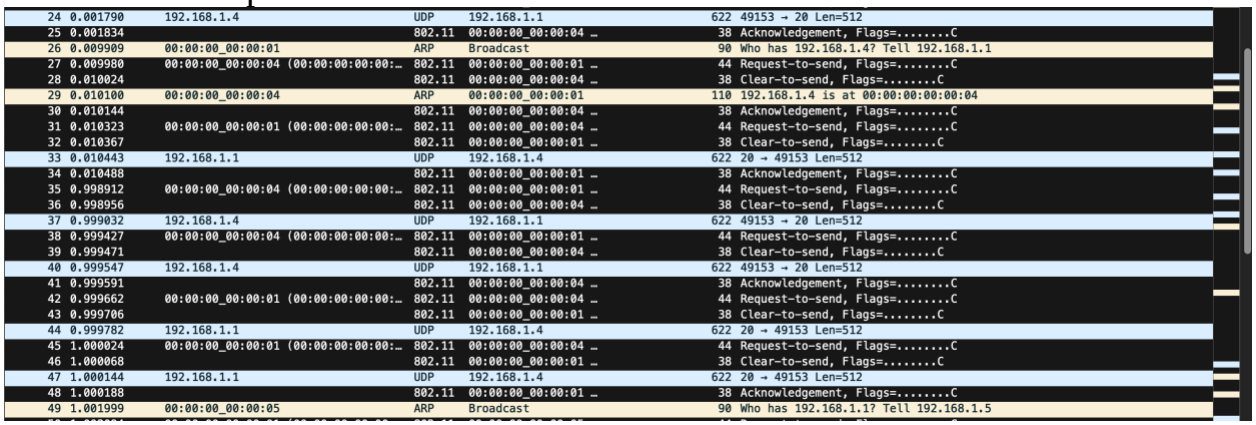
3. Learnt Lessons

The RTS/CTS handshake is designed to minimize collision risks in wireless networks by reserving the communication channel for the necessary duration of data transmission. Its significance becomes evident in scenarios where the hidden node problem may emerge. This issue arises when two stations, lacking direct awareness of each other's transmissions, concurrently attempt to send data to a shared third station, resulting in collisions.

4. Question Answers

1) **Are all the frames acknowledged? Explain why.**

Solution: Yes all the frames are acknowledged because we can see that there is a communication protocol followed where the frames need ack before transmission.



24	0.001790	192.168.1.4	UDP	192.168.1.1	622 49153 → 20 Len=512
25	0.001834		802.11	00:00:00_00:00:04 …	38 Acknowledgement, Flags=.......C
26	0.009909	00:00:00_00:00:01	ARP	Broadcast	90 Who has 192.168.1.4? Tell 192.168.1.1
27	0.009980	00:00:00_00:00:04 (00:00:00:00:00:…	802.11	00:00:00_00:00:01 …	44 Request-to-send, Flags=.......C
28	0.010024		802.11	00:00:00_00:00:04 …	38 Clear-to-send, Flags=.......C
29	0.010100	00:00:00_00:00:04	ARP	00:00:00_00:00:01	110 192.168.1.4 is at 00:00:00:00:00:04
30	0.010144		802.11	00:00:00_00:00:04 …	38 Acknowledgement, Flags=.......C
31	0.010323	00:00:00_00:00:01 (00:00:00:00:00:…	802.11	00:00:00_00:00:04 …	44 Request-to-send, Flags=.......C
32	0.010367		802.11	00:00:00_00:00:01 …	38 Clear-to-send, Flags=.......C
33	0.010443	192.168.1.1	UDP	192.168.1.4	622 20 → 49153 Len=512
34	0.010488		802.11	00:00:00_00:00:01 …	38 Acknowledgement, Flags=.......C
35	0.998912	00:00:00_00:00:04 (00:00:00:00:00:…	802.11	00:00:00_00:00:01 …	44 Request-to-send, Flags=.......C
36	0.998956		802.11	00:00:00_00:00:04 …	38 Clear-to-send, Flags=.......C
37	0.999032	192.168.1.4	UDP	192.168.1.1	622 49153 → 20 Len=512
38	0.999427	00:00:00_00:00:04 (00:00:00:00:00:…	802.11	00:00:00_00:00:01 …	44 Request-to-send, Flags=.......C
39	0.999471		802.11	00:00:00_00:00:04 …	38 Clear-to-send, Flags=.......C
40	0.999547	192.168.1.4	UDP	192.168.1.1	622 49153 → 20 Len=512
41	0.999591		802.11	00:00:00_00:00:04 …	38 Acknowledgement, Flags=.......C
42	0.999662	00:00:00_00:00:01 (00:00:00:00:00:…	802.11	00:00:00_00:00:04 …	44 Request-to-send, Flags=.......C
43	0.999706		802.11	00:00:00_00:00:01 …	38 Clear-to-send, Flags=.......C
44	0.999782	192.168.1.1	UDP	192.168.1.4	622 20 → 49153 Len=512
45	1.000024	00:00:00_00:00:01 (00:00:00:00:00:…	802.11	00:00:00_00:00:04 …	44 Request-to-send, Flags=.......C
46	1.000068		802.11	00:00:00_00:00:01 …	38 Clear-to-send, Flags=.......C
47	1.000144	192.168.1.1	UDP	192.168.1.4	622 20 → 49153 Len=512
48	1.000188		802.11	00:00:00_00:00:01 …	38 Acknowledgement, Flags=.......C
49	1.001999	00:00:00_00:00:05	ARP	Broadcast	90 Who has 192.168.1.1? Tell 192.168.1.5

2) **Are there any collisions in the network? Explain why. How have you reached this conclusion?**

Solution:

Without the RTS/CTS there is not packet collision. Packet collision should mean retransmission of packets and packet loss which is not indicated in the PCAP file. With the RTS/CTS there is not packet collision. Packet collision should mean retransmission of packets and packet loss which is not indicated in the PCAP file.

3) **How can the nodes be forced to utilize the RTS/CTS handshake procedure?**

Solution:

Command - Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", UintegerValue (0)); is used to force the utilization of RTS/CTS

4) **Force the utilization of RTS/CTS in the network:** ***TIP: In the ns-3 Doxygen, check the attributes of ns3::WifiRemoteStationManager. In the"Detailed Description", look for an attribute used for this purpose.***

- **Are there any collisions in data frames now?**

Solution: No there are no collision in frames visible.

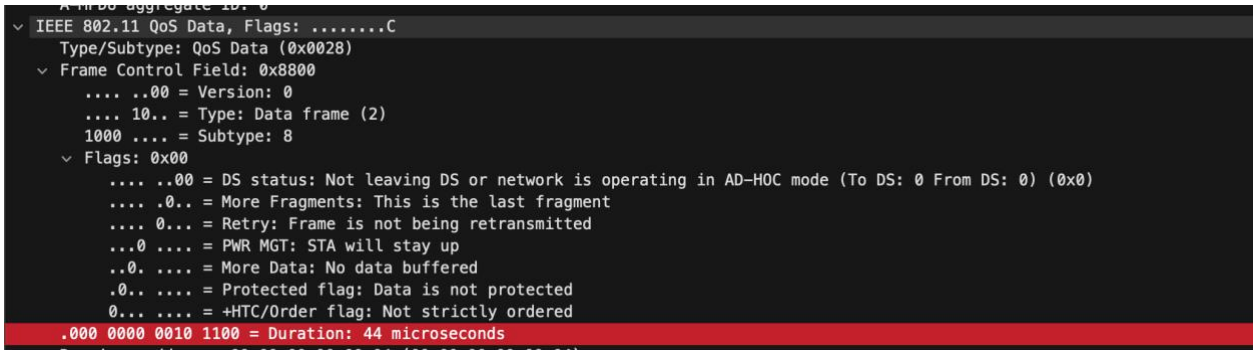
- **Which is the benefit or RTS/CTS? Briefly explain how RTS/CTS works.**

Solution:

RTS/CTS has a number of benefits like Reduced collisions, Improved performance, Increased fairness.

The MAC address of the transmitting node The MAC address of the receiving node The length of the data frame that the transmitting node wants to send The receiving node checks to see if it is ready to receive the data frame. If it is, the receiving node sends a CTS frame back to the transmitting node. The transmitting node receives the CTS frame and begins sending the data frame. The receiving node receives the data frame and sends an ACK frame back to the transmitting node. The RTS/CTS handshaking process ensures that only one node is transmitting at a time, which helps to reduce collisions and improve performance.

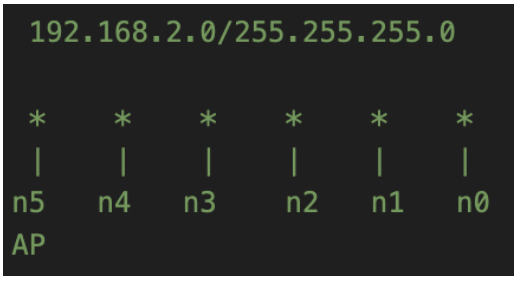
- **Where can you find the Network Allocation Vector information?**
  Solution: We can find it under IEEE 802.11 where duration is 44 microseconds.



Task 2

1. Experimental setup

In this lab task 1, we have made 5 Wi-Fi nodes and one Access Point.



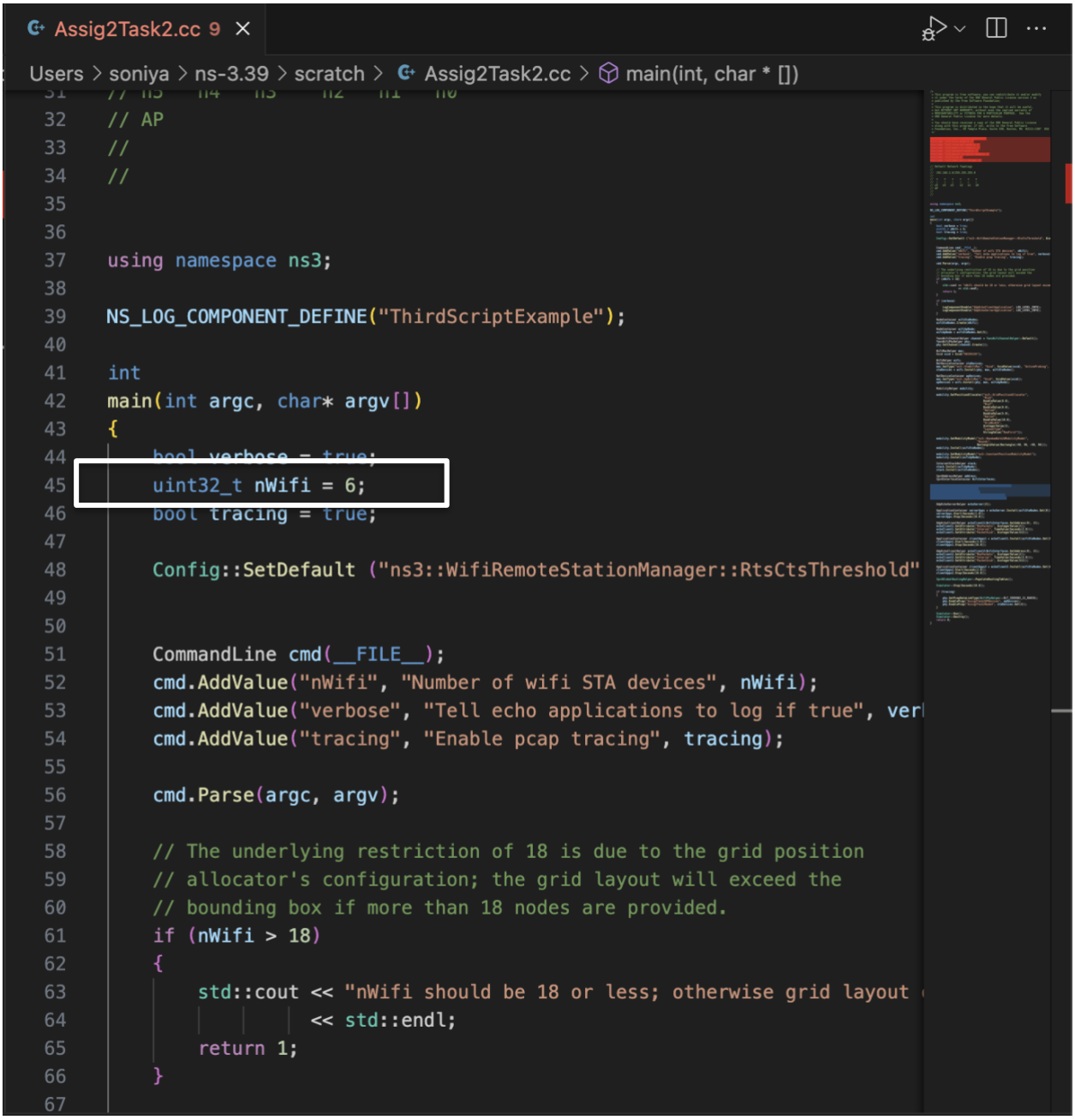
a) The IP addresses assigned for the network is 192.168.2.0 / 255.255.255.0. The nodes assign IP address according to node number. Here the node vise IP addresses are –

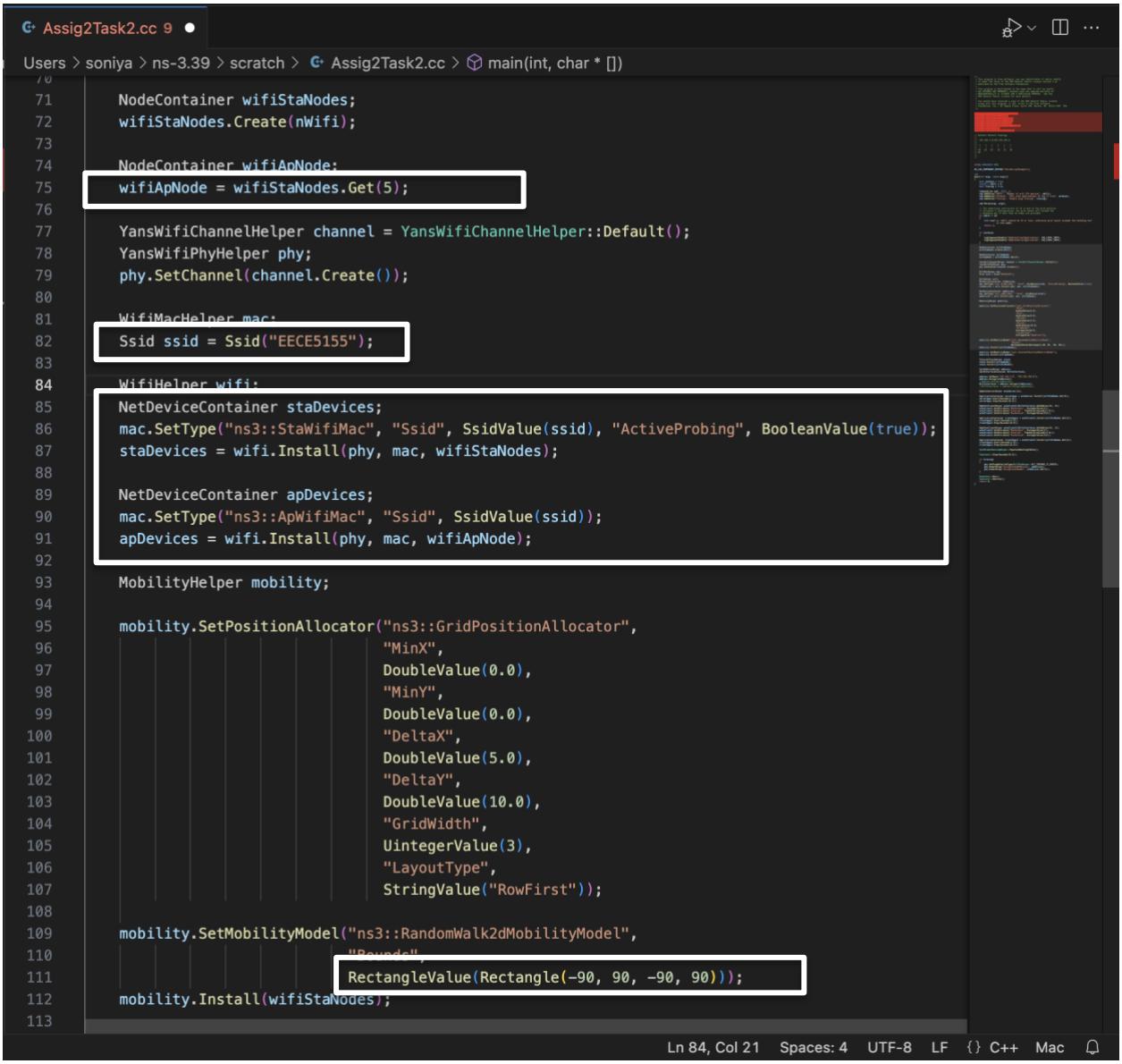
- 192.168.2.1 – Node 0
- 192.168.2.2 – Node 1
- 192.168.2.3 – Node 2
- 192.168.2.4 – Node 3
- 192.168.2.5 – Node 4
- 192.168.2.6 – Node 5 – Access Point

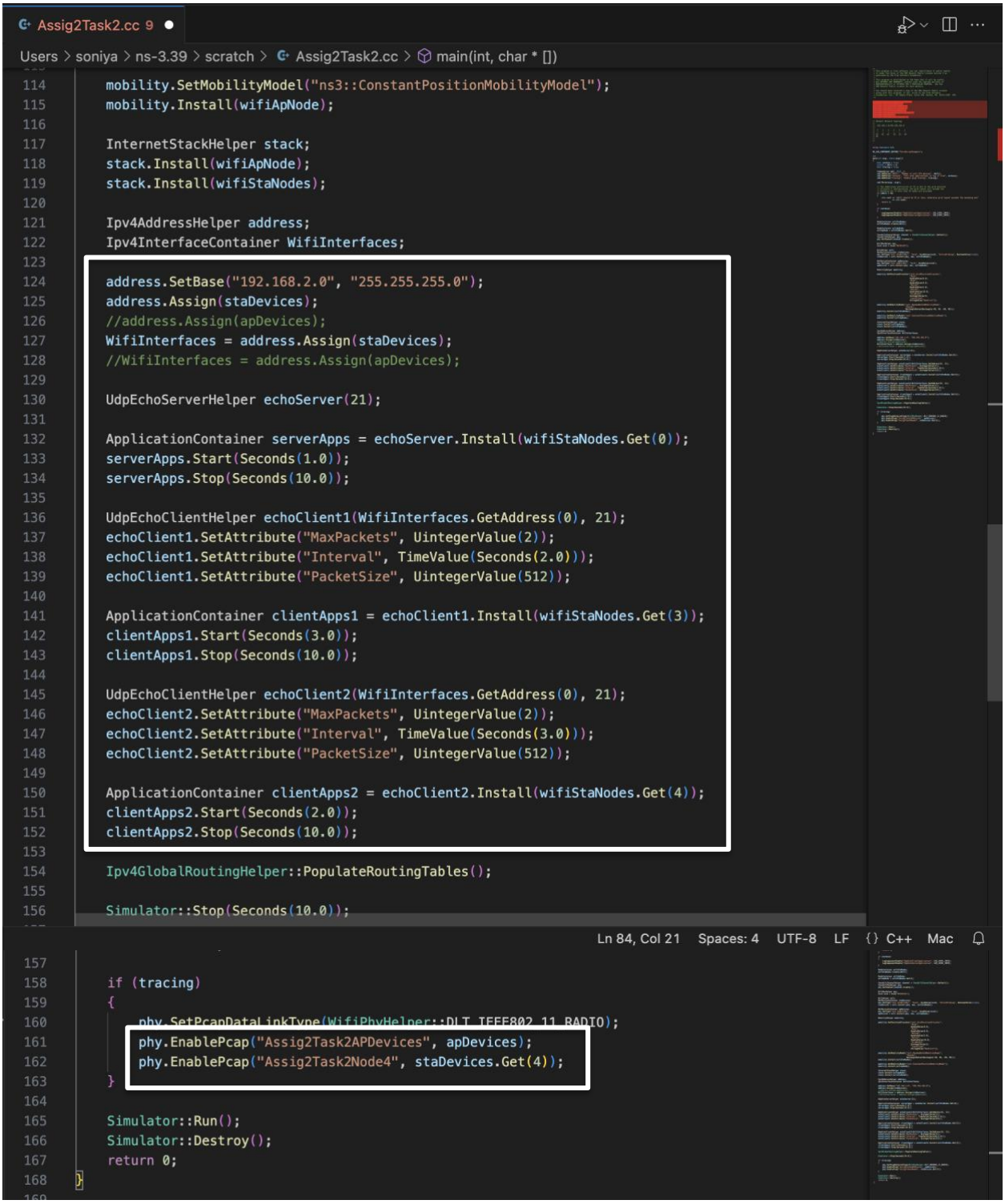
b) We have removed the Point-to-point connection and the CSMA link because we only require the 5 WIFI nodes and 1 Access Point.

c) Rather than assigning one node as Client we have 2 nodes (node 3 & node 4), sending data to one Server at node 0 at port number 21.
d) We are echoing the Server at 4 instances, 2 instances by Node 3 with IP 192.168.2.4 at 2 second and 5 second mark and the size of the packet is 512 bytes. 2 instances by Node 4 with IP 192.168.2.5 at 3 second and 5 second mark and the size of the packet is 512 bytes.
e) The dimensions of the rectangle have also changed to -90,90,-90,90
f) We are creating PCAP file at the Access point which is node 5 in the network and the Client at Node 4.
g) We have also changed the SSID.
h) We have forced RTC/CTS to the code and verified the output.

COMMENTS ARE ADDED IN THE CODE


Assig2Task2.cc 9
Users > soniya > ns-3.39 > scratch > Assig2Task2.cc > main(int, char * [])
// AP
//
//
using namespace ns3;
NS_LOG_COMPONENT_DEFINE("ThirdScriptExample");
int
main(int argc, char* argv[])
{
uint32_t nWifi = 6;
bool tracing = true;
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold"
CommandLine cmd(__FILE__);
cmd.AddValue("nWifi", "Number of wifi STA devices", nWifi);
cmd.AddValue("verbose", "Tell echo applications to log if true", ver
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
<< std::endl;
return 1;
}


Assig2Task2.cc 9
Users > soniya > ns-3.39 > scratch > Assig2Task2.cc > main(int, char * [])
NodeContainer wifiStaNodes;
wifiStaNodes.Create(nWifi);
NodeContainer wifiApNode;
wifiApNode = wifiStaNodes.Get(5);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy;
phy.SetChannel(channel.Create());
WifiMacHelper mac;
Ssid ssid = Ssid("EECE5155");
WifiHelper wifi;
NetDeviceContainer staDevices;
mac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssid), "ActiveProbing", BooleanValue(true));
staDevices = wifi.Install(phy, mac, wifiStaNodes);
NetDeviceContainer apDevices;
mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid));
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",
RectangleValue(Rectangle(-90, 90, -90, 90)));
mobility.Install(wifiStaNodes);
Ln 84, Col 21
Spaces: 4
UTF-8
LF
C++
Mac

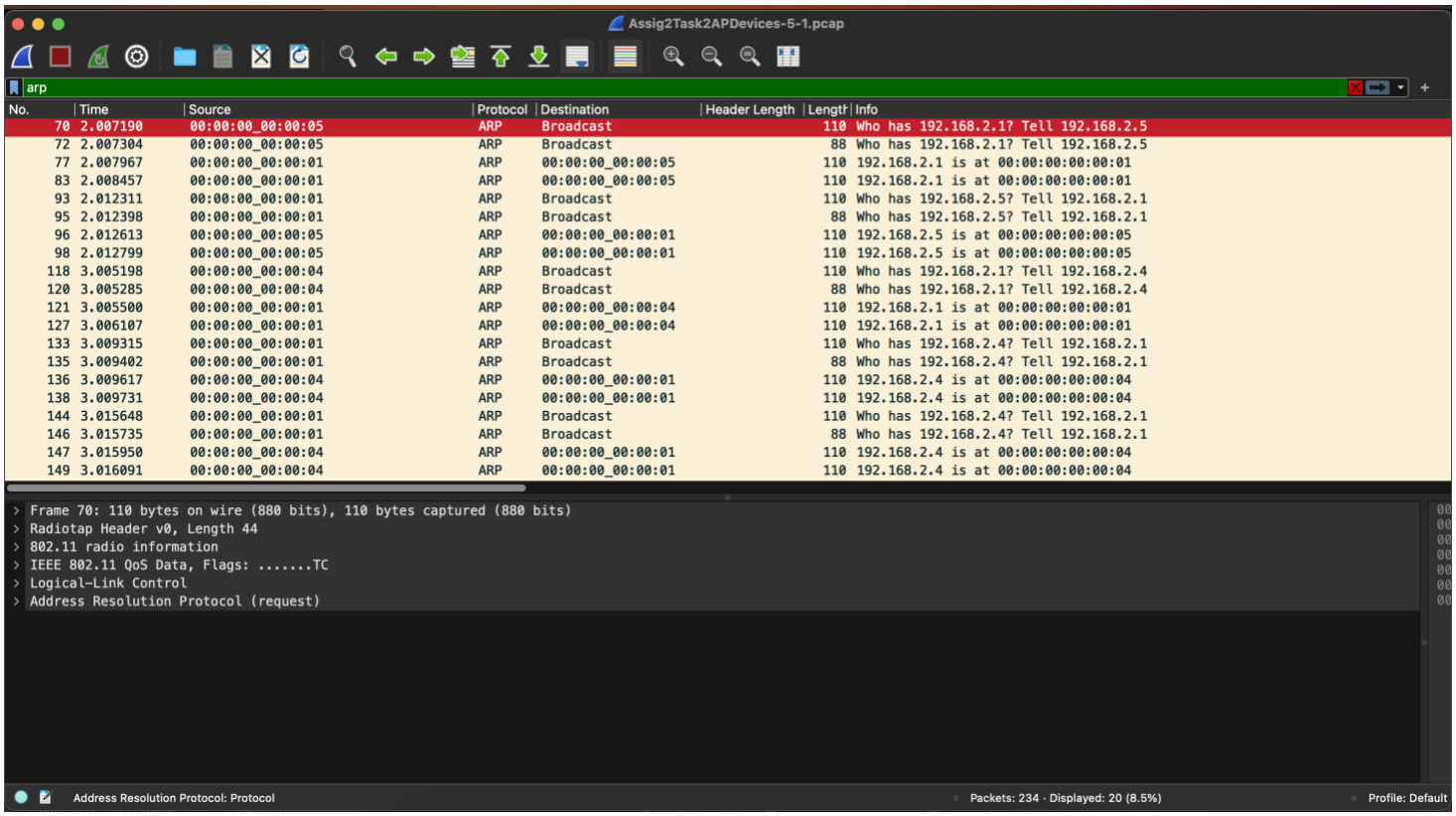

Assig2Task2.cc 9
Users > soniya > ns-3.39 > scratch > Assig2Task2.cc > main(int, char * [])
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiApNode);
InternetStackHelper stack;
stack.Install(wifiApNode);
stack.Install(wifiStaNodes);
Ipv4AddressHelper address;
Ipv4InterfaceContainer WifiInterfaces;
address.SetBase("192.168.2.0", "255.255.255.0");
address.Assign(staDevices);
//address.Assign(apDevices);
WifiInterfaces = address.Assign(staDevices);
//WifiInterfaces = address.Assign(apDevices);
UdpEchoServerHelper echoServer(21);
ApplicationContainer serverApps = echoServer.Install(wifiStaNodes.Get(0));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));
UdpEchoClientHelper echoClient1(WifiInterfaces.GetAddress(0), 21);
echoClient1.SetAttribute("MaxPackets", UintegerValue(2));
echoClient1.SetAttribute("Interval", TimeValue(Seconds(2.0)));
echoClient1.SetAttribute("PacketSize", UintegerValue(512));
ApplicationContainer clientApps1 = echoClient1.Install(wifiStaNodes.Get(3));
clientApps1.Start(Seconds(3.0));
clientApps1.Stop(Seconds(10.0));
UdpEchoClientHelper echoClient2(WifiInterfaces.GetAddress(0), 21);
echoClient2.SetAttribute("MaxPackets", UintegerValue(2));
echoClient2.SetAttribute("Interval", TimeValue(Seconds(3.0)));
echoClient2.SetAttribute("PacketSize", UintegerValue(512));
ApplicationContainer clientApps2 = echoClient2.Install(wifiStaNodes.Get(4));
clientApps2.Start(Seconds(2.0));
clientApps2.Stop(Seconds(10.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
Simulator::Stop(Seconds(10.0));
Ln 84, Col 21 Spaces: 4 UTF-8 LF C++ Mac
if (tracing)
{
phy.SetPcapDataLinkType(WifiPhyHelper::DLT_IEEE802_11_RADIO);
phy.EnablePcap("Assig2Task2APDevices", apDevices);
phy.EnablePcap("Assig2Task2Node4", staDevices.Get(4));
}
Simulator::Run();
Simulator::Destroy();
return 0;
}

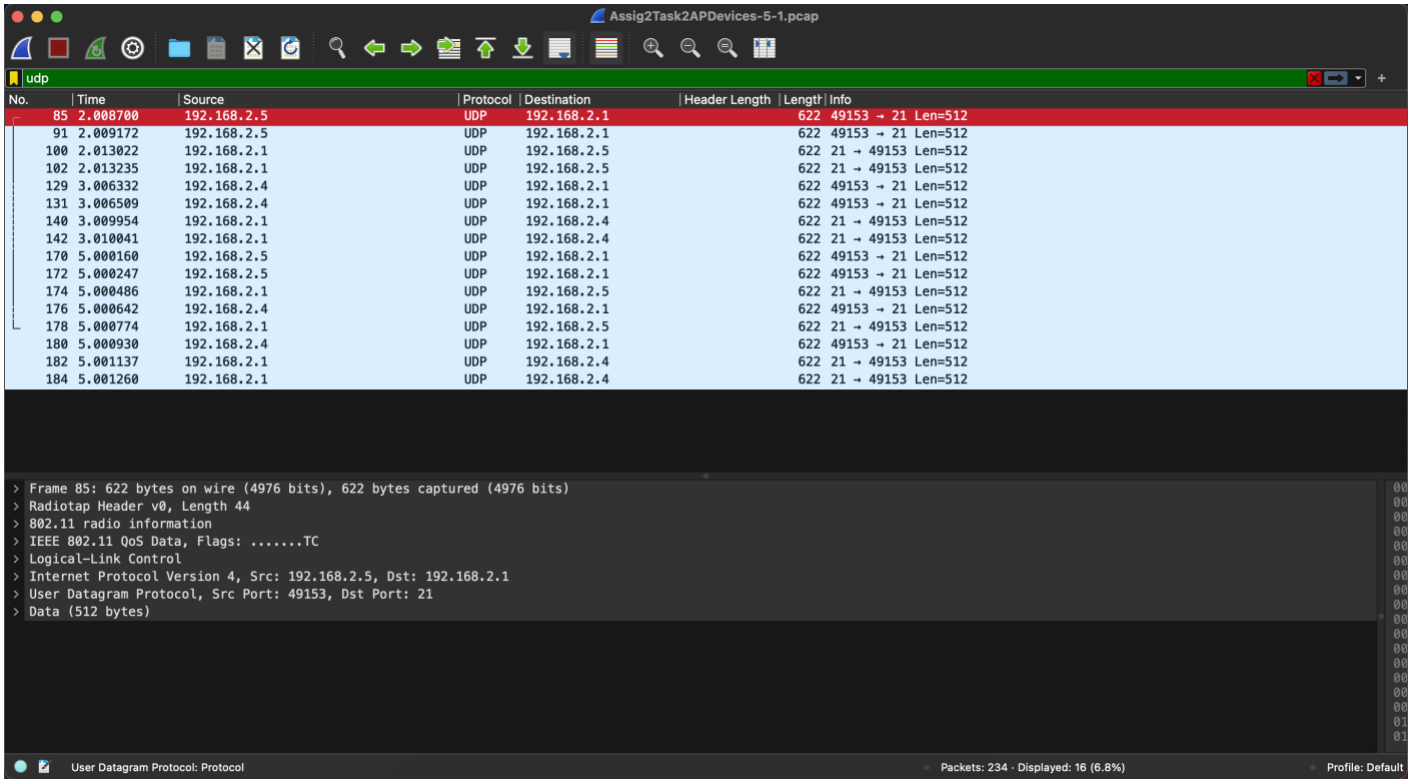
2. Results

```
ns-3.39 — -zsh — 80×24
soniya@Soniyas-MacBook-Air ns-3.39 % ./ns3 run scratch/Assig2Task2.cc
[0/2] Re-checking globbed directories...
ninja: no work to do.
At time +2s client sent 512 bytes to 192.168.2.1 port 21
At time +2.00951s server received 512 bytes from 192.168.2.5 port 49153
At time +2.00951s server sent 512 bytes to 192.168.2.5 port 49153
At time +2.01356s client received 512 bytes from 192.168.2.1 port 21
At time +3s client sent 512 bytes to 192.168.2.1 port 21
At time +3.00685s server received 512 bytes from 192.168.2.4 port 49153
At time +3.00685s server sent 512 bytes to 192.168.2.4 port 49153
At time +3.01037s client received 512 bytes from 192.168.2.1 port 21
At time +5s client sent 512 bytes to 192.168.2.1 port 21
At time +5s client sent 512 bytes to 192.168.2.1 port 21
At time +5.00059s server received 512 bytes from 192.168.2.5 port 49153
At time +5.00059s server sent 512 bytes to 192.168.2.5 port 49153
At time +5.0011s client received 512 bytes from 192.168.2.1 port 21
At time +5.00126s server received 512 bytes from 192.168.2.4 port 49153
At time +5.00126s server sent 512 bytes to 192.168.2.4 port 49153
At time +5.00159s client received 512 bytes from 192.168.2.1 port 21
soniya@Soniyas-MacBook-Air ns-3.39 %
```

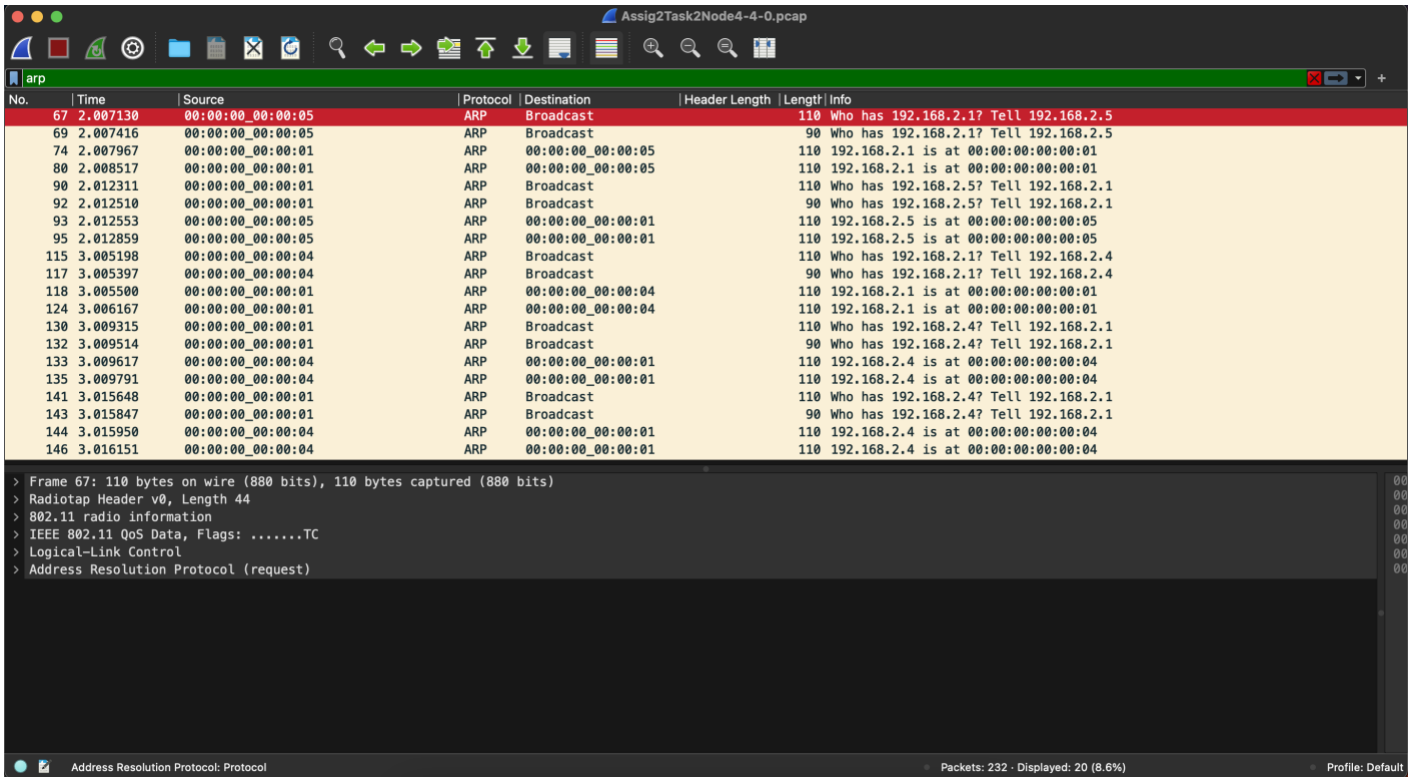
Without RTS/CTS

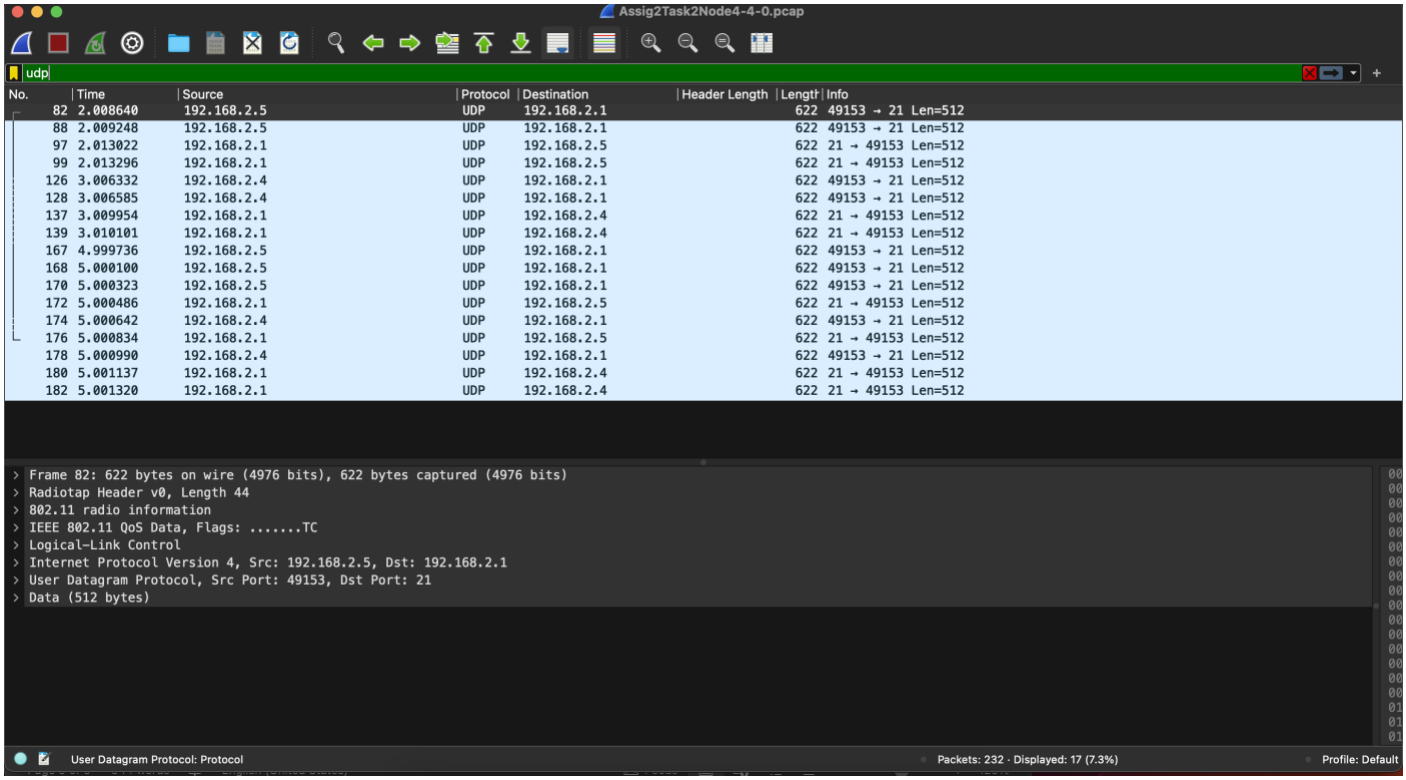
AP Node




Assig2Task2APDevices-5-1.pcap
udp
No. Time Source Protocol Destination Header Length Length Info
85 2.008700 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
91 2.009172 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
100 2.013022 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
102 2.013235 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
129 3.006332 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
131 3.006509 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
140 3.009954 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
142 3.010041 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
170 5.000160 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
172 5.000247 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
174 5.000486 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
176 5.000642 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
178 5.000774 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
180 5.000930 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
182 5.001137 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
184 5.001260 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
Frame 85: 622 bytes on wire (4976 bits), 622 bytes captured (4976 bits)
Radiotap Header v0, Length 44
802.11 radio information
IEEE 802.11 QoS Data, Flags: ....TC
Logical-Link Control
Internet Protocol Version 4, Src: 192.168.2.5, Dst: 192.168.2.1
User Datagram Protocol, Src Port: 49153, Dst Port: 21
Data (512 bytes)
User Datagram Protocol: Protocol
Packets: 234 · Displayed: 16 (6.8%)
Profile: Default

Node 4

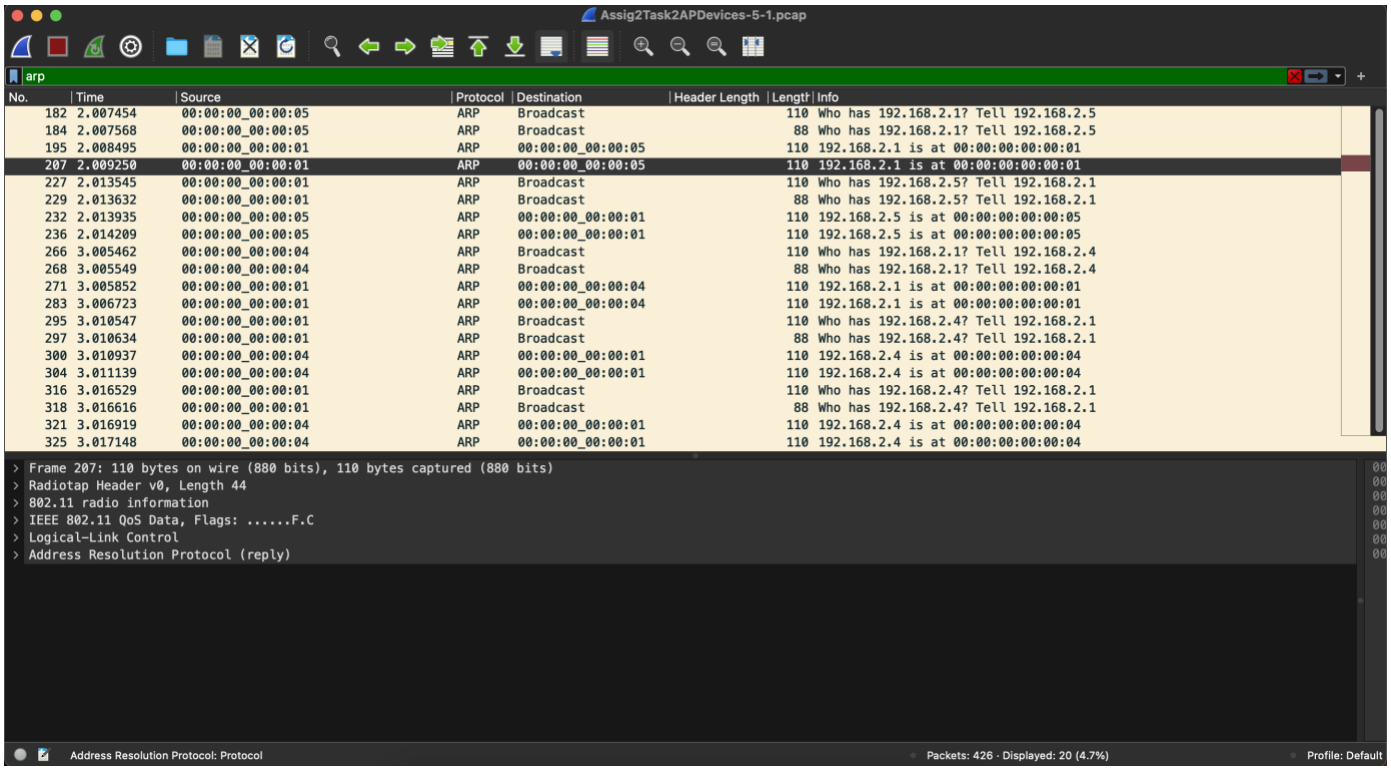

Assig2Task2Node4-4-0.pcap
arp
No. Time Source Protocol Destination Header Length Length Info
67 2.007130 00:00:00_00:00:05 ARP Broadcast 110 Who has 192.168.2.1? Tell 192.168.2.5
69 2.007416 00:00:00_00:00:05 ARP Broadcast 90 Who has 192.168.2.1? Tell 192.168.2.5
74 2.007967 00:00:00_00:00:01 ARP 00:00:00_00:00:05 110 192.168.2.1 is at 00:00:00:00:00:01
80 2.008517 00:00:00_00:00:01 ARP 00:00:00_00:00:05 110 192.168.2.1 is at 00:00:00:00:00:01
90 2.012311 00:00:00_00:00:01 ARP Broadcast 110 Who has 192.168.2.5? Tell 192.168.2.1
92 2.012510 00:00:00_00:00:01 ARP Broadcast 90 Who has 192.168.2.5? Tell 192.168.2.1
93 2.012553 00:00:00_00:00:05 ARP 00:00:00_00:00:01 110 192.168.2.5 is at 00:00:00:00:00:05
95 2.012859 00:00:00_00:00:05 ARP 00:00:00_00:00:01 110 192.168.2.5 is at 00:00:00:00:00:05
115 3.005198 00:00:00_00:00:04 ARP Broadcast 110 Who has 192.168.2.1? Tell 192.168.2.4
117 3.005397 00:00:00_00:00:04 ARP Broadcast 90 Who has 192.168.2.1? Tell 192.168.2.4
118 3.005500 00:00:00_00:00:01 ARP 00:00:00_00:00:04 110 192.168.2.1 is at 00:00:00:00:00:01
124 3.006167 00:00:00_00:00:01 ARP 00:00:00_00:00:04 110 192.168.2.1 is at 00:00:00:00:00:01
130 3.009315 00:00:00_00:00:01 ARP Broadcast 110 Who has 192.168.2.4? Tell 192.168.2.1
132 3.009514 00:00:00_00:00:01 ARP Broadcast 90 Who has 192.168.2.4? Tell 192.168.2.1
133 3.009617 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
135 3.009791 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
141 3.015648 00:00:00_00:00:01 ARP Broadcast 110 Who has 192.168.2.4? Tell 192.168.2.1
143 3.015847 00:00:00_00:00:01 ARP Broadcast 90 Who has 192.168.2.4? Tell 192.168.2.1
144 3.015950 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
146 3.016151 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
Frame 67: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
Radiotap Header v0, Length 44
802.11 radio information
IEEE 802.11 QoS Data, Flags: ....TC
Logical-Link Control
Address Resolution Protocol (request)
Address Resolution Protocol: Protocol
Packets: 232 · Displayed: 20 (8.6%)
Profile: Default

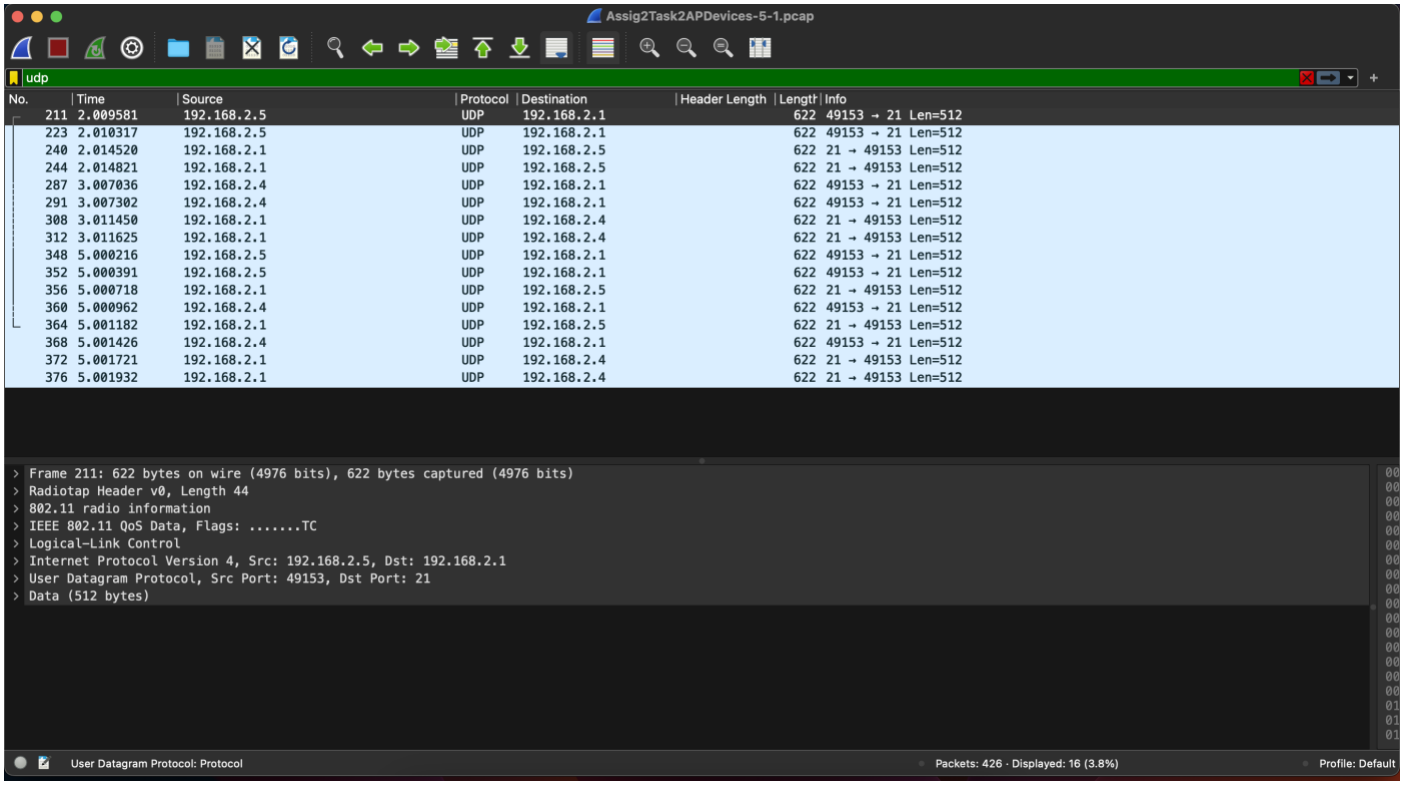


With RTS/CTS

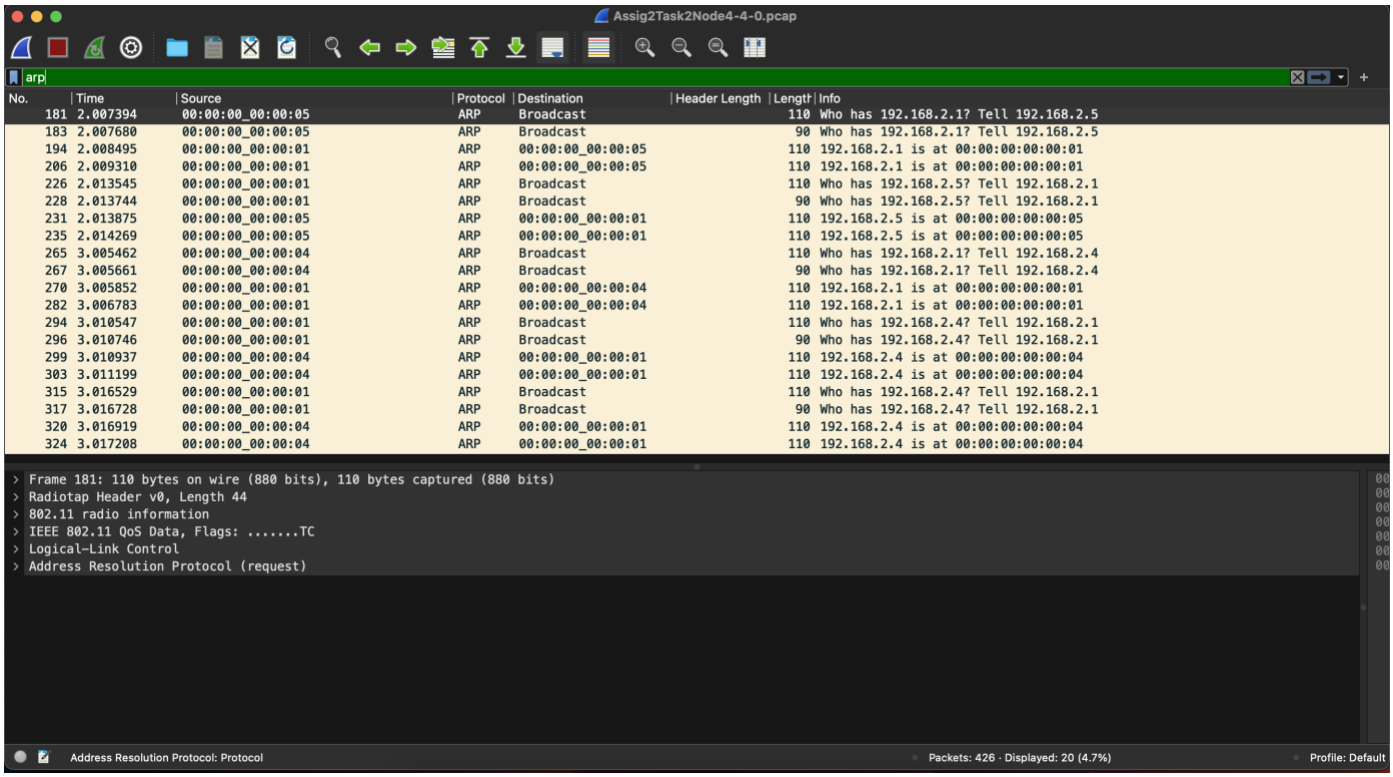
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", UintegerValue (0));

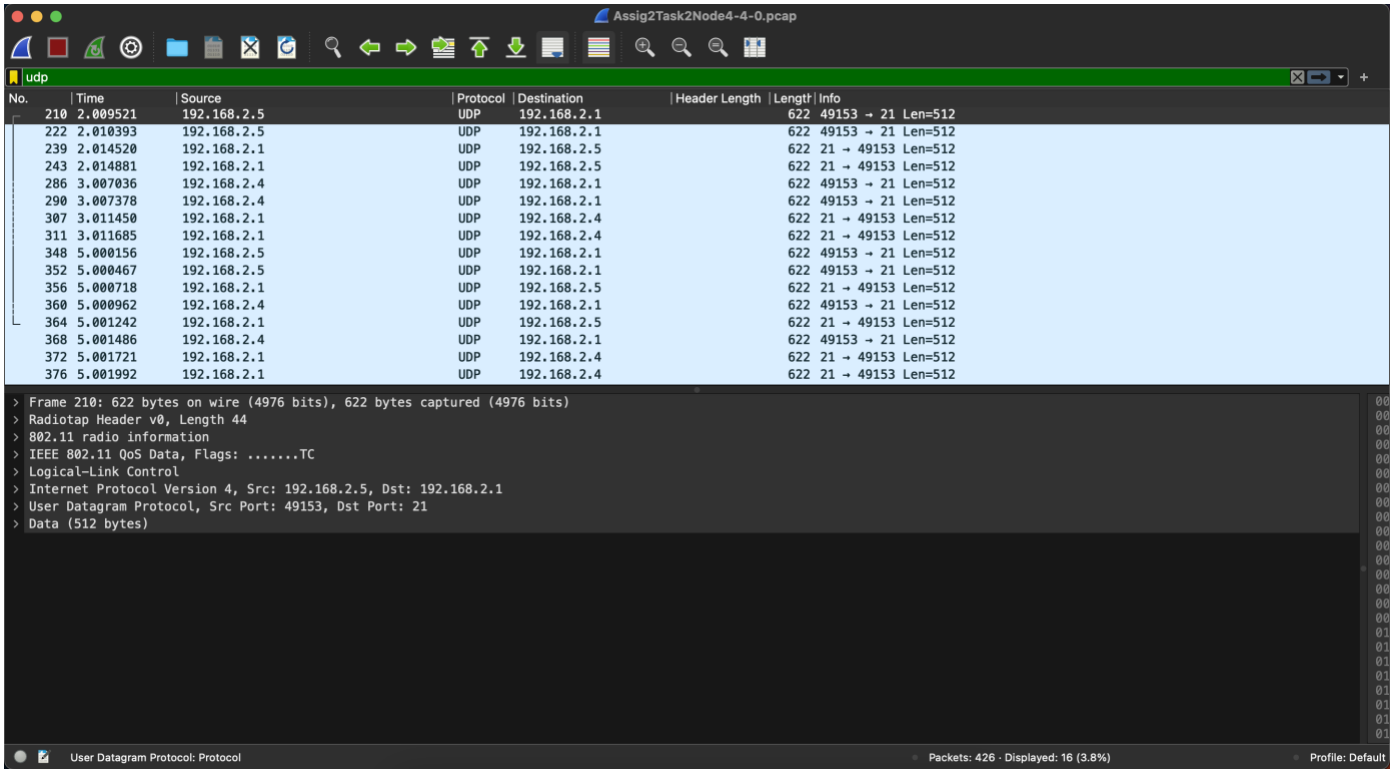
AP Node




Assig2Task2APDevices-5-1.pcap
udp
No. Time Source Protocol Destination Header Length Lengtl Info
211 2.009581 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
223 2.010317 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
240 2.014520 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
244 2.014821 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
287 3.007036 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
291 3.007302 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
308 3.011450 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
312 3.011625 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
348 5.000216 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
352 5.000391 192.168.2.5 UDP 192.168.2.1 622 49153 → 21 Len=512
356 5.000718 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
360 5.000962 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
364 5.001182 192.168.2.1 UDP 192.168.2.5 622 21 → 49153 Len=512
368 5.001426 192.168.2.4 UDP 192.168.2.1 622 49153 → 21 Len=512
372 5.001721 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
376 5.001932 192.168.2.1 UDP 192.168.2.4 622 21 → 49153 Len=512
Frame 211: 622 bytes on wire (4976 bits), 622 bytes captured (4976 bits)
Radiotap Header v0, Length 44
802.11 radio information
IEEE 802.11 QoS Data, Flags: ....TC
Logical-Link Control
Internet Protocol Version 4, Src: 192.168.2.5, Dst: 192.168.2.1
User Datagram Protocol, Src Port: 49153, Dst Port: 21
Data (512 bytes)
User Datagram Protocol: Protocol
Packets: 426 · Displayed: 16 (3.8%)
Profile: Default

Node 4


Assig2Task2Node4-4-0.pcap
arp
No. Time Source Protocol Destination Header Length Lengtl Info
181 2.007394 00:00:00_00:00:05 ARP Broadcast 110 Who has 192.168.2.1? Tell 192.168.2.5
183 2.007680 00:00:00_00:00:05 ARP Broadcast 90 Who has 192.168.2.1? Tell 192.168.2.5
194 2.008495 00:00:00_00:00:01 ARP 00:00:00_00:00:05 110 192.168.2.1 is at 00:00:00:00:00:01
206 2.009310 00:00:00_00:00:01 ARP 00:00:00_00:00:05 110 192.168.2.1 is at 00:00:00:00:00:01
226 2.013545 00:00:00_00:00:01 ARP Broadcast 110 Who has 192.168.2.5? Tell 192.168.2.1
228 2.013744 00:00:00_00:00:01 ARP Broadcast 90 Who has 192.168.2.5? Tell 192.168.2.1
231 2.013875 00:00:00_00:00:05 ARP 00:00:00_00:00:01 110 192.168.2.5 is at 00:00:00:00:00:05
235 2.014269 00:00:00_00:00:05 ARP 00:00:00_00:00:01 110 192.168.2.5 is at 00:00:00:00:00:05
265 3.005462 00:00:00_00:00:04 ARP Broadcast 110 Who has 192.168.2.1? Tell 192.168.2.4
267 3.005661 00:00:00_00:00:04 ARP Broadcast 90 Who has 192.168.2.1? Tell 192.168.2.4
270 3.005852 00:00:00_00:00:01 ARP 00:00:00_00:00:04 110 192.168.2.1 is at 00:00:00:00:00:01
282 3.006783 00:00:00_00:00:01 ARP 00:00:00_00:00:04 110 192.168.2.1 is at 00:00:00:00:00:01
294 3.010547 00:00:00_00:00:01 ARP Broadcast 110 Who has 192.168.2.4? Tell 192.168.2.1
296 3.010746 00:00:00_00:00:01 ARP Broadcast 90 Who has 192.168.2.4? Tell 192.168.2.1
299 3.010937 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
303 3.011199 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
315 3.016529 00:00:00_00:00:01 ARP Broadcast 110 Who has 192.168.2.4? Tell 192.168.2.1
317 3.016728 00:00:00_00:00:01 ARP Broadcast 90 Who has 192.168.2.4? Tell 192.168.2.1
320 3.016919 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
324 3.017208 00:00:00_00:00:04 ARP 00:00:00_00:00:01 110 192.168.2.4 is at 00:00:00:00:00:04
Frame 181: 110 bytes on wire (880 bits), 110 bytes captured (880 bits)
Radiotap Header v0, Length 44
802.11 radio information
IEEE 802.11 QoS Data, Flags: ....TC
Logical-Link Control
Address Resolution Protocol (request)
Address Resolution Protocol: Protocol
Packets: 426 · Displayed: 20 (4.7%)
Profile: Default



3. Learnt Lessons

An access point (AP) is a networking device that connects devices to a wired network using Wi-Fi or other wireless technologies.

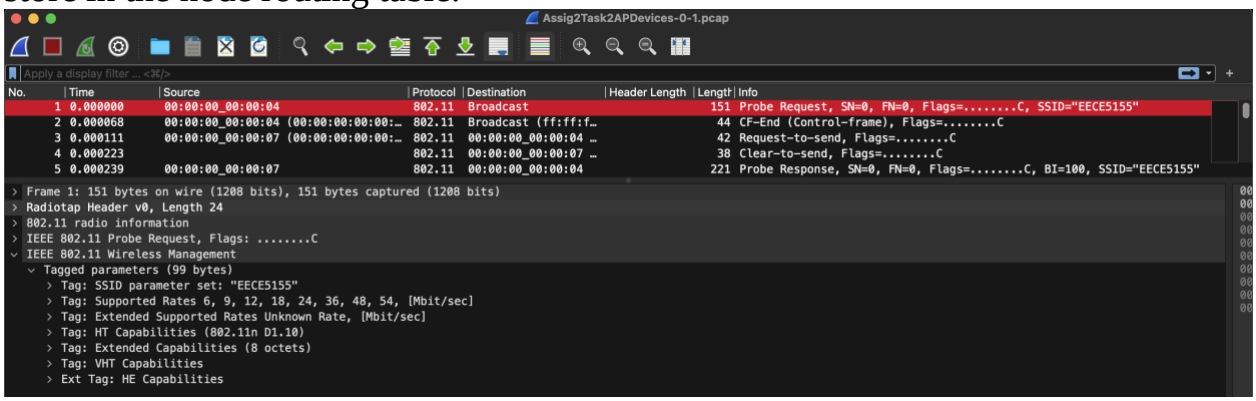
4. Question Answers

1) **Explain the behaviour of the AP. What is happening since the very first moment the network starts operating?**

   Solution: Multiple broadcast commands are seen, this is because the AP node is sending and receiving data about node information to all the nodes with the same SSID.

2) **Take a look to a beacon frame. Which are the most relevant parameters defined in it?**

   Solution: Most relevant paraments is the support rates the SSID the capabilities . These packets also send information about the capabilities of the other nodes to store in the node routing table.



3) **Are there any collisions in the network? When are these collisions happening?**
   Solution:
   No, collisions are visible in the packet. The connection between the server and the client is completed without any issues.
4) **As in Task 1, force the utilization of the RTS/CTS handshaking process and repeat the simulation. Are there any collisions in data frames now? Explain why.**
   Solution: No, collisions are visible in the packet even after the utilization of the RTS/CTS handshake process.