**Assignment 02\_Task 01**

**Task 1:**

**Experimental Setup:**

Used **Third.cc** as reference and modified following lines to complete task1.

Line 30: added this code. This code just pulls the position information from the mobility model and unconditionally logs the x and y position of the node

void

CourseChange (std::string context, Ptr<const MobilityModel> model)

{

Vector position = model->GetPosition ();

NS\_LOG\_UNCOND (context <<

" x = " << position.x << ", y = " << position.y);

}

Line 42: changed the number of nodes from 3 to 6

uint32\_t nWifi = 6;

Line 44: added useRts declaration

bool useRts = true;

Line 70: added the code to force the nodes to utilize the RTS/CTS handshake procedure

  if (useRts)

    {

    Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold",

    StringValue ("0"));

    }

Line 76: Added code for wifi nodes creation.

 NodeContainer wifiNodes;

 wifiNodes.Create(nWifi);

Line 81: Setting default parameters in IEEE 802.11G standard

WifiHelper wifi;

  wifi.SetStandard (WIFI\_STANDARD\_80211g);

Line 92: changed the station mode (sta) to Ad-hoc mode

mac.SetType ("ns3::AdhocWifiMac",

               "Ssid", SsidValue (ssid));

Line 115: changed the dimensions of lower-left corner and upper-right corner from 50 to 100m as mentioned

mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",

                             "Bounds", RectangleValue (Rectangle (-100, 100, -100, 100)));

Line 128: Assigned the Ip address as 192.168.1.0

address.SetBase ("192.168.1.0", "255.255.255.0");

Line 132-135: changed the UDP Echo Server port to port 20 and added server node information.

  UdpEchoServerHelper echoServer (20);

ApplicationContainer serverApps = echoServer.Install(wifiNodes.Get(0));

serverApps.Start(Seconds(1.0));

   serverApps.Stop(Seconds(10.0));

Line 138-145: Added client 1 code at node 5

 UdpEchoClientHelper echoClient(wifiInterfaces.GetAddress(0), 20);

    echoClient.SetAttribute("MaxPackets", UintegerValue(2));

    echoClient.SetAttribute("Interval", TimeValue(Seconds(2.0)));

    echoClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApps = echoClient.Install(wifiNodes.Get(5));

    clientApps.Start(Seconds(2.0));

    clientApps.Stop(Seconds(10.0));

Line 148-155: Added client 2 code at node 4

 UdpEchoClientHelper echoClient2(wifiInterfaces.GetAddress(0), 20);

    echoClient2.SetAttribute("MaxPackets", UintegerValue(2));

    echoClient2.SetAttribute("Interval", TimeValue(Seconds(1.0)));

    echoClient2.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApps2 = echoClient2.Install(wifiNodes.Get(4));

    clientApps2.Start(Seconds(3.0));

    clientApps2.Stop(Seconds(10.0));

Line 159-164: Added code for packet tracing at Node 2.

phy.SetPcapDataLinkType (WifiPhyHelper::DLT\_IEEE802\_11\_RADIO);

phy.EnablePcap (useRts ? "rtscts-AT21" : "nortscts-AT21", staDevices.Get (2), true);

std::ostringstream oss;

oss <<"/NodeList/" << wifiNodes.Get (2)->GetId ()

    <<"/$ns3::MobilityModel/CourseChange";

Config::Connect (oss.str (), MakeCallback (&CourseChange));

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

No, All the frames are not acknowledged as the frames transmitted to a broadcast destination have a duration of 0 seconds. These frames are not acknowledged by the receivers as they are not part of t exchange.

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

As per the screenshot that is attached below, there is a collision between Frames 11 and 12 in the network. Both of these frames wait for a seemingly arbitrary length of time before being sent again allows us to detect a collision.

A screenshot of a computer

Description automatically generated

**3. How can you force the nodes to utilize the RTS/CTS handshake procedure seen in class? What is the reasoning behind this procedure?**

We can use the following sample code to implement RTS/CTS in the ns-3 program:

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

We can force the nodes to utilize this handshake procedure by setting the RTS/CTS packet size threshold to zero.

In wireless networks, when two frames sent from different nodes A and C to B, they may collide at B , but A and C will be unaware of collision due to hidden terminal problem. Hence, RTS/CTS handshake procedure is used to avoid the collision instead of detecting it.

**4. Force the utilization of RTS/CTS in the network:**

**a. Are there any collisions now?**

After implementation of RTS/CTS methodology, no collisions were observed **b**. **Which is the benefit of RTS/CTS?**

RTS/CTS helps to avoid collision that occurs due to Hidden Terminal (nodes).

**c. Where can you find the Network Allocation Vector information?**

Duration field contains the Network Allocation Vector information. Access to the medium is restricted for the time specified by the NAV.

**Output of task1 without RTS/CTS**

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

**Output of task1 with forcing the utilization RTS/CTS**

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated