**9) a) Write a NS3 program to connect two nodes with a point-to-point link, which have a unique interface. Analyze the network performance using UDP client server.**

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

int main(int argc, char *argv[]) {
   CommandLine cmd;
   cmd.Parse(argc, argv);

   // Create two nodes
   NodeContainer nodes;
   nodes.Create(2);

   // Set up the point-to-point link
   PointToPointHelper pointToPoint;
   pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
   pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

   // Install network devices
   NetDeviceContainer devices;
   devices = pointToPoint.Install(nodes);

   // Install Internet stack
   InternetStackHelper internet;
   internet.Install(nodes);

   // Assign IP addresses
   Ipv4AddressHelper address;
   address.SetBase("10.1.1.0", "255.255.255.0");
   Ipv4InterfaceContainer interfaces = address.Assign(devices);

   // Set up the UDP server on node 1
   uint16_t port = 9; // Well-known port for echo
   UdpServerHelper server(port);
```

```
    ApplicationContainer serverApp = server.Install(nodes.Get(1));
    serverApp.Start(Seconds(1.0));
    serverApp.Stop(Seconds(10.0));

    // Set up the UDP client on node 0
    UdpClientHelper client(interfaces.GetAddress(1), port);
    client.SetAttribute("MaxPackets", UintegerValue(320));
    client.SetAttribute("Interval", TimeValue(Seconds(0.05)));
    client.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApp = client.Install(nodes.Get(0));
    clientApp.Start(Seconds(2.0));
    clientApp.Stop(Seconds(10.0));

    // Enable tracing
    pointToPoint.EnablePcapAll("point-to-point-udp");

    // Run the simulation
    Simulator::Run();
    Simulator::Destroy();

    return 0;
}
```

**9) b) Write NS3 program to configure two nodes on an 802.11b physical layer, 802.11b NICs in AD hoc mode, and by default, sends one packet of 1000(applications) bytes to the other node. The physical layer is configured to receive at a fixed RSS (regardless of distance and the transmit power); therefore changing the position of the nodes has no effect. Analyze the performance.**

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/mobility-helper.h"
#include "ns3/wifi-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

int main(int argc, char *argv[]) {
   CommandLine cmd;
   cmd.Parse(argc, argv);

   // Create two nodes
   NodeContainer nodes;
   nodes.Create(2);

   // Configure the WiFi physical layer with 802.11b standard
   YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
   YansWifiPhyHelper phy;
   phy.SetChannel(channel.Create());
   phy.Set("RxGain", DoubleValue(0));
   phy.Set("CcaEdThreshold", DoubleValue(-64));
   phy.Set("TxPowerStart", DoubleValue(16));
   phy.Set("TxPowerEnd", DoubleValue(16));
   phy.Set("RxNoiseFigure", DoubleValue(7));

   // Configure the WiFi MAC layer
   WifiHelper wifi;
   wifi.SetStandard(WIFI_STANDARD_80211b);
   WifiMacHelper mac;
   mac.SetType("ns3::AdhocWifiMac");
```

```
// Install WiFi devices
NetDeviceContainer devices = wifi.Install(phy, mac, nodes);

// Install Internet stack
InternetStackHelper internet;
internet.Install(nodes);

// Assign IP addresses
Ipv4AddressHelper address;
address.SetBase("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign(devices);

// Mobility model - fixed positions
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                              "MinX", DoubleValue(0.0),
                              "MinY", DoubleValue(0.0),
                              "DeltaX", DoubleValue(5.0),
                              "DeltaY", DoubleValue(0.0),
                              "GridWidth", UintegerValue(2),
                              "LayoutType", StringValue("RowFirst"));
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodes);

// UDP Application setup
UdpServerHelper server(9);
ApplicationContainer serverApp = server.Install(nodes.Get(1));
serverApp.Start(Seconds(1.0));
serverApp.Stop(Seconds(10.0));


UdpClientHelper client(interfaces.GetAddress(1), 9);
client.SetAttribute("MaxPackets", UintegerValue(1));
client.SetAttribute("Interval", TimeValue(Seconds(1.0)));
client.SetAttribute("PacketSize", UintegerValue(1000));

ApplicationContainer clientApp=client.Install(nodes.Get(0));
clientApp.Start(Seconds(2.0));
clientApp.Stop(Seconds(10.0));
```

```
  // Enable tracing
  phy.EnablePcap("wifi-adhoc", devices);

    // Run simulation
    Simulator::Run();
    Simulator::Destroy();

    return 0;
}
```