Exp1:

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/mobility-module.h"

#include "ns3/internet-module.h"

#include "ns3/wifi-module.h"

#include "ns3/energy-module.h"

#include "ns3/applications-module.h"

using namespace ns3;

int main() {

LogComponentEnable("WifiSimpleAdhocGrid", LOG\_LEVEL\_INFO);

NodeContainer sensorNodes;

sensorNodes.Create(10);

MobilityHelper mobility;

mobility.SetPositionAllocator("ns3::GridPositionAllocator",

"MinX", DoubleValue(0.0),

"MinY", DoubleValue(0.0),

"DeltaX", DoubleValue(5.0),

"DeltaY", DoubleValue(5.0),

"GridWidth", UintegerValue(3),

"LayoutType", StringValue("RowFirst"));

mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");

mobility.Install(sensorNodes);

WifiHelper wifi;

wifi.SetStandard(WIFI\_PHY\_STANDARD\_80211b);

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();

YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default();

wifiPhy.SetChannel(wifiChannel.Create());

WifiMacHelper wifiMac;

wifiMac.SetType("ns3::AdhocWifiMac");

NetDeviceContainer devices = wifi.Install(wifiPhy, wifiMac, sensorNodes);

InternetStackHelper internet;

internet.Install(sensorNodes);

Ipv4AddressHelper ipv4;

ipv4.SetBase("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = ipv4.Assign(devices);

uint16\_t port = 9;

UdpEchoServerHelper echoServer(port);

ApplicationContainer serverApp = echoServer.Install(sensorNodes.Get(0));

serverApp.Start(Seconds(1.0));

serverApp.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient(interfaces.GetAddress(0), port);

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

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

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

ApplicationContainer clientApp = echoClient.Install(sensorNodes.Get(1));

clientApp.Start(Seconds(2.0));

clientApp.Stop(Seconds(10.0));

Simulator::Run();

Simulator::Destroy();

return 0;

}

Exp 2:

Arduino:

#include <DHT.h>

#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

if (isnan(temperature) || isnan(humidity)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.print(" °C, Humidity: ");

Serial.print(humidity);

Serial.println(" %");

delay(2000);

}

Python visualization:

import matplotlib.pyplot as plt

import pandas as pd

data = pd.read\_csv("sensor\_data.csv")

plt.figure(figsize=(10, 5))

plt.plot(data['Time'], data['Temperature'], label='Temperature (\*C)')

plt.plot(data['Time'], data['Humidity'], label='Humidity (%)')

plt.xlabel('Time (s)')

plt.ylabel('Value')

plt.title('Temperature and Humidity Over Time')

plt.legend()

plt.grid()

plt.show()

Exp 3:

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/wifi-module.h"

#include "ns3/energy-module.h"

#include "ns3/mobility-module.h"

#include "ns3/internet-apps-module.h"

using namespace ns3;

int main() {

NodeContainer sensorNodes;

sensorNodes.Create(10);

WifiHelper wifi;

wifi.SetStandard(WIFI\_PHY\_STANDARD\_80211b);

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();

YansWifiChannelHelper wifiChannel;

wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");

wifiChannel.AddPropagationLoss("ns3::LogDistancePropagationLossModel");

wifiPhy.SetChannel(wifiChannel.Create());

WifiMacHelper wifiMac;

wifiMac.SetType("ns3::AdhocWifiMac");

NetDeviceContainer devices = wifi.Install(wifiPhy, wifiMac, sensorNodes);

MobilityHelper mobility;

mobility.SetPositionAllocator("ns3::GridPositionAllocator",

"MinX", DoubleValue(0.0),

"MinY", DoubleValue(0.0),

"DeltaX", DoubleValue(5.0),

"DeltaY", DoubleValue(5.0),

"GridWidth", UintegerValue(5),

"LayoutType", StringValue("RowFirst"));

mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");

mobility.Install(sensorNodes);

BasicEnergySourceHelper energySourceHelper;

energySourceHelper.Set("BasicEnergySourceInitialEnergyJ", DoubleValue(100.0));

EnergySourceContainer energySources = energySourceHelper.Install(sensorNodes);

WifiRadioEnergyModelHelper radioEnergyHelper;

radioEnergyHelper.Set("TxCurrentA", DoubleValue(0.017));

radioEnergyHelper.Set("RxCurrentA", DoubleValue(0.013));

radioEnergyHelper.Install(devices, energySources);

InternetStackHelper internet;

internet.Install(sensorNodes);

Ipv4AddressHelper ipv4;

ipv4.SetBase("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = ipv4.Assign(devices);

uint16\_t port = 9;

UdpEchoServerHelper echoServer(port);

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

serverApps.Start(Seconds(1.0));

serverApps.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient(interfaces.GetAddress(0), port);

echoClient.SetAttribute("MaxPackets", UintegerValue(5));

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

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

ApplicationContainer clientApps = echoClient.Install(sensorNodes.Get(9));

clientApps.Start(Seconds(2.0));

clientApps.Stop(Seconds(10.0));

Simulator::Stop(Seconds(10.0));

Simulator::Run();

Simulator::Destroy();

return 0;

}

Exp 4:

#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"

#include "ns3/wifi-module.h"

using namespace ns3;

bool DetectIntrusion(Ptr<Packet> packet) {

if (packet->GetSize() > 1024) {

NS\_LOG\_UNCOND("Intrusion detected: Packet size exceeds threshold!");

return true;

}

return false;

}

void ReceivePacket(Ptr<Socket> socket) {

Ptr<Packet> packet = socket->Recv();

if (DetectIntrusion(packet)) {

NS\_LOG\_UNCOND("Intrusion logged for further analysis.");

} else {

NS\_LOG\_UNCOND("Normal packet received.");

}

}

int main(int argc, char \*argv[]) {

NodeContainer nodes;

nodes.Create(3);

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));

pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

NetDeviceContainer devices = pointToPoint.Install(nodes);

InternetStackHelper stack;

stack.Install(nodes);

Ipv4AddressHelper address;

address.SetBase("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces = address.Assign(devices);

uint16\_t port = 9;

UdpServerHelper server(port);

ApplicationContainer serverApps = server.Install(nodes.Get(1));

serverApps.Start(Seconds(1.0));

serverApps.Stop(Seconds(10.0));

UdpClientHelper client(interfaces.GetAddress(1), port);

client.SetAttribute("MaxPackets", UintegerValue(10));

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

client.SetAttribute("PacketSize", UintegerValue(512));

ApplicationContainer clientApps = client.Install(nodes.Get(0));

clientApps.Start(Seconds(2.0));

clientApps.Stop(Seconds(10.0));

TypeId tid = TypeId::LookupByName("ns3::UdpSocketFactory");

Ptr<Socket> recvSocket = Socket::CreateSocket(nodes.Get(1), tid);

InetSocketAddress local = InetSocketAddress(Ipv4Address::GetAny(), port);

recvSocket->Bind(local);

recvSocket->SetRecvCallback(MakeCallback(&ReceivePacket));

Simulator::Run();

Simulator::Destroy();

return 0;

}

Exp 5:

#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"

#include "ns3/wifi-module.h"

using namespace ns3;

using namespace std;

Vector EstimatePosition(Vector anchor1, Vector anchor2, Vector anchor3, double d1, double d2, double d3) {

double x = (anchor1.x + anchor2.x + anchor3.x) / 3.0;

double y = (anchor1.y + anchor2.y + anchor3.y) / 3.0;

return Vector(x, y, 0);

}

int main(int argc, char \*argv[]) {

Vector anchor1(0, 0, 0);

Vector anchor2(100, 0, 0);

Vector anchor3(50, 50, 0);

double d1 = 10.0, d2 = 10.0, d3 = 10.0;

Vector estimatedPosition = EstimatePosition(anchor1, anchor2, anchor3, d1, d2, d3);

cout << "Estimated Position: " << estimatedPosition << endl;

Simulator::Run();

Simulator::Destroy();

return 0;

}

Exp 6:

import paho.mqtt.client as mqtt

import random

import time

def on\_connect(client, userdata, flags, rc):

print("Connected to MQTT Broker")

client.subscribe("sensor/data")

def on\_message(client, userdata, msg):

print(f"Received: {msg.payload.decode()}")

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect("localhost", 1883, 60)

while True:

sensor\_data = random.randint(20, 30)

client.publish("sensor/data", sensor\_data)

print(f"Sent: {sensor\_data}")

time.sleep(2)

Exp 7:

import paho.mqtt.client as mqtt

client = mqtt.Client()

client.connect("localhost", 1883, 60)

def control\_light(state):

client.publish("home/lights", state)

print(f"Light turned {state}")

control\_light("ON")

control\_light("OFF")

Exp 8:

import paho.mqtt.client as mqtt

import random

import time

client = mqtt.Client()

client.connect("localhost", 1883, 60)

def publish\_sensor\_data():

temperature = random.randint(20, 30)

humidity = random.randint(30, 70)

air\_quality = random.randint(50, 100)

client.publish("environment/temperature", temperature)

client.publish("environment/humidity", humidity)

client.publish("environment/air\_quality", air\_quality)

print(f"Temperature: {temperature}°C, Humidity: {humidity}%, Air Quality: {air\_quality}")

while True:

publish\_sensor\_data()

time.sleep(5)