Exp 5

// Experiment 5: Implementing and Evaluating Node Localization Algorithms in a WSN

// (NS3 Trilateration Localization)

#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"

#include <iostream>

#include <cmath>

using namespace ns3;

// Function to estimate the position of a node using trilateration

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

// Trilateration calculations (placeholder)

// Using a simple averaging method for demonstration purposes.

// Actual trilateration involves solving equations based on circle intersections.

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[]) {

// Define three anchor nodes with known positions

Vector anchor1(0, 0, 0);

Vector anchor2(100, 0, 0);

Vector anchor3(50, 50, 0);

// Distances from the unknown node to the anchor nodes

double d1 = 10.0;

double d2 = 15.0;

double d3 = 12.0;

// Estimate the position of the unknown node

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

// Output the estimated position

std::cout << "Estimated Position: (" << estimatedPosition.x << ", " << estimatedPosition.y << ")" << std::endl;

// Run NS3 simulation (even though no actual network is simulated in this case)

Simulator::Run();

Simulator::Destroy();

return 0;

}

55555555555555555555555555555555555555555555555555555555555555555555555555

Exp 6

# Experiment 6: Implementing a Simple IoT Communication Using the MQTT Protocol

# (MQTT Publisher in Python)

import paho.mqtt.client as mqtt

import random

import time

# Callback function when connected to the broker

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

print("Connected with result code " + str(rc))

client.subscribe("sensor/data")

# Callback function when a message is received

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

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

# Setup MQTT client

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

# Connect to MQTT broker (use the correct broker address if not running locally)

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

# Start MQTT loop in a separate thread

client.loop\_start()

# Publish random sensor data

try:

while True:

sensor\_data = random.randint(20, 30) # Example: Random temperature data

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

print(f"Published: {sensor\_data} to sensor/data")

time.sleep(2) # Publish data every 2 seconds

except KeyboardInterrupt:

print("Stopping MQTT Publisher...")

# Stop the MQTT loop and disconnect

client.loop\_stop()

client.disconnect()

66666666666666666666666666666666666666666666666666666666

Exp 7

# Experiment 7: Create a Simple Home Automation System Using IoT

# (MQTT Control for Smart Light)

import paho.mqtt.client as mqtt

import time

# Setup MQTT client

client = mqtt.Client()

# Connect to MQTT broker (use the correct broker address if not running locally)

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

# Function to control the smart light

def control\_light(state):

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

print(f"Sent command: {state}")

# Turn the light ON and OFF with a delay

control\_light("ON")

time.sleep(2) # Simulate delay before turning it off

control\_light("OFF")

# Disconnect the client

client.disconnect()

7777777777777777777777777777777777

Exp 8

# Experiment 8: Building a Basic IoT Environmental Monitoring System

# (Environmental Monitoring with MQTT)

import paho.mqtt.client as mqtt

import random

import time

# Setup MQTT client

client = mqtt.Client()

# Connect to MQTT broker (use the correct broker address if not running locally)

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

# Simulate sensor readings and publish

try:

while True:

temperature = random.randint(20, 30) # Example temperature data

humidity = random.randint(30, 70) # Example humidity data

air\_quality = random.randint(50, 100) # Example air quality data

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

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

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

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

time.sleep(5) # Publish every 5 seconds

except KeyboardInterrupt:

print("\nStopping sensor data transmission.")

# Disconnect the client

client.disconnect()