Exp 1

// Experiment 1: Setting up a Simple Wireless Sensor Network (NS3 Simulation Script)

#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() {

// Set up logging for debugging

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

// Create nodes for the network

NodeContainer sensorNodes;

sensorNodes.Create(10);

// Set up mobility model

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"));

// Assign a constant position mobility model

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

mobility.Install(sensorNodes);

// Install Wi-Fi devices

WifiHelper wifi;

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

// Create Wi-Fi PHY and channel helpers

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();

YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default();

wifiPhy.SetChannel(wifiChannel.Create());

// Configure Wi-Fi MAC

WifiMacHelper wifiMac;

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

// Install Wi-Fi devices on nodes

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

// Install Internet stack on nodes

InternetStackHelper internet;

internet.Install(sensorNodes);

// Assign IP addresses

Ipv4AddressHelper ipv4;

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

Ipv4InterfaceContainer interfaces = ipv4.Assign(devices);

// Set up a UDP echo server on node 0

uint16\_t port = 9; // UDP port number

UdpEchoServerHelper echoServer(port);

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

serverApp.Start(Seconds(1.0)); // Start at 1 second

serverApp.Stop(Seconds(10.0)); // Stop at 10 seconds

// Set up a UDP echo client on node 1

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

echoClient.SetAttribute("MaxPackets", UintegerValue(2)); // Send 2 packets

echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0))); // 1-second interval

echoClient.SetAttribute("PacketSize", UintegerValue(1024)); // Packet size of 1024 bytes

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

clientApp.Start(Seconds(2.0)); // Start at 2 seconds

clientApp.Stop(Seconds(10.0)); // Stop at 10 seconds

// Run the simulation

Simulator::Run();

Simulator::Destroy();

return 0;

}

5555555555555555555555555555555555555555555555555555555555555

Exp 2

// Experiment 2: Collecting and Analyzing Environmental Data using Arduino

#include <DHT.h> // Include the DHT sensor library

#define DHTPIN 2 // Digital pin connected to the DHT sensor

#define DHTTYPE DHT22 // Define sensor type (DHT22 or AM2302)

DHT dht(DHTPIN, DHTTYPE); // Initialize the DHT sensor

void setup() {

Serial.begin(9600); // Start serial communication at 9600 baud

Serial.println("DHT Sensor Test!");

dht.begin(); // Initialize the sensor

}

void loop() {

delay(2000); // Wait 2 seconds between measurements

// Read humidity and temperature from the sensor

float humidity = dht.readHumidity();

float temperature = dht.readTemperature();

// Check if the sensor reading failed

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

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

return; // Exit loop and try again

}

// Print values to the Serial Monitor

Serial.print("Humidity: ");

Serial.print(humidity);

Serial.print(" %\t");

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" \*C");

// The collected data can be logged to a file and analyzed using Python

}

# Experiment 2: Plotting Temperature and Humidity Data using Python

import matplotlib.pyplot as plt # Import Matplotlib for plotting

import pandas as pd # Import Pandas for data handling

# Load data from a CSV file (assume it contains 'Time', 'Temperature', and 'Humidity' columns)

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

# Plot the data

plt.figure(figsize=(10, 5)) # Set figure size

# Plot temperature over time

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

# Plot humidity over time

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

# Label the axes

plt.xlabel('Time (s)')

plt.ylabel('Value')

# Set the title

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

# Add legend and grid

plt.legend()

plt.grid()

# Show the plot

plt.show()

555555555555555555555555555555555555555555555555555555

Exp 3

// Experiment 3: Measuring Energy Efficiency of Sensor Nodes (NS3 Energy Model Example)

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

// Create a container for sensor nodes

NodeContainer sensorNodes;

sensorNodes.Create(10);

// Configure WiFi settings

WifiHelper wifi;

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

// Create WiFi PHY and channel helpers

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();

YansWifiChannelHelper wifiChannel;

// Set propagation delay and loss models

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

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

// Set up WiFi channel

wifiPhy.SetChannel(wifiChannel.Create());

// Configure WiFi MAC

WifiMacHelper wifiMac;

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

// Install WiFi devices on nodes

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

// Configure mobility model for sensor nodes

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"));

// Assign a constant position mobility model

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

mobility.Install(sensorNodes);

// Install energy model for sensor nodes

BasicEnergySourceHelper energySourceHelper;

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

// Assign energy sources to nodes

EnergySourceContainer energySources = energySourceHelper.Install(sensorNodes);

// Configure WiFi radio energy consumption model

WifiRadioEnergyModelHelper radioEnergyHelper;

radioEnergyHelper.Set("TxCurrentA", DoubleValue(0.017)); // Transmission current

radioEnergyHelper.Set("RxCurrentA", DoubleValue(0.013)); // Reception current

// Install energy model on devices

radioEnergyHelper.Install(devices, energySources);

// Install Internet stack on nodes

InternetStackHelper internet;

internet.Install(sensorNodes);

// Assign IP addresses

Ipv4AddressHelper ipv4;

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

Ipv4InterfaceContainer interfaces = ipv4.Assign(devices);

// Set up a UDP echo server on node 0

uint16\_t port = 9; // UDP port number

UdpEchoServerHelper echoServer(port);

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

serverApps.Start(Seconds(1.0)); // Start at 1 second

serverApps.Stop(Seconds(10.0)); // Stop at 10 seconds

// Set up a UDP echo client on node 9

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

echoClient.SetAttribute("MaxPackets", UintegerValue(5)); // Send 5 packets

echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0))); // 1-second interval

echoClient.SetAttribute("PacketSize", UintegerValue(1024)); // Packet size of 1024 bytes

// Install client application on node 9

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

clientApps.Start(Seconds(2.0)); // Start at 2 seconds

clientApps.Stop(Seconds(10.0)); // Stop at 10 seconds

// Set simulation stop time

Simulator::Stop(Seconds(10.0));

// Run the simulation

Simulator::Run();

Simulator::Destroy();

return 0;

}

55555555555555555555555555555555555555555555555555555555555555555555555555555555

Exp 4

// Experiment 4: Implementing a Basic Intrusion Detection System (IDS) in WSN

#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;

// Function to simulate intrusion detection

bool DetectIntrusion(Ptr<Packet> packet) {

if (packet->GetSize() > 1024) { // Example: Flag packets larger than 1024 bytes

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

return true;

}

return false;

}

// Packet reception callback function

void ReceivePacket(Ptr<Socket> socket) {

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

if (DetectIntrusion(packet)) {

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

} else {

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

}

}

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

// Create nodes

NodeContainer nodes;

nodes.Create(3);

// Create a PointToPoint connection

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps")); // Set data rate

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

// Install PointToPoint devices on nodes

NetDeviceContainer devices = pointToPoint.Install(nodes);

// Install Internet Stack on nodes

InternetStackHelper stack;

stack.Install(nodes);

// Assign IP addresses

Ipv4AddressHelper address;

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

Ipv4InterfaceContainer interfaces = address.Assign(devices);

// Configure a UDP server on node 1

uint16\_t port = 9; // UDP port number

UdpServerHelper server(port);

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

serverApps.Start(Seconds(1.0)); // Start server at 1 second

serverApps.Stop(Seconds(10.0)); // Stop server at 10 seconds

// Configure a UDP client on node 0

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

client.SetAttribute("MaxPackets", UintegerValue(10)); // Send 10 packets

client.SetAttribute("Interval", TimeValue(Seconds(1.0))); // 1-second interval

client.SetAttribute("PacketSize", UintegerValue(512)); // Packet size of 512 bytes

// Install client application on node 0

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

clientApps.Start(Seconds(2.0)); // Start at 2 seconds

clientApps.Stop(Seconds(10.0)); // Stop at 10 seconds

// Run the simulation

Simulator::Run();

Simulator::Destroy();

return 0;

}