SIES College of Arts, Science & Commerce

RISE WITH EDUCATION

NAAC REACCREDITED - 'A' GRADE

ISO 9001 : 2008

S.I.E.S College of Arts, Science and Commerce(Autonomous)
Sion(W), Mumbai – 400 022.

CERTIFICATE

This is to certify that ~~Miss.~~/Mr ___**Varun**_**Vangari**__________________
Roll No.__**TCS2324086**__ has successfully completed the necessary course of experiments in the subject of **Wireless Sensor Networks & Mobile Communication** during the academic year **2023 – 2024** complying with the requirements of **University of Mumbai**, for the course of **TYBSc Computer Science [Semester-VI].**

Prof. In-Charge
JESICA D'CRUZ

Examination date:

Examiner's Signature & Date:

Head of the Department
Prof. Manoj Singh

College Seal

Index Page

Sr. No	Description	Page No	Date	Faculty Signature
1	Understanding the Sensor Node Hardware. (For Eg. Sensors, Nodes(Sensor mote), Base Station, Graphical User Interface.)	3	8/1/24	
2	Exploring and understanding TinyOS computational concepts:- Events, Commands and Task. - nesC model - nesC Components	5	8/1/24	
3	Understanding TOSSIM for - Mote-mote radio communication - Mote-PC serial communication	8	8/1/24	
4	Create and simulate a simple adhoc network	13	15/1/24	
5	Understanding, Reading and Analyzing Routing Table of a network.	16	15/1/24	
6	Create a basic MANET implementation simulation for Packet animation and Packet Trace.	19	22/1/24	
7	Create MAC protocol simulation implementation for wireless sensor Network.	21	29/1/24	
8	Simulate Mobile Adhoc Network with Directional Antenna.	23	5/2/24	
9	Create a mobile network using Cell Tower, Central Office Server, Web browser and Web Server.Simulate connection between them.	26	5/2/24	

SIES

Wireless Sensor Networks & Mobile Communication

Practical No.1

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	8/1/24	**Practical No**	1

A) AIM:

Understanding the Sensor Node Hardware. (For Eg. Sensors, Nodes(Sensor mote), Base Station, Graphical User Interface.)

B) DESCRIPTION:

Sensor nodes typically consist of several components:

Sensors: These are devices that detect physical properties such as temperature, humidity, light, motion, etc. They convert these properties into electrical signals that can be processed by the node.

Nodes (Sensor Mote): The sensor nodes themselves contain microcontrollers or microprocessors, memory, power management systems, and communication interfaces. They gather data from the sensors, process it if necessary, and transmit it wirelessly to a base station.

Base Station: This is a central hub that collects data from multiple sensor nodes. It typically has more computational power and storage capacity compared to the nodes and is responsible for aggregating, storing, and sometimes analyzing the data received from the nodes.

Graphical User Interface (GUI): This is a software interface that allows users to interact with the sensor network. It may display real-time data, provide options for configuring the nodes or network parameters, and offer tools for data visualization and analysis. GUIs make it easier for users to monitor and manage the sensor network effectively

C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

Understanding sensor node hardware typically involves familiarizing oneself with the components involved in a wireless sensor network (WSN) setup. Here's a breakdown:

Sensors: Sensors are the primary components responsible for gathering data from the environment. They can vary widely depending on the application but commonly include temperature sensors, humidity sensors, light sensors, motion sensors, etc. These sensors convert physical phenomena into electrical signals that can be processed by the sensor node.

Sensor Nodes (Sensor Motes): Sensor nodes, also known as sensor motes, are small, autonomous devices equipped with one or more sensors, a processing unit (microcontroller or microprocessor), memory, communication interfaces (e.g., Wi-Fi, Zigbee, Bluetooth Low Energy), and power management circuitry. These nodes are typically battery-powered and are designed to be deployed in large numbers to form a wireless sensor network.

Base Station (Sink Node): The base station, also referred to as the sink node, serves as the gateway between the sensor network and the external world (e.g., a computer or server). It is usually more powerful than individual sensor nodes and may have greater processing capabilities and energy resources. The base station collects data from sensor nodes either periodically or upon request, aggregates the data, and may perform initial processing before transmitting it to a central server or storage system.

Graphical User Interface (GUI): The graphical user interface provides a means for users to interact with the sensor network, visualize data, configure settings, and monitor the system's status in a user-friendly manner. The GUI may be a standalone application running on a computer or a web-based interface accessible through a browser. It allows users to view real-time or historical data, set thresholds or alarms, and manage the sensor network remotely.

SIES

Wireless Sensor Networks & Mobile Communication

Practical No.2

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	8/1/24	**Practical No**	2

A) AIM:

Exploring and understanding TinyOS computational concepts:- Events, Commands and Task.

- nesC model

- nesC Components

B) DESCRIPTION:

TinyOS computational concepts:

Events: Events in TinyOS represent asynchronous occurrences or triggers that can initiate actions or computations within the system. Components can define event handlers to react to these events, enabling event-driven programming paradigms.

Commands: Commands are synchronous operations that components can invoke to perform specific tasks or actions within the system. They typically involve interactions between components or with underlying hardware resources.

Tasks: Tasks in TinyOS represent asynchronous operations or computations that are executed sequentially by the system. They are typically used for non-blocking operations that may involve lengthy computations or interactions with external entities.

nesC model: nesC (Network Embedded Systems C) is a programming language specifically designed for programming embedded systems, particularly those with resource-constrained environments like TinyOS-based

systems. It follows a component-based model where functionality is encapsulated within reusable modules called components, facilitating modularity and code reuse.

nesC Components: nesC components are modular building blocks of TinyOS applications. They encapsulate functionality and provide well-defined interfaces for communication with other components. Components can interact through events, commands, and tasks, enabling developers to compose complex systems from reusable and interchangeable parts

C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

TinyOS is an operating system designed for wireless sensor networks (WSNs) and is known for its lightweight and event-driven architecture. To understand TinyOS computational concepts, let's delve into its key components and computational model:

Events, Commands, and Tasks:

Events:

Events are the primary mechanism for asynchronous communication and coordination in TinyOS.

They represent occurrences or conditions in the system, such as sensor readings, timer expirations, or communication events.

Components can generate events to signal the occurrence of specific conditions.

Other components can subscribe to these events and define event handlers to react accordingly.

Events in TinyOS are typically implemented using a publish-subscribe model, where components publish events, and interested components subscribe to them.

Commands:

Commands are synchronous operations that components can perform on each other.

They are used for requesting actions or services from other components.

Commands are typically blocking and are executed in a sequential manner.

Components can define interfaces exposing commands that other components can invoke.

Components that provide services implement command handlers to process incoming command requests.

Tasks:

Tasks represent units of work or computations that can be executed asynchronously.

They are typically initiated in response to events or commands.

Tasks are non-blocking and execute concurrently with other tasks.

Tasks in TinyOS are implemented using concurrency constructs like concurrency blocks or software-defined tasks.

Tasks can be scheduled by the system scheduler based on priorities or execution constraints.

nesC Model:

nesC (Network Embedded Systems C) is the programming language used to write applications and components for TinyOS. It is an extension of the C programming language with constructs for event-driven programming and component-based development. The nesC model revolves around the following principles:

Component-Based: nesC encourages modularization through components, which encapsulate functionality and expose interfaces for interaction with other components.

Event-Driven: The programming model is event-driven, where components react to events generated by the system or other components.

Concurrency: nesC supports concurrency through tasks, allowing multiple tasks to execute concurrently and asynchronously.

Static Configuration: Components in nesC are statically configured at compile time, meaning that connections between components and allocation of resources are determined at compile time rather than runtime.

nesC Components:

nesC components are building blocks of TinyOS applications, encapsulating functionality and defining interfaces for interaction with other components. A nesC component typically consists of the following elements:

Module Interface: Defines the interfaces exposed by the component, including commands, events, and configuration parameters.

Module Implementation: Contains the implementation of the component, including event handlers, command handlers, and task definitions.

Configuration: Specifies the wiring of components and configuration parameters. Components can be instantiated and interconnected within a configuration to form the application structure.

Uses and Provides Interfaces: Components can use interfaces provided by other components (uses) and provide interfaces for use by other components (provides).

Wireless Sensor Networks & Mobile Communication

Practical No.3

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	8/1/24	**Practical No**	3a & 3b

A) AIM:

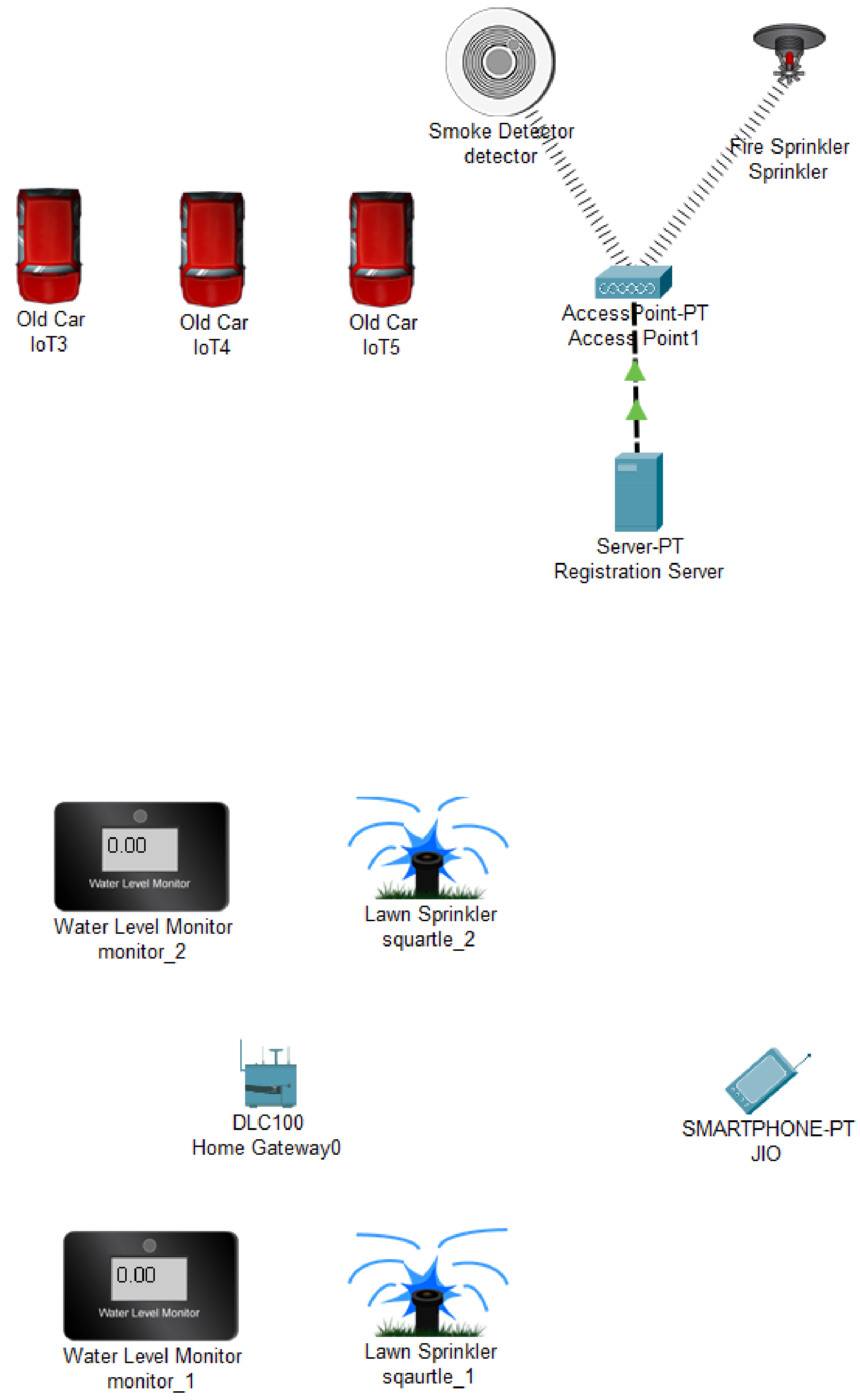
Understanding TOSSIM for
- Mote-mote radio communication
- Mote-PC serial communication

B) DESCRIPTION:

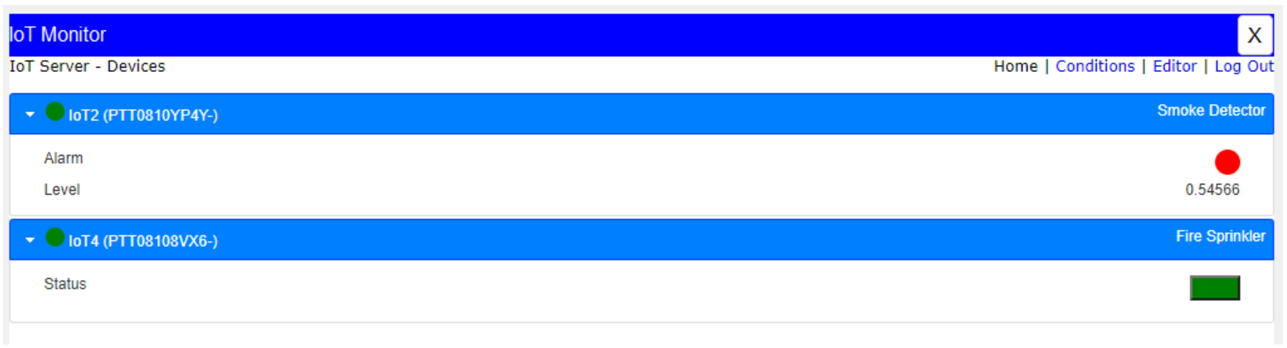
TOSSIM (TinyOS Simulation) facilitates mote-mote radio communication by emulating radio propagation effects and allowing developers to simulate message transmission and reception between virtual motes. It provides a platform to evaluate the performance of communication protocols and algorithms in a controlled environment, aiding in debugging and optimizing wireless sensor network applications.

C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

Topology:



Configuration:



IoT Monitor

IoT Server - Devices

Home | Conditions | Editor | Log Out

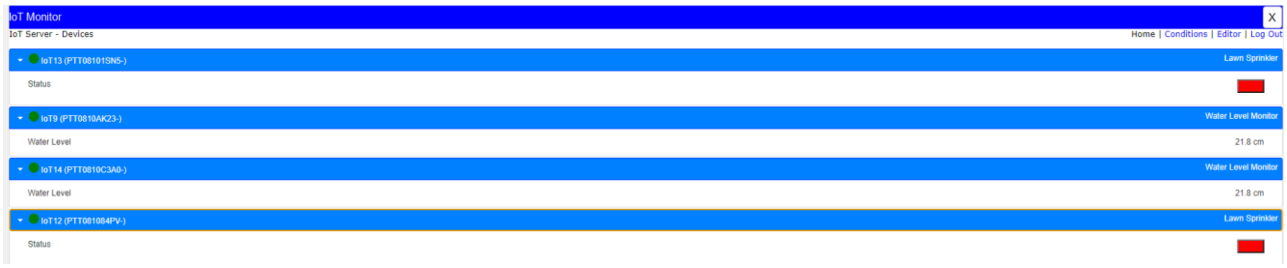
IoT2 (PTT0810YP4Y-) Smoke Detector

Alarm ●

Level 0.54566

IoT4 (PTT08108VX6-) Fire Sprinkler

Status 🟩



IoT Monitor

IoT Server - Devices

Home | Conditions | Editor | Log Out

IoT13 (PTT08101SN5-) Lawn Sprinkler

Status 🟥

IoT9 (PTT0810AK23-) Water Level Monitor

Water Level 21.8 cm

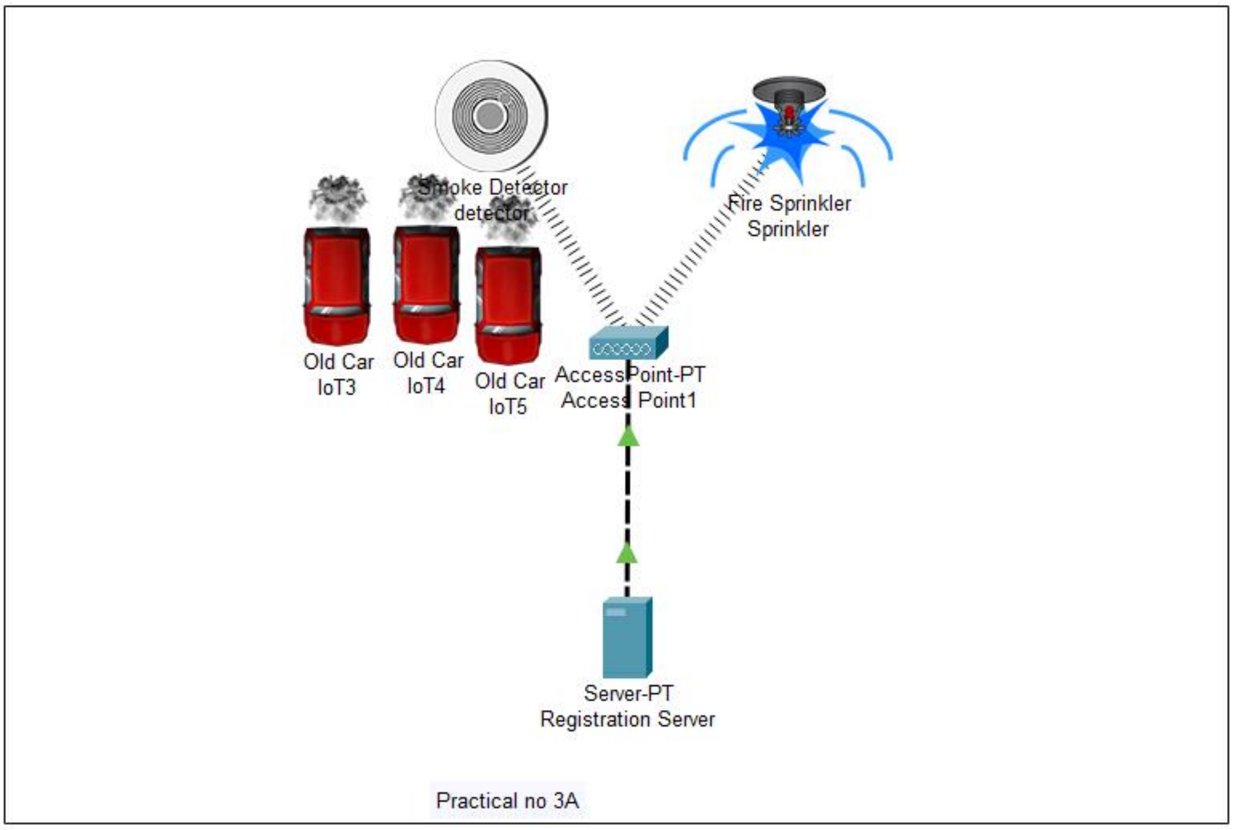
IoT14 (PTT0810C3A0-) Water Level Monitor

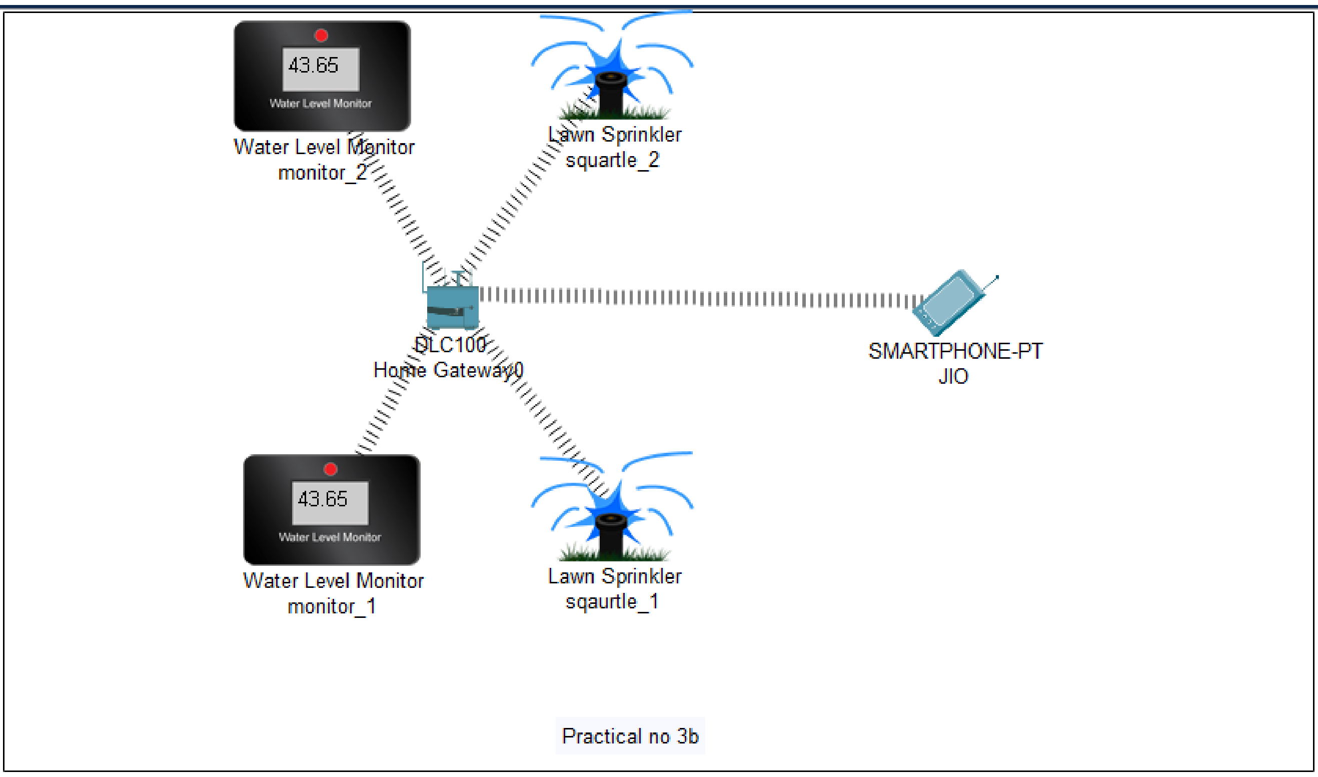
Water Level 21.8 cm

IoT12 (PTT081084PV-) Lawn Sprinkler

Status 🟥

Output:


Smoke Detector
detector
Fire Sprinkler
Sprinkler
Old Car
IoT3
Old Car
IoT4
Old Car
IoT5
Access Point-PT
Access Point1
Server-PT
Registration Server
Practical no 3A


43.65
Water Level Monitor
Water Level Monitor
monitor_2
Lawn Sprinkler
squartle_2
DLC100
Home Gateway0
SMARTPHONE-PT
JIO
43.65
Water Level Monitor
Water Level Monitor
monitor_1
Lawn Sprinkler
sqaurtle_1
Practical no 3b

Wireless Sensor Networks & Mobile Communication

Practical No.4

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	15/01/24	**Practical No**	4

A) AIM:

Create and simulate a simple adhoc network

B) DESCRIPTION:

A simple ad hoc network is established using multiple devices directly communicating with each other, without the need for a centralized infrastructure. Devices exchange data directly with nearby peers, forming a dynamic network topology, allowing for decentralized and flexible communication. Simulate this network by showcasing devices connecting and exchanging information without reliance on fixed infrastructure.

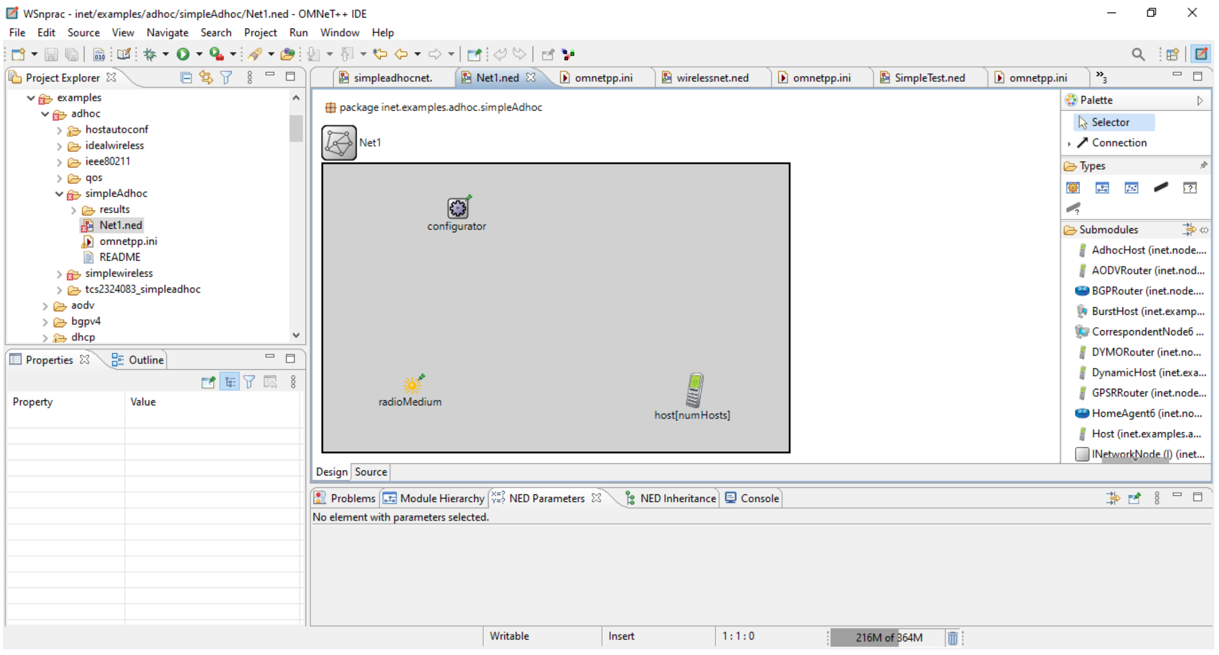
C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

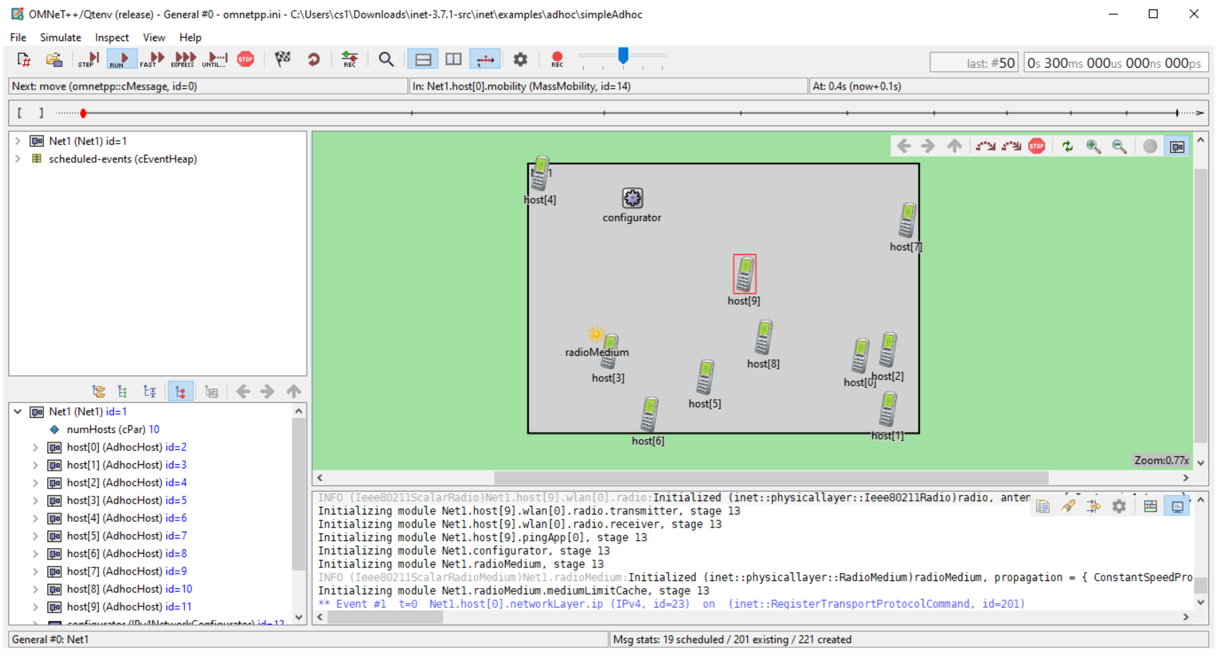
Network topology(only for cisco packet tracer practical's):

Configurations:

```
package inet.examples.adhoc.simpleAdhoc;

import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;

import inet.networklayer.ipv4.IPv4;

import inet.node.inet.AdhocHost;

import inet.physicallayer.ieee80211.packetlevel.Ieee80211ScalarRadioMedium;

import inet.world.radio.ChannelControl;

network Net1

{parameters:

    int numHosts;

  submodules:

    host[numHosts]: AdhocHost {

      parameters:

        @display("r=,,#707070");}

    configurator: IPv4NetworkConfigurator {

      @display("p=152,50");}

    radioMedium: Ieee80211ScalarRadioMedium {

      parameters:

        @display("p=100,250");

    }

}
```

Output




OMNeT++/Qtenv (release) - General #0 - omnetpp.ini - C:\Users\cs1\Downloads\inet-3.7.1-src\inet\examples\adhoc\simpleAdhoc
File Simulate Inspect View Help
last: #50
0s 300ms 000us 000ns 000ps
Next: move (omnetpp::cMessage, id=0)
In: Net1.host[0].mobility (MassMobility, id=14)
At: 0.4s (now+0.1s)
Net1 (Net1) id=1
scheduled-events (cEventHeap)
host[4]
configurator
host[7]
host[9]
radioMedium
host[8]
host[3]
host[5]
host[0]
host[2]
host[6]
host[1]
Zoom:0.77x
Net1 (Net1) id=1
numHosts (cPar) 10
host[0] (AdhocHost) id=2
host[1] (AdhocHost) id=3
host[2] (AdhocHost) id=4
host[3] (AdhocHost) id=5
host[4] (AdhocHost) id=6
host[5] (AdhocHost) id=7
host[6] (AdhocHost) id=8
host[7] (AdhocHost) id=9
host[8] (AdhocHost) id=10
host[9] (AdhocHost) id=11
General #0: Net1
Msg stats: 19 scheduled / 201 existing / 221 created

```
INFO (Ieee80211ScalarRadio)Net1.host[9].wlan[0].radio:Initialized (inet::physicallayer::Ieee80211Radio)radio, anten
Initializing module Net1.host[9].wlan[0].radio.transmitter, stage 13
Initializing module Net1.host[9].wlan[0].radio.receiver, stage 13
Initializing module Net1.host[9].pingApp[0], stage 13
Initializing module Net1.configurator, stage 13
Initializing module Net1.radioMedium, stage 13
INFO (Ieee80211ScalarRadioMedium)Net1.radioMedium:Initialized (inet::physicallayer::RadioMedium)radioMedium, propagation = { ConstantSpeedPro
Initializing module Net1.radioMedium.mediumLimitCache, stage 13
** Event #1  t=0  Net1.host[0].networkLayer.ip (IPv4, id=23)  on  (inet::RegisterTransportProtocolCommand, id=201)
```

Wireless Sensor Networks & Mobile Communication

Practical No.5

DEPARTMENT OF COMPUTER SCIENCE

Name:	Ganesh Kumaraswamy Udutha	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	15/1/24	**Practical No**	5

A) AIM:

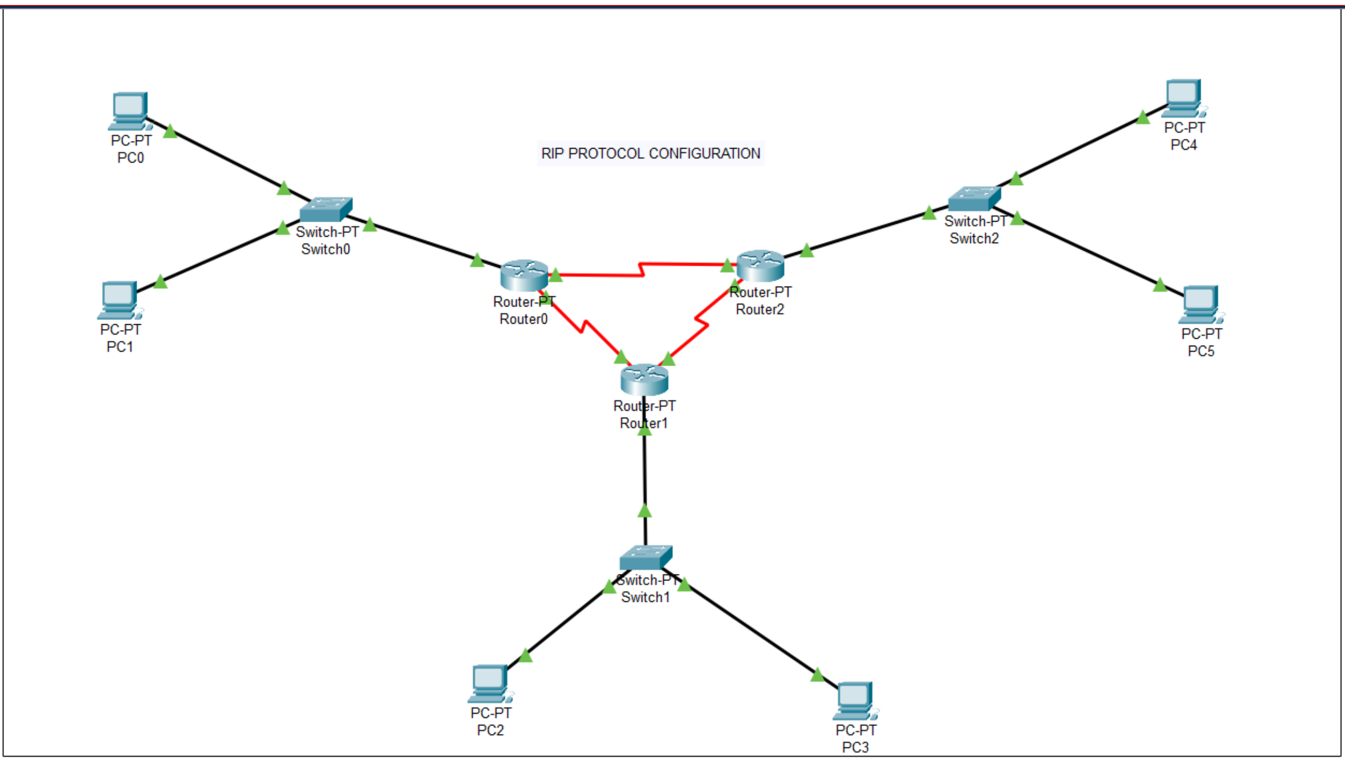
Understanding, Reading and Analyzing Routing Table of a network

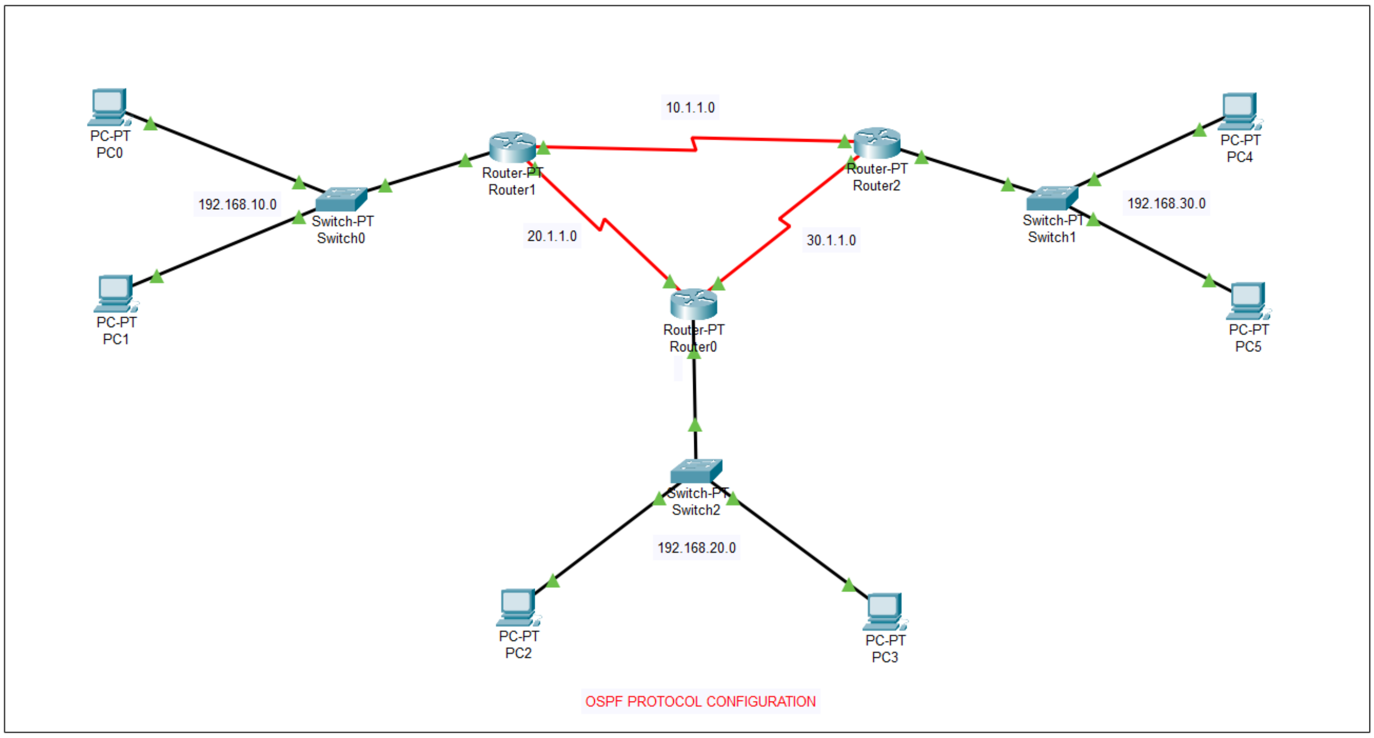
B) DESCRIPTION:

A routing table in a network device like a router or switch lists available routes to specific destinations, including destination network addresses, subnet masks, and next-hop addresses. It helps administrators ensure efficient traffic forwarding, troubleshoot connectivity issues, and optimize network performance by analyzing route metrics, path selection, and protocol stability.

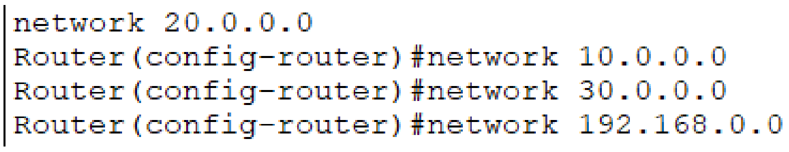
C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

Network topology(only for cisco packet tracer practical's):

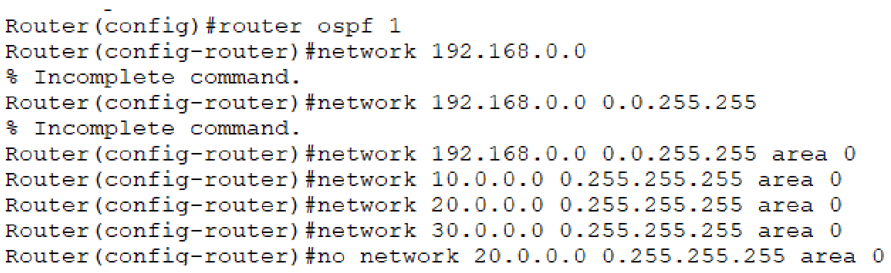

RIP PROTOCOL CONFIGURATION
PC-PT PC0
PC-PT PC1
Switch-PT Switch0
Router-PT Router0
Router-PT Router2
Router-PT Router1
Switch-PT Switch2
PC-PT PC4
PC-PT PC5
Switch-PT Switch1
PC-PT PC2
PC-PT PC3


10.1.1.0
PC-PT PC0
192.168.10.0
Switch-PT Switch0
PC-PT PC1
Router-PT Router1
20.1.1.0
Router-PT Router2
30.1.1.0
Switch-PT Switch1
192.168.30.0
PC-PT PC4
PC-PT PC5
Router-PT Router0
Switch-PT Switch2
192.168.20.0
PC-PT PC2
PC-PT PC3
OSPF PROTOCOL CONFIGURATION

Configurations:



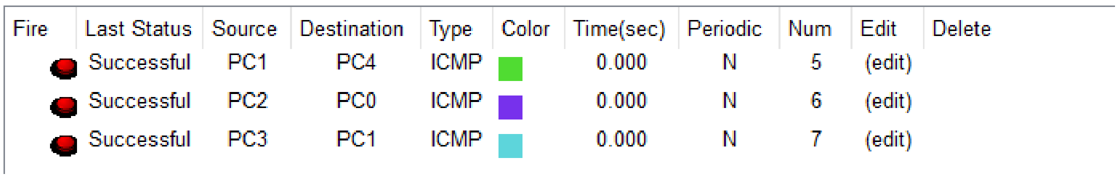
```
network 20.0.0.0
Router(config-router)#network 10.0.0.0
Router(config-router)#network 30.0.0.0
Router(config-router)#network 192.168.0.0
```



```
Router(config)#router ospf 1
Router(config-router)#network 192.168.0.0
% Incomplete command.
Router(config-router)#network 192.168.0.0 0.0.255.255
% Incomplete command.
Router(config-router)#network 192.168.0.0 0.0.255.255 area 0
Router(config-router)#network 10.0.0.0 0.255.255.255 area 0
Router(config-router)#network 20.0.0.0 0.255.255.255 area 0
Router(config-router)#network 30.0.0.0 0.255.255.255 area 0
Router(config-router)#no network 20.0.0.0 0.255.255.255 area 0
```

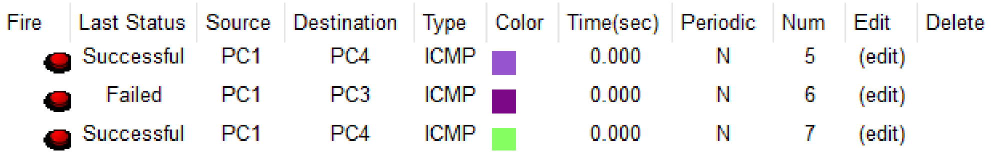
Output

RIP Protocol :-



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC1	PC4	ICMP	🟩	0.000	N	5	(edit)	
●	Successful	PC2	PC0	ICMP	🟪	0.000	N	6	(edit)	
●	Successful	PC3	PC1	ICMP	🟦	0.000	N	7	(edit)	

OSPF Protocol :-



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	Successful	PC1	PC4	ICMP	🟪	0.000	N	5	(edit)	
●	Failed	PC1	PC3	ICMP	🟪	0.000	N	6	(edit)	
●	Successful	PC1	PC4	ICMP	🟩	0.000	N	7	(edit)	

SIES

Wireless Sensor Networks & Mobile Communication

Practical No.6

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	08/01/24	**Practical No**	6

A) AIM:

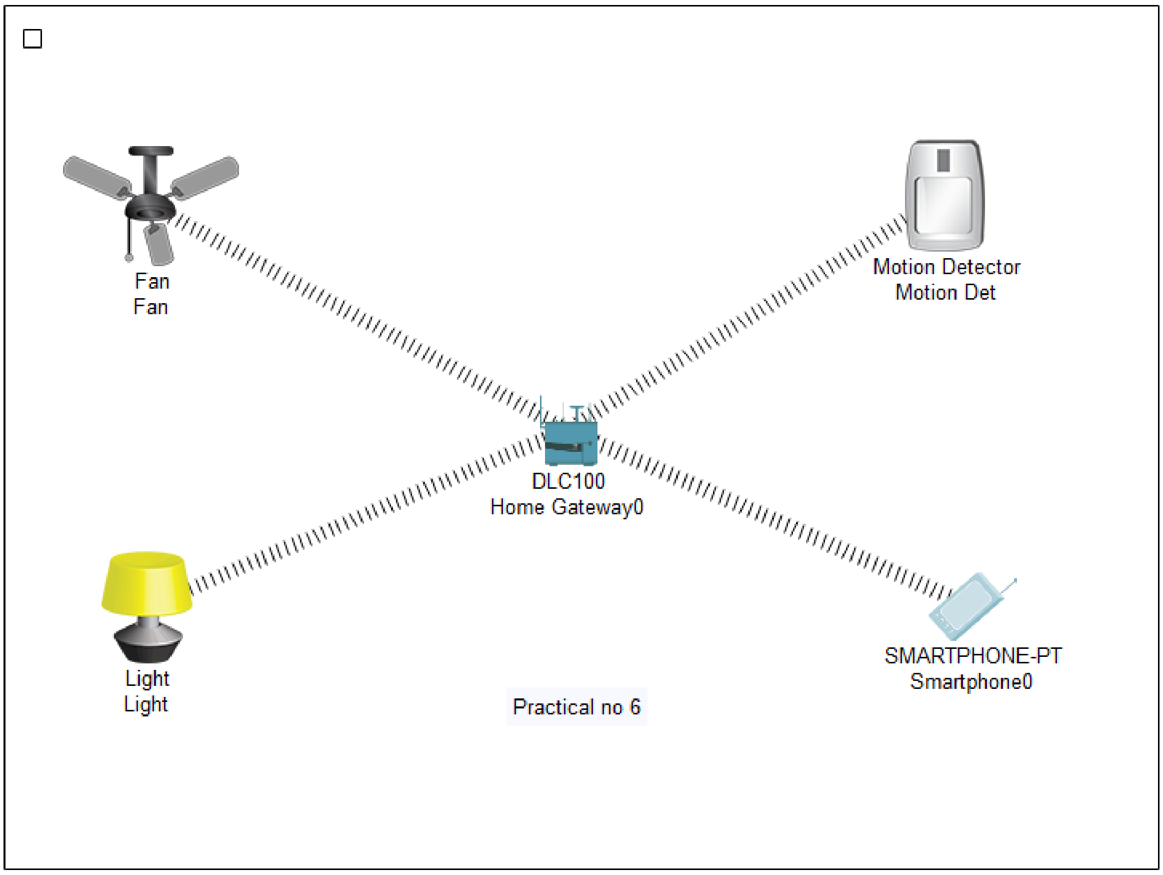
Create a basic MANET implementation simulation for Packet animation and Packet Trace.

B) DESCRIPTION:

A basic MANET (Mobile Ad-Hoc Network) simulation involves nodes moving randomly within a defined area, communicating with each other via wireless links. Packet animation visualizes the movement of packets between nodes, while packet tracing tracks the path taken by packets through the network, highlighting the dynamic routing and communication patterns within the ad-hoc network environment.

C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

Network topology(only for cisco packet tracer practical’s):



Configuration :-

Output: -

na

Wireless Sensor Networks & Mobile Communication

Practical No.7

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	22/1/24	**Practical No**	7

A) AIM:

Create MAC protocol simulation implementation for wireless sensor Network.

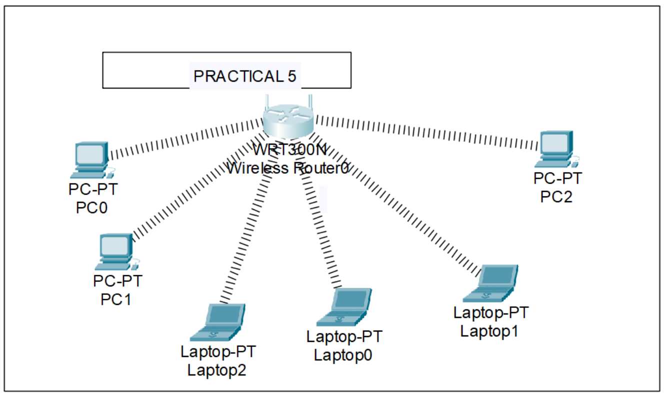
B) DESCRIPTION:

Develop a simulation model in NS-3 for the TDMA-based MAC protocol tailored for a wireless sensor network. Validate its performance metrics such as throughput and latency under varying network densities and traffic loads, aiming to optimize network efficiency and energy consumption

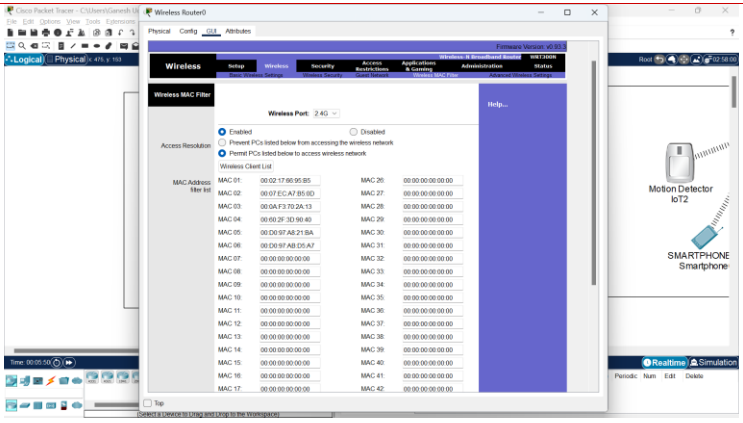
C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

Network Topology:

Star Topology



Configuration:



Wireless MAC Filter

Wireless Port: 2.4G

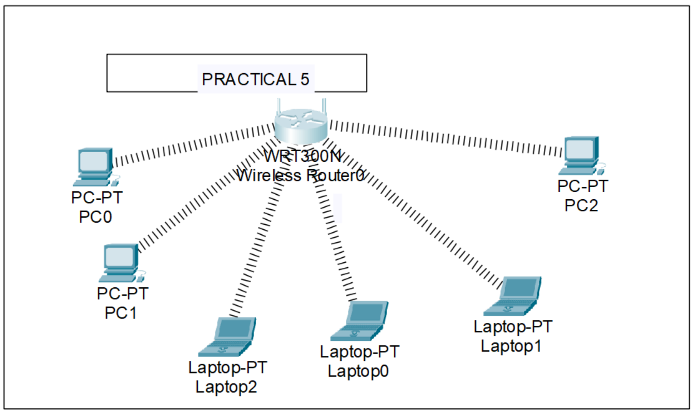
Access Resolution: [✓ Enabled] Disabled
Prevent PCs listed below from accessing the wireless network
[✓ Permit PCs listed below to access wireless network]

Wireless Client List

MAC Address filter list

MAC 01:	00:02:17:66:95:B5	MAC 26:	00:00:00:00:00:00
MAC 02:	00:07:EC:A7:B5:0D	MAC 27:	00:00:00:00:00:00
MAC 03:	00:0A:F3:70:2A:13	MAC 28:	00:00:00:00:00:00
MAC 04:	00:60:2F:3D:90:40	MAC 29:	00:00:00:00:00:00
MAC 05:	00:D0:97:A8:21:BA	MAC 30:	00:00:00:00:00:00
MAC 06:	00:D0:97:AB:D5:A7	MAC 31:	00:00:00:00:00:00
MAC 07:	00:00:00:00:00:00	MAC 32:	00:00:00:00:00:00
MAC 08:	00:00:00:00:00:00	MAC 33:	00:00:00:00:00:00
MAC 09:	00:00:00:00:00:00	MAC 34:	00:00:00:00:00:00
MAC 10:	00:00:00:00:00:00	MAC 35:	00:00:00:00:00:00
MAC 11:	00:00:00:00:00:00	MAC 36:	00:00:00:00:00:00
MAC 12:	00:00:00:00:00:00	MAC 37:	00:00:00:00:00:00
MAC 13:	00:00:00:00:00:00	MAC 38:	00:00:00:00:00:00
MAC 14:	00:00:00:00:00:00	MAC 39:	00:00:00:00:00:00
MAC 15:	00:00:00:00:00:00	MAC 40:	00:00:00:00:00:00
MAC 16:	00:00:00:00:00:00	MAC 41:	00:00:00:00:00:00
MAC 17:	00:00:00:00:00:00	MAC 42:	00:00:00:00:00:00

Output:



SIES

Wireless Sensor Networks & Mobile Communication

Practical No.8

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	29/1/24	**Practical No**	8

A) AIM:

Simulate Mobile Adhoc Network with Directional Antenna

B) DESCRIPTION:

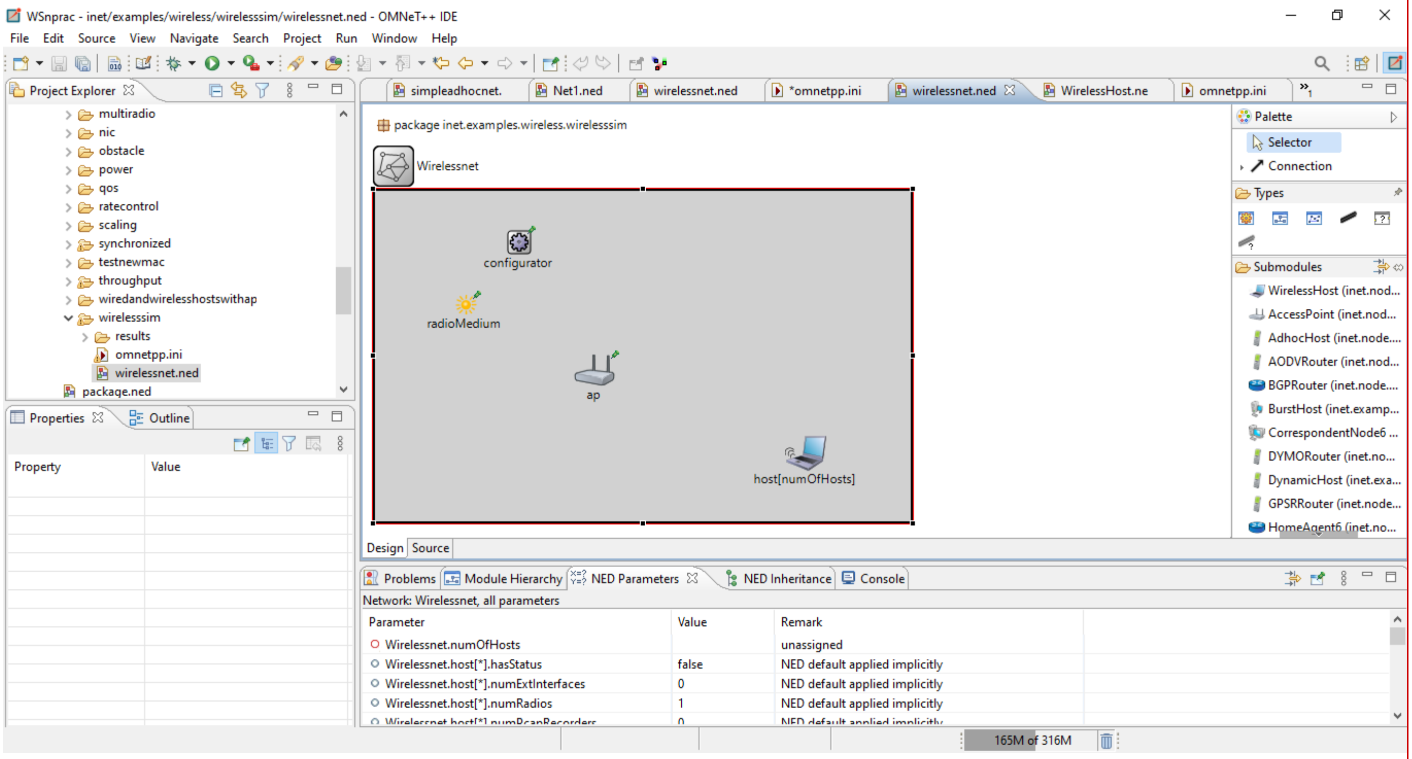
A wireless sensor network simulation involves deploying sensor nodes across a geographical area to collect and transmit data wirelessly to a central base station. Nodes monitor environmental conditions such as temperature, humidity, or pollution levels, and transmit this data to the base station for analysis or processing, enabling remote monitoring and control of the monitored area.

C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

Configurations:

```
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;

import inet.node.inet.WirelessHost;

import inet.node.wireless.AccessPoint;

import inet.physicallayer.contract.packetlevel.IRadioMedium;

import inet.physicallayer.ieee80211.packetlevel.Ieee80211ScalarRadioMedium;

network Wirelessnet

{

  parameters:

    int numOfHosts;
submodules:
  host[numOfHosts]: WirelessHost {
    @display("r=,,#707070");
  }

  ap: AccessPoint {
    @display("p=213,174;r=,,#707070");
  }
  configurator: IPv4NetworkConfigurator {
    @display("p=140,50");
  }
  radioMedium: Ieee80211ScalarRadioMedium {
    @display("p=88,111");}}
```

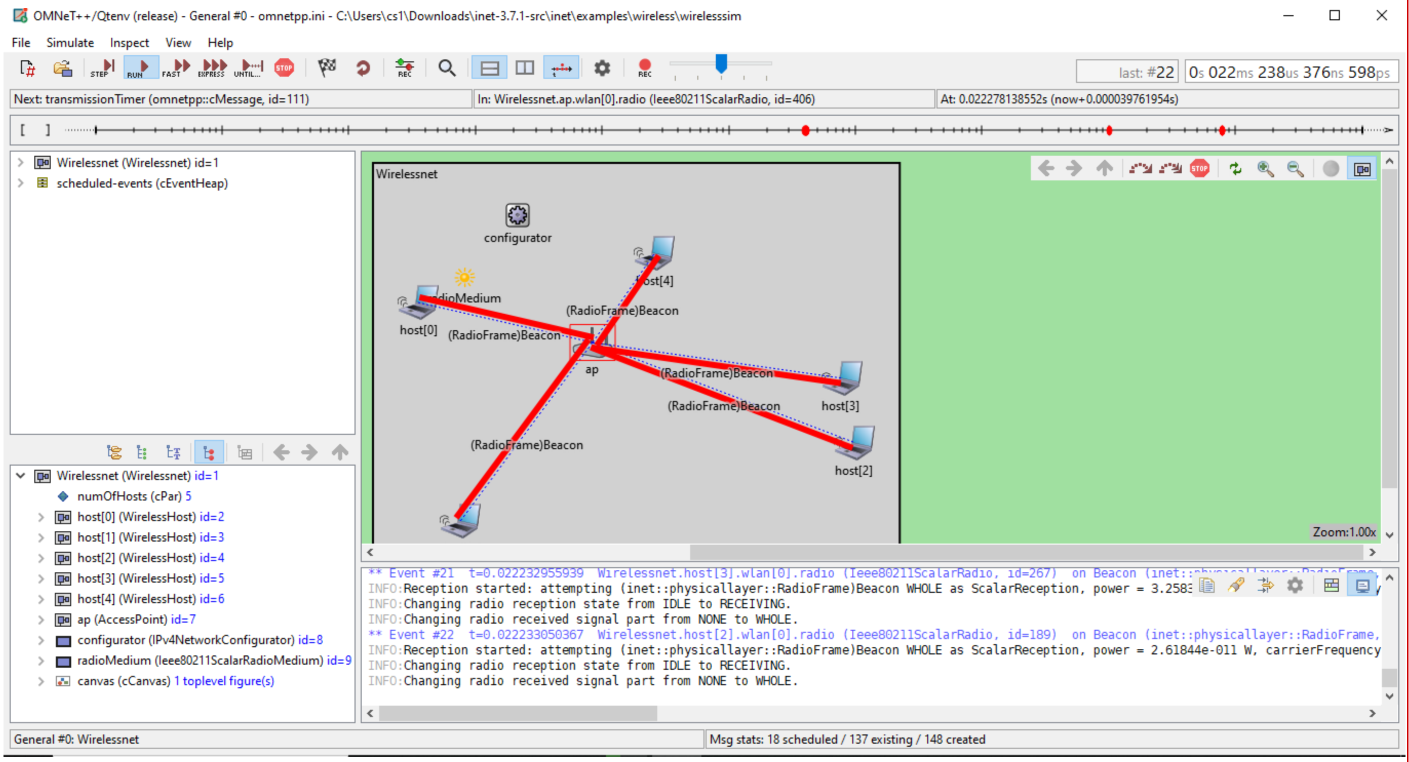
Output


WSnprac - inet/examples/wireless/wirelesssim/wirelessnet.ned - OMNeT++ IDE

package inet.examples.wireless.wirelesssim

Network: Wirelessnet, all parameters

Parameter	Value	Remark
Wirelessnet.numOfHosts		unassigned
Wirelessnet.host[*].hasStatus	false	NED default applied implicitly
Wirelessnet.host[*].numExtInterfaces	0	NED default applied implicitly
Wirelessnet.host[*].numRadios	1	NED default applied implicitly



OMNeT++/Qtenv (release) - General #0 - omnetpp.ini - C:\Users\cs1\Downloads\inet-3.7.1-src\inet\examples\wireless\wirelesssim

Next: transmissionTimer (omnetpp::cMessage, id=111) | In: Wirelessnet.ap.wlan[0].radio (Ieee80211ScalarRadio, id=406) | At: 0.022278138552s (now+0.000039761954s)

last: #22 | 0s 022ms 238us 376ns 598ps

- Wirelessnet (Wirelessnet) id=1
- scheduled-events (cEventHeap)

- Wirelessnet (Wirelessnet) id=1
  - numOfHosts (cPar) 5
  - host[0] (WirelessHost) id=2
  - host[1] (WirelessHost) id=3
  - host[2] (WirelessHost) id=4
  - host[3] (WirelessHost) id=5
  - host[4] (WirelessHost) id=6
  - ap (AccessPoint) id=7
  - configurator (IPv4NetworkConfigurator) id=8
  - radioMedium (Ieee80211ScalarRadioMedium) id=9
  - canvas (cCanvas) 1 toplevel figure(s)

```
** Event #21  t=0.022232955939  Wirelessnet.host[3].wlan[0].radio (Ieee80211ScalarRadio, id=267)  on Beacon (inet::physicallayer::RadioFrame,
INFO:Reception started: attempting (inet::physicallayer::RadioFrame)Beacon WHOLE as ScalarReception, power = 3.2583
INFO:Changing radio reception state from IDLE to RECEIVING.
INFO:Changing radio received signal part from NONE to WHOLE.
** Event #22  t=0.022233050367  Wirelessnet.host[2].wlan[0].radio (Ieee80211ScalarRadio, id=189)  on Beacon (inet::physicallayer::RadioFrame,
INFO:Reception started: attempting (inet::physicallayer::RadioFrame)Beacon WHOLE as ScalarReception, power = 2.61844e-011 W, carrierFrequency
INFO:Changing radio reception state from IDLE to RECEIVING.
INFO:Changing radio received signal part from NONE to WHOLE.
```

General #0: Wirelessnet | Msg stats: 18 scheduled / 137 existing / 148 created

Wireless Sensor Networks & Mobile Communication

Practical No.9

DEPARTMENT OF COMPUTER SCIENCE

Name:	Varun Vangari	**Roll Number**	TCS2324086
Paper Code:	SIUSCS64	**Class**	TYBSc(Computer Science)
Topic:	Wireless Sensor Networks & Mobile Communication	**Batch**	II
Date:	5/2/24	**Practical No**	9

A) AIM:

Create a mobile network using Cell Tower, Central Office Server, Web browser and Web Server.

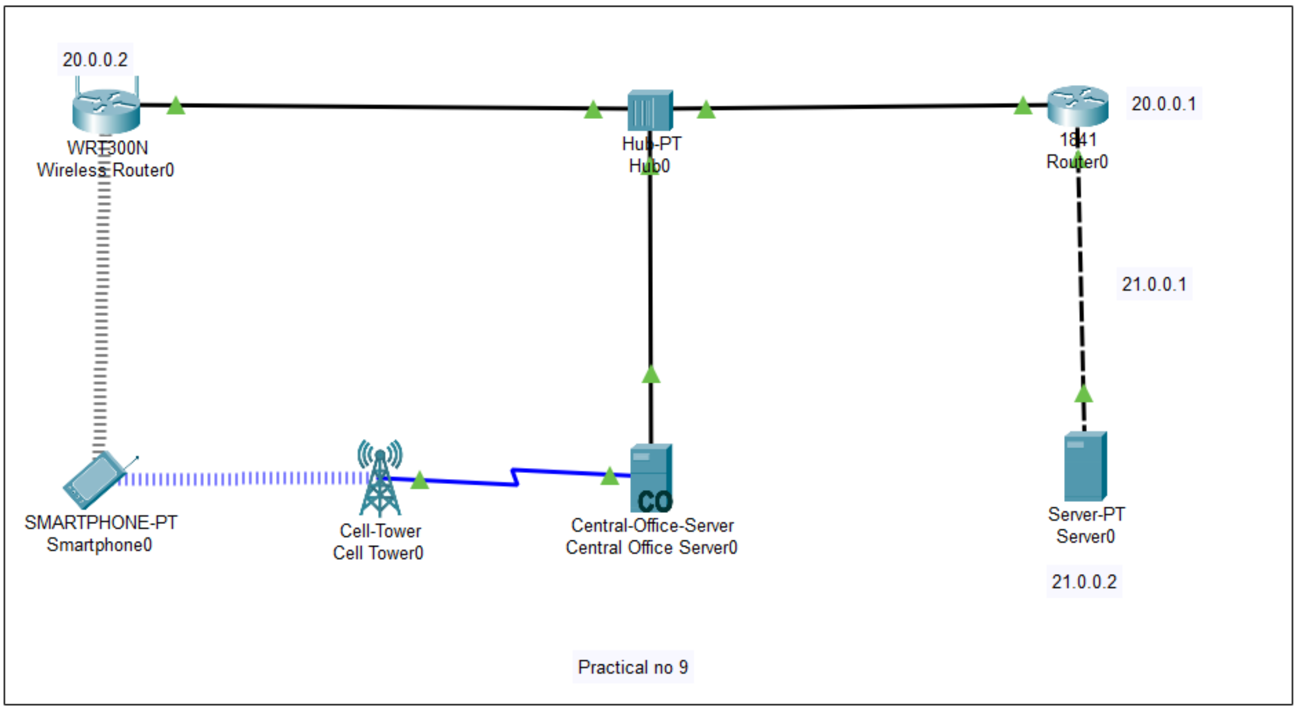
Simulate connection between them.

B) DESCRIPTION:

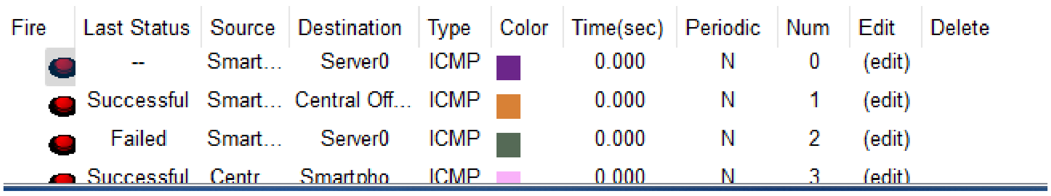
A mobile network is created using cell towers for signal transmission, central office servers for managing network operations, web browsers for user interaction, and web servers for hosting network services. The connection is simulated by cell towers relaying signals to central office servers, which communicate with web servers hosting content accessed by web browsers on mobile devices.

C) NETWORK TOPOLOGY, CONFIGURATIONS AND OUTPUT:

Network topology(only for cisco packet tracer practical’s):



Output :-



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	--	Smart...	Server0	ICMP	🟪	0.000	N	0	(edit)	
●	Successful	Smart...	Central Off...	ICMP	🟧	0.000	N	1	(edit)	
●	Failed	Smart...	Server0	ICMP	🟩	0.000	N	2	(edit)	
●	Successful	Centr	Smartpho	ICMP	🟪	0.000	N	3	(edit)	