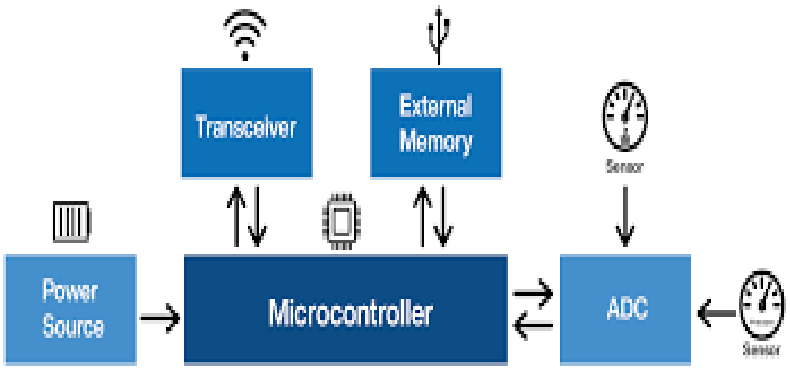
Understanding the Sensor Node Hardware

(For E.g. Sensors, Nodes (Sensor mote), Base Station, Graphical User Interface.)

Aim:

To create understand the Sensor Node Hardware and various types of Sensor nodes

Theory:



Sensor:

A sensor is a device or system that detects and measures a physical quantity or property and converts it into an electrical or optical signal that can be easily processed, stored, or transmitted. The physical quantity or property being measured can include temperature, pressure, humidity, light, sound, motion, and many others.

There are many types of sensors used for measuring physical quantities. Some common examples are as follows:

1) Temperature Sensors: These sensors measure the temperature of an object or environment. Examples of temperature sensors include thermocouples, thermistors, and RTDs (resistance temperature detectors).

2) Pressure Sensors: These sensors measure the pressure of a fluid or gas. Examples of pressure sensors include piezoelectric sensors, capacitive sensors, and strain gauges.

3) Motion Sensors: These sensors measure changes in motion or position. Examples of motion sensors include accelerometers, gyroscopes, and proximity sensors.

4) Light Sensors: These sensors measure the amount of light in an environment. Examples of light sensors include photoresistors, photodiodes, and phototransistors.

5) Magnetic Sensors: These sensors measure magnetic fields. Examples of magnetic sensors include Hall-effect sensors, magneto-resistive sensors, and fluxgate sensors.

6) Chemical Sensors: These sensors measure the concentration of a specific chemical or gas in an environment. Examples of chemical sensors include electrochemical sensors, gas sensors, and biosensors.

7) Humidity Sensors: These sensors measure the amount of moisture in the air. Examples of humidity sensors include capacitive sensors, resistive sensors, and thermal conductivity sensors.

8) Sound Sensors: These sensors measure sound waves. Examples of sound sensors include microphones, piezoelectric sensors, and accelerometers.

Temperature Sensor:

Temperature sensors are devices that detect and measure temperature, typically by converting temperature into an electrical signal that can be easily processed, stored, or transmitted. Temperature sensors are used in many applications, including Heating Ventilation and AC systems, industrial process control, medical devices, and consumer electronics.

Some examples of temperature sensors:

1) Thermocouples: Thermocouples are temperature sensors that consist of two different metals that are connected at two points. When the two points are at different temperatures, a voltage is generated that is proportional to the temperature difference. Thermocouples are commonly used in high-temperature applications.

2) Resistance Temperature Detectors (RTDs): RTDs are temperature sensors that operate on the principle that the electrical resistance of a material changes with temperature. RTDs typically use platinum as the sensing material, and their resistance changes linearly with temperature. RTDs are commonly used in industrial and laboratory applications where high accuracy and stability are required.

3) Thermistors: Thermistors are temperature sensors that operate on the principle that the electrical resistance of a material changes with temperature. Unlike RTDs, thermistors typically use ceramics as the sensing material, and their resistance changes non-linearly with temperature. Thermistors are commonly used in consumer electronics, automotive, and medical applications.

4) Infrared Temperature Sensors: Infrared temperature sensors are non-contact temperature sensors that detect infrared radiation emitted by an object and use it to calculate its temperature. Infrared temperature sensors are commonly used in industrial process control and medical applications where non-contact temperature measurement is required.

5) Bimetallic Temperature Sensors: Bimetallic temperature sensors consist of two strips of different metals that are bonded together. When the temperature changes, the two metals expand or contract at different rates, causing the sensor to bend. Bimetallic temperature sensors are commonly used in HVAC systems and other applications where low-cost temperature sensing is required.

Pressure Sensor:

Pressure sensors are devices that detect and measure pressure, typically by converting pressure into an electrical signal that can be easily processed, stored, or transmitted. Pressure sensors are used in many applications, including automotive, aerospace, medical devices, and industrial process control.

Some examples of pressure sensors:

1) Strain Gauge Pressure Sensors: Strain gauge pressure sensors use a metal strain gauge that changes resistance as it is stretched or compressed due to pressure. This change in resistance is converted into an electrical signal that is proportional to the pressure being applied. Strain gauge pressure sensors are commonly used in applications where high accuracy is required.

2) Capacitive Pressure Sensors: Capacitive pressure sensors use a diaphragm that is located between two plates. When pressure is applied, the diaphragm is deflected, causing a change in the distance between the two plates. This change in distance changes the capacitance of the sensor, which is then converted into an electrical signal that is proportional to the pressure being applied. Capacitive pressure sensors are commonly used in low-pressure applications.

3) Piezoelectric Pressure Sensors: Piezoelectric pressure sensors use a crystal that generates an electrical charge when it is subjected to pressure. This charge is then converted into an electrical signal that is proportional to the pressure being applied. Piezoelectric pressure sensors are commonly used in applications where high sensitivity is required.

4) Resonant Pressure Sensors: Resonant pressure sensors use a diaphragm that is located between two resonators. When pressure is applied, the resonant frequency of the sensor changes, which is then converted into an electrical signal that is proportional to the pressure being applied. Resonant pressure sensors are commonly used in high-pressure applications.

5) Optical Pressure Sensors: Optical pressure sensors use a fiber optic cable that is subjected to pressure. The pressure changes the refractive index of the cable, which is then measured using optical techniques. Optical pressure sensors are commonly used in applications where high accuracy and immunity to electromagnetic interference are required.

Humidity Sensor:

Humidity sensors are electronic devices that measure the amount of moisture in the air, also known as the relative humidity. They are widely used in various applications, including heating, ventilation and air conditioning (HVAC) systems, weather stations, and environmental monitoring.

There are several types of humidity sensors:

1) Capacitive humidity sensors: These sensors operate by measuring the change in capacitance of a material due to the absorption or release of moisture. They are highly accurate, reliable, and fast-responding.

2) Resistive humidity sensors: These sensors measure the change in resistance of a material due to changes in moisture content. They are typically less accurate than capacitive sensors, but they are less expensive and can be used in a wider range of applications.

3) Thermal conductivity humidity sensors: These sensors measure the change in thermal conductivity of a material due to changes in moisture content. They are commonly used in industrial applications and can operate at high temperatures.

4) Hair hygrometers: These mechanical sensors use a bundle of human or animal hair to measure humidity. The hair expands or contracts as the humidity changes, causing a mechanical movement that can be measured and converted into a humidity reading.

5) Dew point sensors: These sensors measure the dew point temperature, which is the temperature at which water vapor begins to condense into liquid form. They are commonly used in industrial and meteorological applications.

6) Gravimetric humidity sensors: These sensors measure the change in mass of a material due to the absorption or release of moisture. They are highly accurate but are typically more expensive and less practical for most applications.

Sensor Mote

A sensor mote is a small wireless device that integrates multiple sensors, a processing unit, and a communication unit, typically used in wireless sensor networks (WSNs). The term "mote" is derived from the word "remote", as these devices are designed to be deployed in remote or hard-to-reach locations where it may be difficult or impractical to run power or communication lines.

Sensor motes typically include a microcontroller, memory, radio transceiver, and a set of sensors, such as temperature, humidity, light, and motion sensors. They are usually powered by batteries and are designed to operate for extended periods of time, often years, without requiring a battery replacement. In addition to sensing environmental variables, sensor motes can also process and transmit the data to a central server or gateway using wireless communication protocols, such as ZigBee, Bluetooth, or Wi-Fi.

Sensor motes have a wide range of applications, including environmental monitoring, industrial process control, agriculture, and healthcare. They can be used to collect and transmit real-time data on temperature, humidity, pressure, vibration, and other physical variables, allowing for better decision-making, predictive maintenance, and process optimization.

Sensor motes are an important component of the Internet of Things (IoT) and are increasingly being used to create intelligent systems and applications that can monitor and control a wide range of physical environments.

Base Station:

In wireless sensor networks (WSNs), a base station is a central device that acts as a gateway or sink for data collected by the sensors in the network. The base station is responsible for collecting data from multiple sensors, processing the data, and transmitting it to a remote location for further analysis.

The base station typically has a more powerful processor and larger memory than the individual sensors, and is capable of running more sophisticated algorithms for data processing and

analysis. It may also have a more powerful communication unit than the individual sensors, allowing it to transmit data over longer distances or using higher-bandwidth wireless protocols.

The base station is usually connected to a power source and a network, and may be located in a central location that is easily accessible for maintenance and management. The base station may also be responsible for managing the network, including routing data between the sensors and the base station, controlling power consumption of the sensors, and detecting and mitigating network failures.

The base station plays a critical role in the operation of WSNs, as it enables the collection and analysis of data from multiple sensors, allowing for better decision-making and optimization of various processes. The design of the base station can have a significant impact on the performance and efficiency of the WSN, including the range of the network, the reliability of the data, and the power consumption of the sensors.

GUI (Graphical User Interface)

While sensor nodes do not typically have a traditional GUI, some development environments may include graphical tools or interfaces that allow developers to visualize and debug the operation of the sensor nodes. For example, some development environments for sensor networks provide visual editors for designing the network topology, configuring the sensors, and visualizing the data flow.

In addition, some sensor nodes may have simple built-in user interfaces, such as LEDs or buttons, that allow basic interaction with the device. For example, LEDs may be used to indicate the status of the device, while buttons may be used to initiate various operations or change the configuration of the device.

Overall, the design of the user interface for WSNs is typically focused on providing a simple and efficient means for developers to program and interact with the sensor nodes, rather than providing a traditional GUI for end-users.

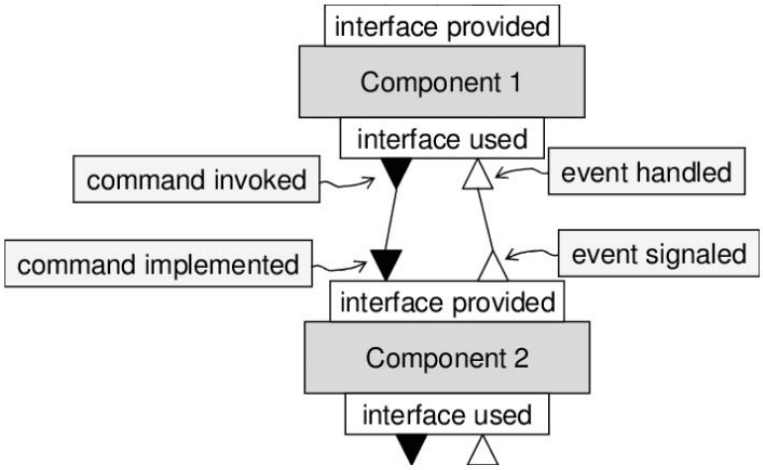
Exploring and understanding TinyOS computational concepts: - Events, Commands and Task.

(nesC model, nesC Components)

Aim:

To understand TinyOS computational concepts through nesC programming language

Theory:



NesC (Network embedded systems C)

A NesC is a programming language that was developed specifically for wireless sensor networks (WSNs). It was created at the University of California, Berkeley, and is based on the C programming language.

NesC is designed to be a component-based language, meaning that it is structured around modular components that can be combined to form larger programs. This makes it well-suited for developing applications for WSNs, which often involve a large number of small, specialized devices working together.

One of the key features of NesC is its support for a programming model known as "event-driven programming". In this model, programs are structured around events and the handlers that respond to them. This approach is particularly useful in WSNs, where devices often need to respond to events such as changes in the environment or changes in the network topology.

NesC also includes features for low-level programming, such as direct memory access and hardware control, which are important in the resource-constrained environment of WSNs.

Overall, NesC is a powerful and flexible programming language that is specifically designed for the unique challenges of developing applications for wireless sensor networks.

NesC Model

The NesC model refers to the programming model used in the NesC programming language. The NesC model is based on a component-based architecture, in which programs are constructed from reusable software components.

In the NesC model, a component is a self-contained module that performs a specific function or provides a specific service. Components can be combined to create larger programs, and the connections between components are defined by interfaces. Interfaces define the methods and signals that are used to interact with a component, and provide a way for components to communicate with each other.

The NesC model also includes support for event-driven programming, which is a programming paradigm in which programs are structured around events and the handlers that respond to them. In the NesC model, events are defined as signals, and the handlers that respond to them are defined as tasks.

Overall, the NesC model provides a powerful and flexible way to develop complex programs for wireless sensor networks by using modular, reusable components and an event-driven programming approach.

NesC Components

In the NesC programming language, a component is a self-contained module that performs a specific function or provides a specific service. NesC components are designed to be highly modular and reusable, which makes it easy to create complex programs by combining and connecting different components together.

Some common types of components in NesC include:

1) Modules: These are the basic building blocks of NesC programs. A module is a self-contained unit of code that defines a single functionality or a single interface. Modules can include functions, variables, and interfaces that other modules can use.

2) Configurations: A configuration is a group of modules and their interconnections that make up a complete program. Configurations define how modules are connected to each other and how they interact to achieve a specific task.

3) Interfaces: Interfaces define the methods and signals that are used to interact with a component. They provide a way for components to communicate with each other, and allow for components to be developed independently and easily swapped out or modified as needed.

4) Components Libraries: Components libraries provide pre-built and reusable components that can be easily included in a program. They can be used to simplify programming tasks, speed up development, and ensure the quality of the code.

5) Libraries: NesC also supports the use of standard C libraries, which can be used to provide additional functionality to NesC programs. Standard C libraries can be used to perform tasks such as memory allocation, string manipulation, and math operations.

Overall, NesC components provide a powerful way to create modular, reusable code that can be easily combined and connected to build complex programs for wireless sensor networks.

Understanding TOSSIM for

- Mote-mote radio communication

- Mote-PC serial communication

Aim:

To understand TOSSIM for mote-mote radio communication and mote-PC serial communication

Theory:

TOSSIM for mote-mote radio communication

TOSSIM (TinyOS Simulation) is a software tool used for simulating wireless sensor networks (WSNs) that use TinyOS, an open-source operating system designed for low-power wireless devices such as motes. Motes are small wireless nodes that can collect and transmit data in a WSN.

TOSSIM allows researchers and developers to simulate the behavior of a WSN, including mote-mote radio transmissions, in a virtual environment. This is useful for testing and debugging new algorithms, protocols, and applications before deploying them on physical hardware.

In the context of mote-mote radio transmission, TOSSIM can simulate the behavior of the radio hardware and network protocols used by the motes. This includes modeling the signal strength, packet loss, and interference that can occur in a real-world wireless network. By simulating mote-mote radio transmission in TOSSIM, developers can test and optimize their network protocols and algorithms without the need for physical hardware, which can be expensive and time-consuming to set up.

TOSSIM can also be used to evaluate the performance of different types of motes and radio hardware. For example, developers can simulate the behavior of a network using different types of motes and radio hardware to see how they perform under different conditions.

Overall, TOSSIM is a powerful tool for simulating and testing mote-mote radio transmission in WSNs, helping developers and researchers to optimize the performance of their network protocols and algorithms before deploying them on physical hardware.

TOSSIM and TinyOS

TOSSIM is a software tool used for simulating wireless sensor networks (WSNs) that use TinyOS, an open-source operating system designed for low-power wireless devices such as motes. TinyOS provides a framework for developing WSN applications and protocols, and it includes a set of built-in components and libraries for managing mote hardware, communication, and power consumption.

TOSSIM is specifically designed to work with TinyOS and provides a simulation environment that emulates the behavior of a WSN running on top of TinyOS. This means that TOSSIM includes an implementation of TinyOS that can run on a host computer and simulate the behavior of motes and their interactions with each other in a virtual environment.

TOSSIM is closely integrated with TinyOS, and it provides a set of simulation tools and APIs that allow developers to write and test TinyOS applications and protocols in a simulated environment.

For example, developers can use the TOSSIM APIs to simulate mote hardware, radio transmissions, and other aspects of a WSN running on TinyOS.

Overall, TOSSIM and TinyOS are tightly coupled, and they provide a powerful platform for developing, testing, and deploying WSN applications and protocols. Together, they enable developers to prototype and optimize WSNs in a simulated environment before deploying them on physical hardware.

TOSSIM for mote-PC serial communication

In addition to simulating mote-mote radio transmission, TOSSIM can also be used to simulate Mote-PC serial communication in a virtual environment.

Mote-PC serial communication involves exchanging data between a mote and a personal computer (PC) through a serial port. This is typically done using a USB-to-serial converter that connects the mote to the PC.

In a TOSSIM simulation, mote-PC serial communication can be simulated using the SerialForwarder component provided by TinyOS. The SerialForwarder component runs on the mote and forwards data received from the mote's serial port to the simulated PC running on the host computer. The PC can then send data back to the mote by writing to the simulated serial port.

Simulating mote-PC serial communication in TOSSIM is useful for testing and debugging applications that rely on serial communication between motes and a PC, such as data collection and visualization applications. It allows developers to test their applications in a simulated environment before deploying them on physical hardware, which can be expensive and time-consuming to set up.

Overall, TOSSIM provides a comprehensive simulation platform for WSNs, including mote-mote radio transmission and mote-PC serial communication, which enables developers to prototype, test, and optimize their applications and protocols in a simulated environment before deploying them on physical hardware**.**

Create and simulate a simple ad hoc network

Aim:

To create a simple ad hoc network and perform its simulation with the required number of hosts

Software Used:

Omnetpp 5.7, INET 4.3.6 framework

Theory:

An ad hoc network is one that is spontaneously formed when devices connect and communicate with each other.
Ad hoc networks are mostly wireless local area networks (LANs).
The devices communicate with each other directly instead of relying on a base station or access points as in wireless LANs for data transfer co-ordination.
Each device participates in routing activity, by determining the route using the routing algorithm and forwarding data to other devices via this route.

Types of Wireless Ad Hoc Networks

Wireless ad hoc networks are categorized into classes. Here are a few examples:

1) Mobile ad hoc network (MANET): An ad hoc network of mobile devices.
2) Vehicular ad hoc network (VANET): Used for communication between vehicles. Intelligent VANETs use artificial intelligence and ad hoc technologies to communicate what should happen during accidents.
3) Smartphone ad hoc network (SPAN): Wireless ad hoc network created on smartphones via existing technologies like Wi-Fi and Bluetooth.
4) Wireless mesh network: A mesh network is an ad hoc network where the nodes communicate directly with each other to relay information throughout the network.
5) Army tactical MENT: Used in the army for "on-the-move" communication, a wireless tactical ad hoc network relies on range and instant operation to establish networks when needed.
6) Wireless sensor network: Wireless sensors that collect everything from temperature and pressure readings to noise and humidity levels can form an ad hoc network to deliver information to a home base without needing to connect directly to it.
7) Disaster rescue ad hoc network: Ad hoc networks are important when disaster strikes and established communication hardware isn't functioning properly.

Limitations of Ad Hoc Wireless Network

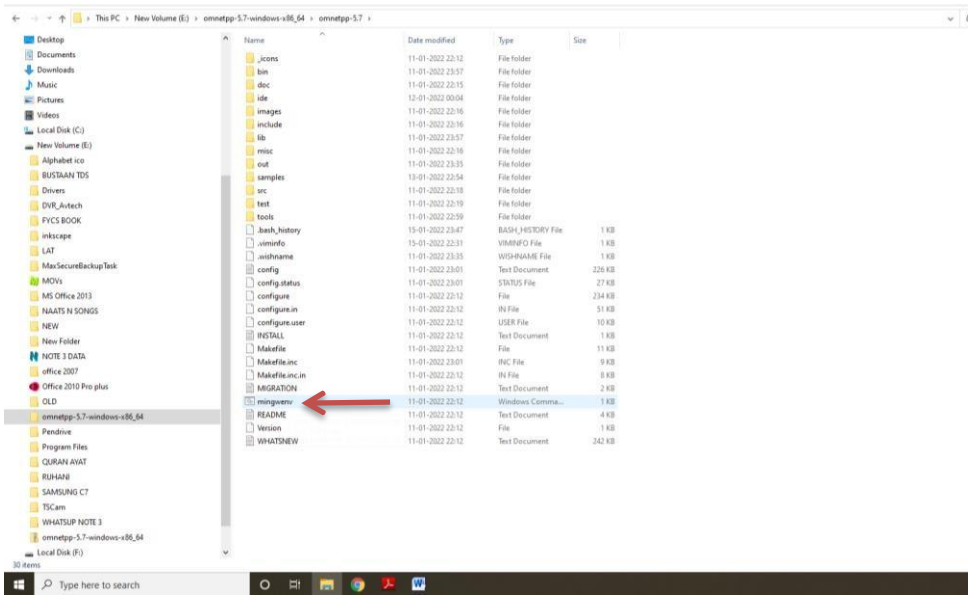
1) For file and printer sharing, all users need to be in the same workgroup, or if one computer is joined to a domain, the other users must have accounts on that computer to access shared items.
2) Other limitations of ad hoc wireless networking include the lack of security and a slow data rate. Ad hoc mode offers minimal security; if attackers come within range of your ad hoc network, they won't have any trouble connecting.

We create an Ad hoc network with 7 hosts using Omnetpp and INET through the following steps

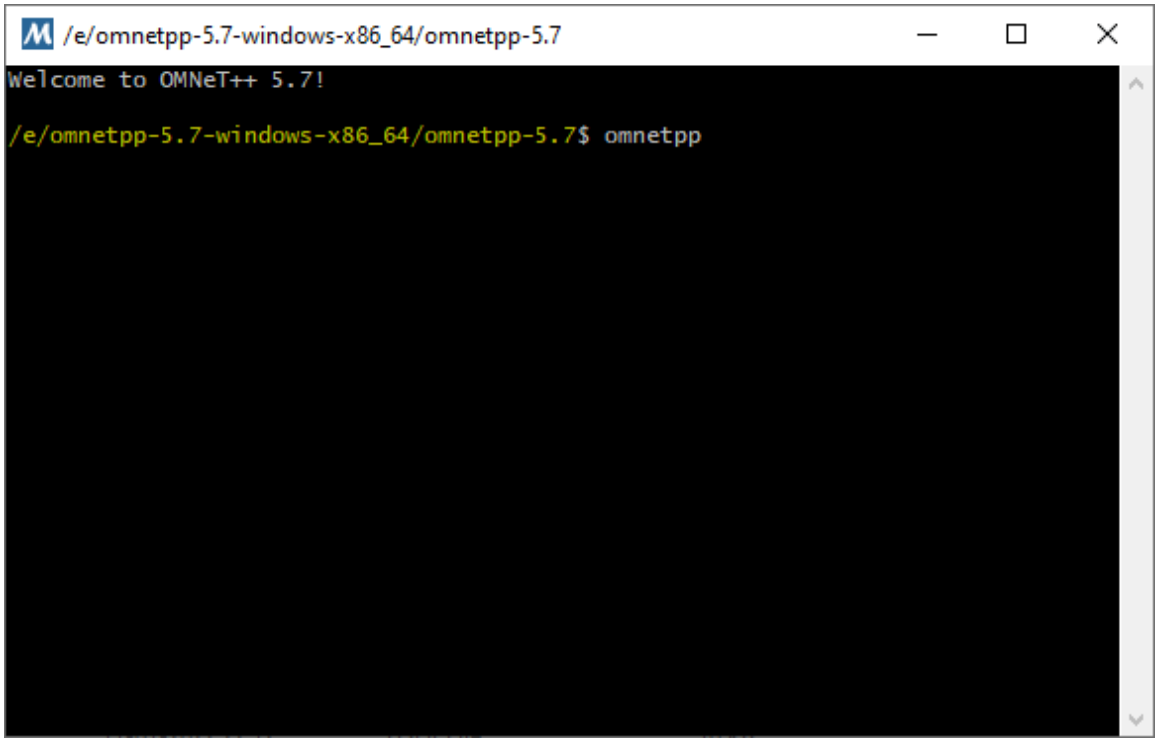
Note: we don't need to do the first step in prac exam as we can start it normally.

Step 1: Start the Omnetpp simulator through the following procedure

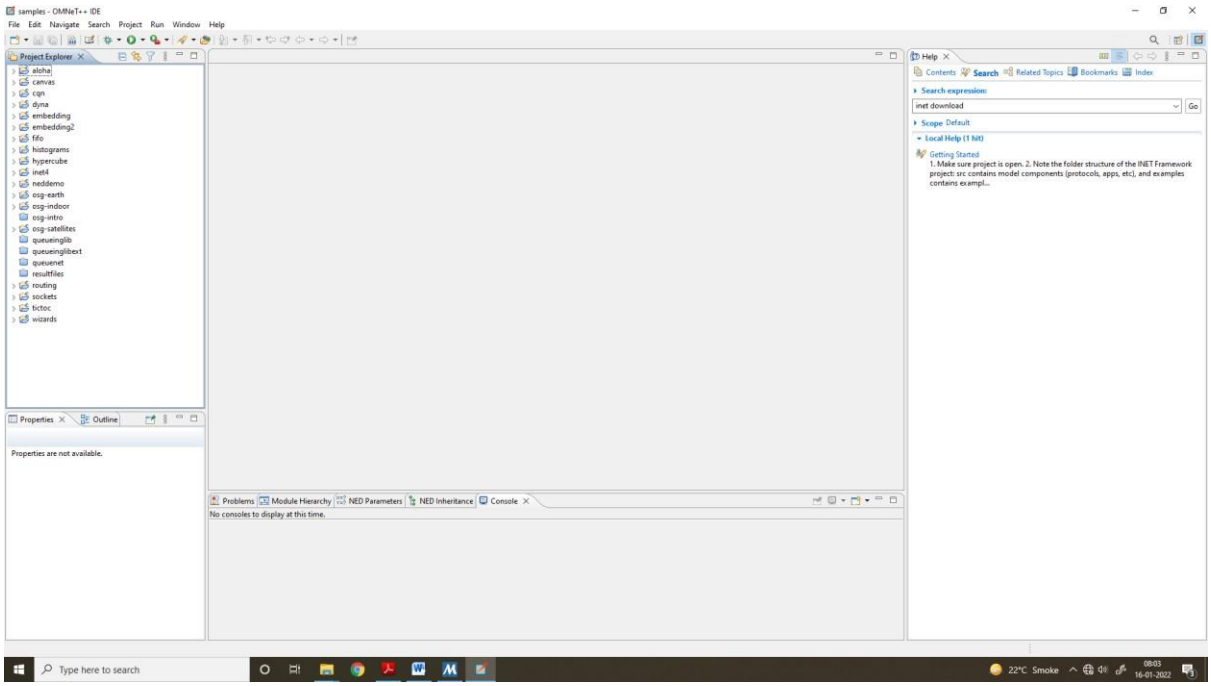
Click on the file mingwenv



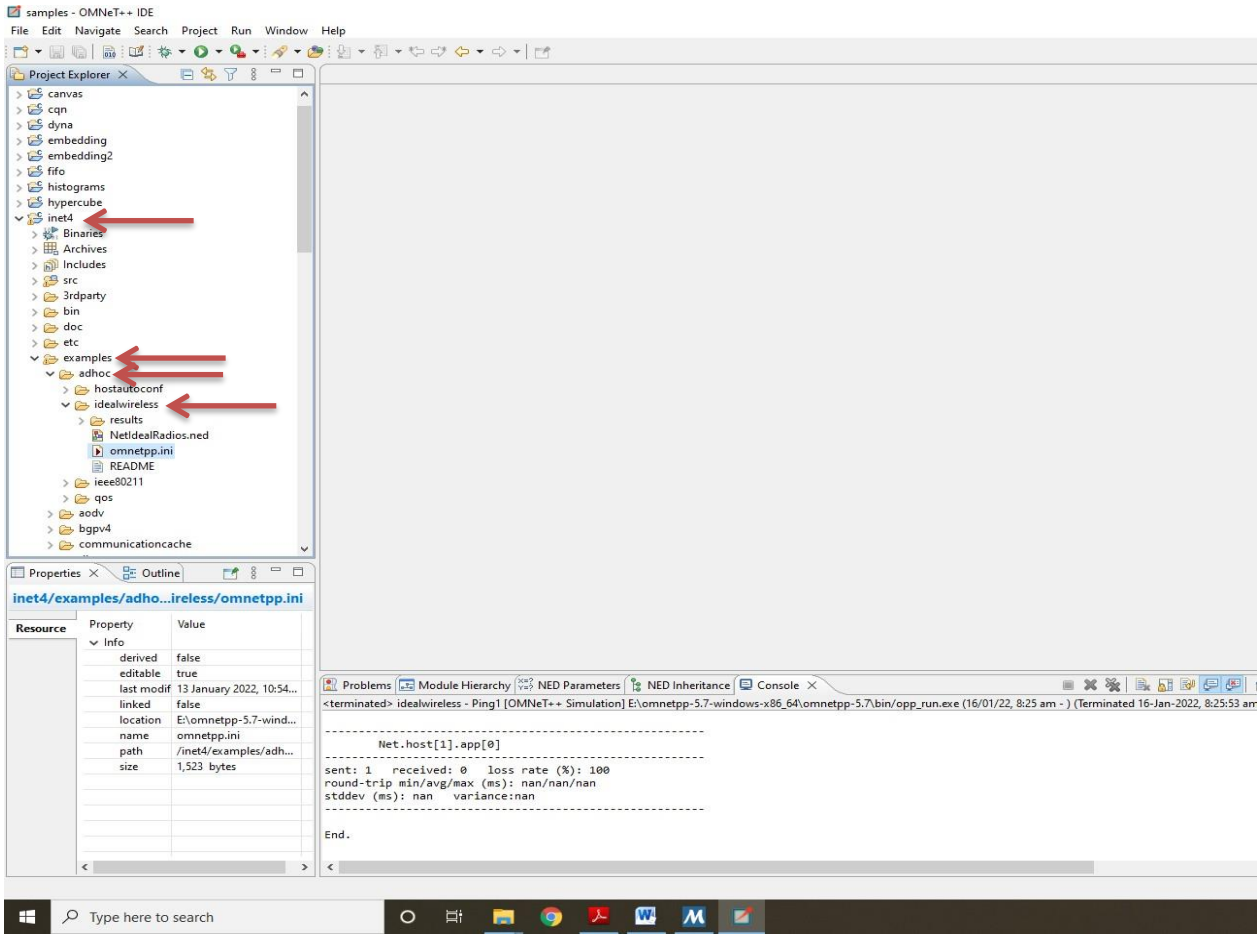
We will get the following $ prompt, type omnetpp and enter



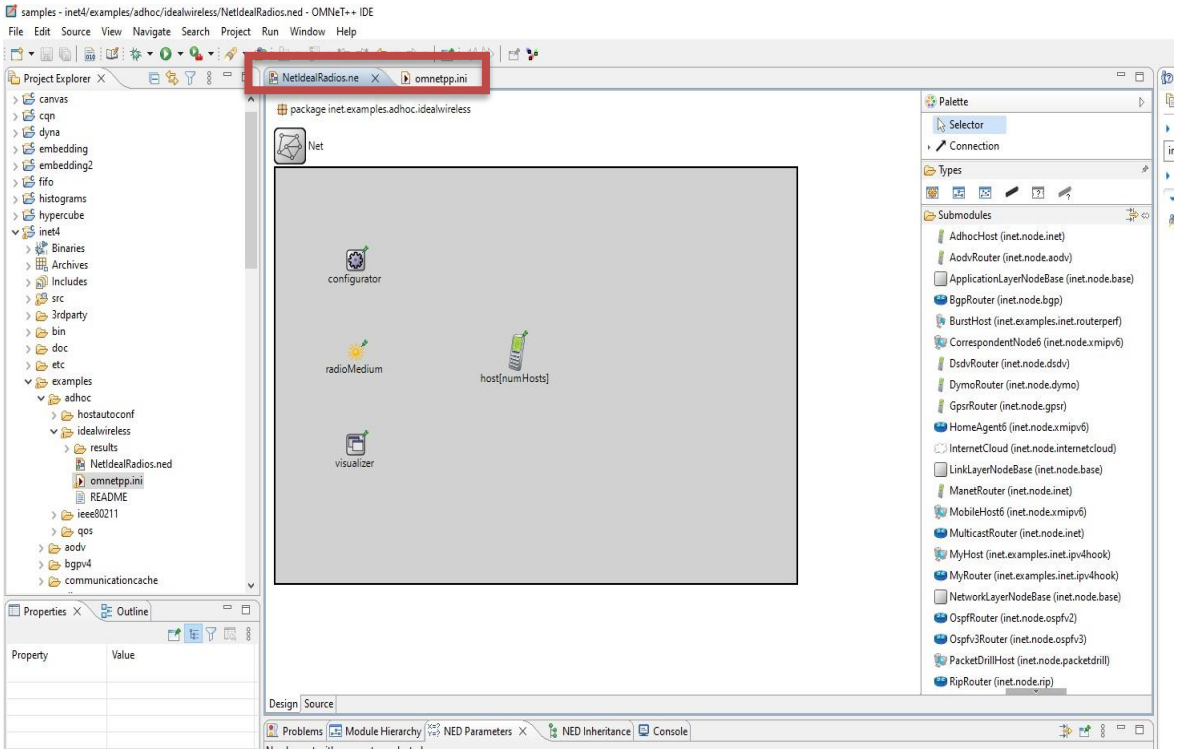
Step 2: The omnetpp simulator in now ready with the following user interface



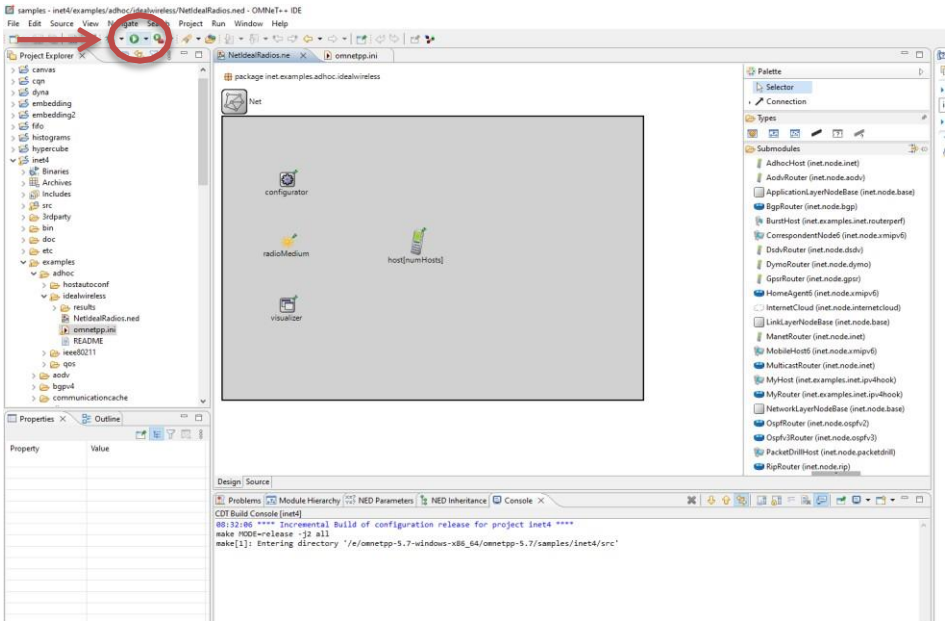
Click on inet folder, then in it click on examples, then on adhoc and then on idealwireless as given



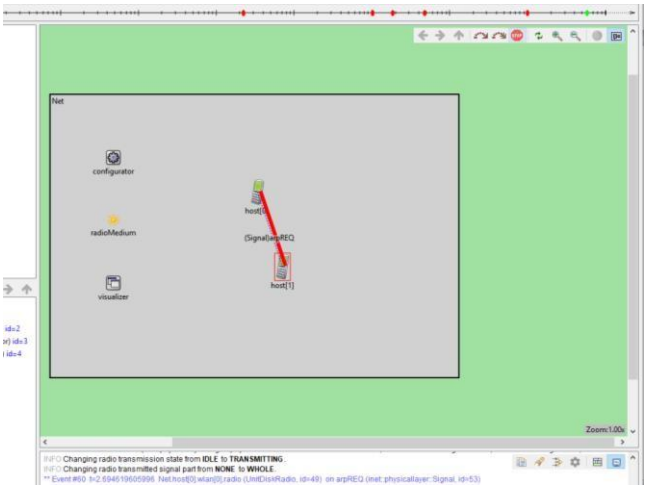
Step 3: In order to load the simulation, double click on two files NetIdealRadios.ned and omnetpp.ini



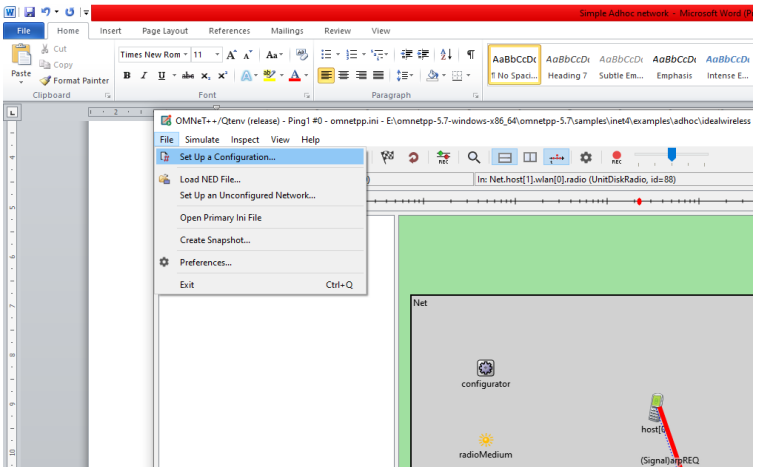
Step 4: Now we run the simulation



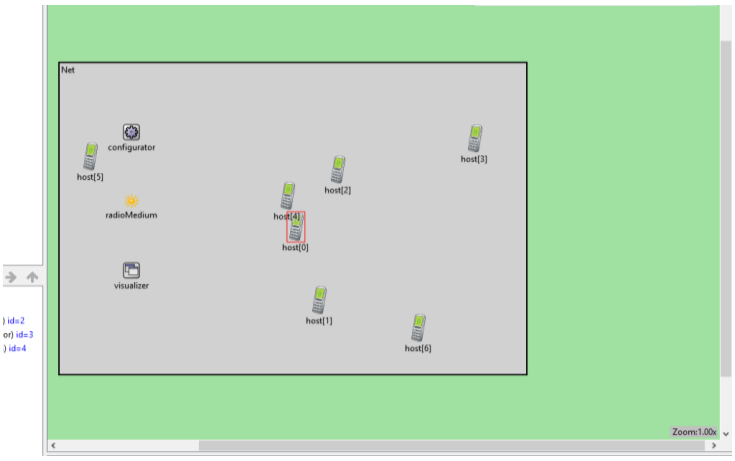
Step 5: After running the simulation we get the following



The number of hosts can be increased by the following



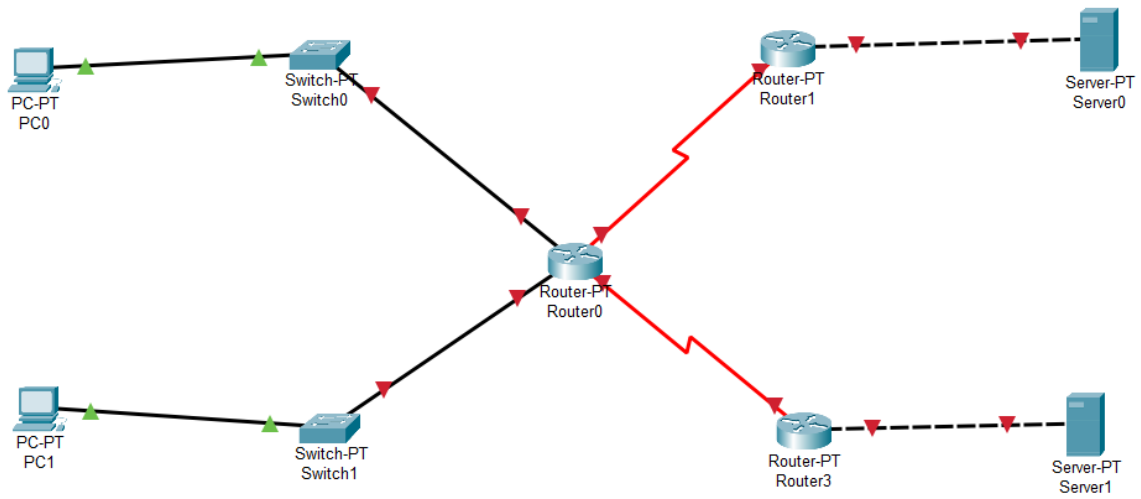
In this we get a dropdown menu, select n host option and enter the required hosts
The following simulation has 7 hosts



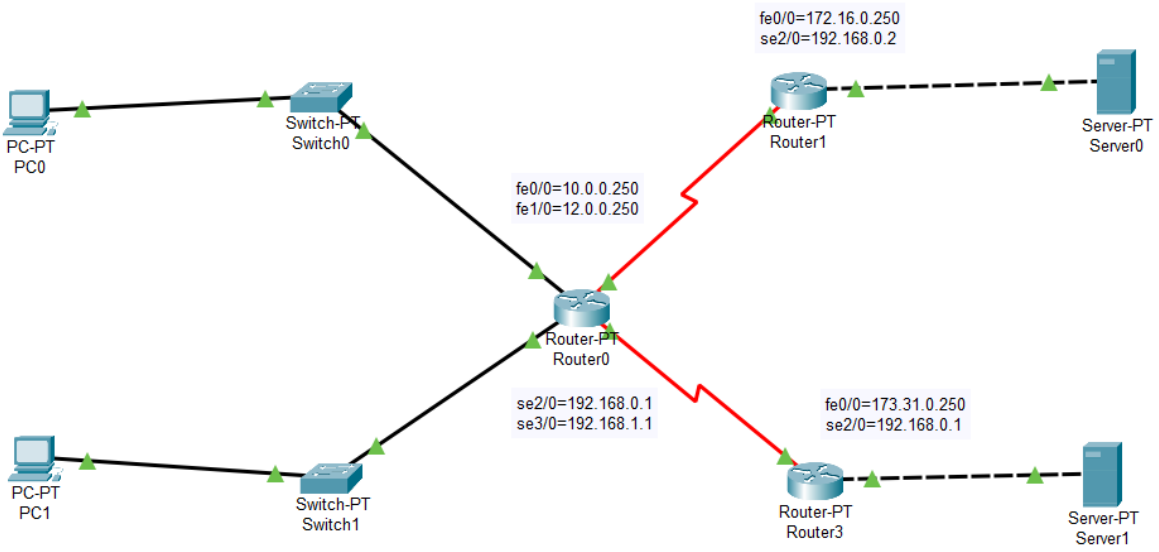
Practical 5

Aim: Understanding , Reading and Analyzing Routing Table of a network.

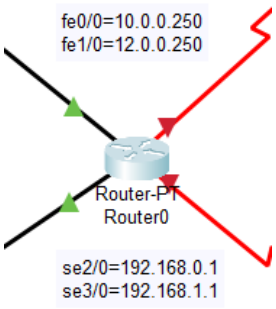
Step 1: Basic setup □

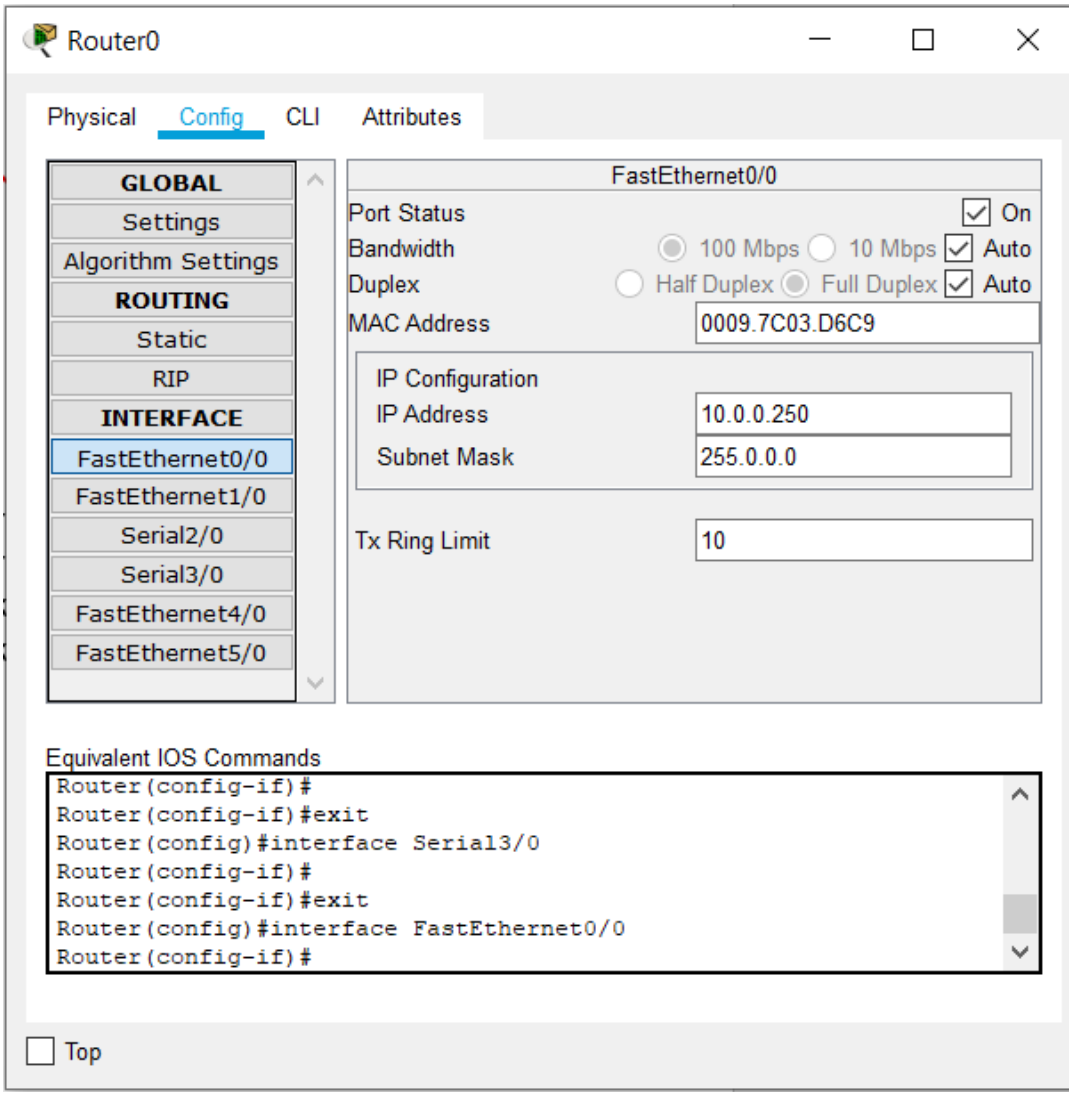


The below image is just for quick reference



Step 2: Click on Router 0, then go to config and make the following changes:
FastEthernet0/0 → IP address : 10.0.0.250 & hit enter & tick the On checkbox
FastEthernet1/0 → IP address : 12.0.0.250 & hit enter & tick the On checkbox
Serial2/0 → IP address : 192.168.0.1 & hit enter & tick the On checkbox
Serial3/0 → IP address : 192.168.1.1 & hit enter & tick the On checkbox

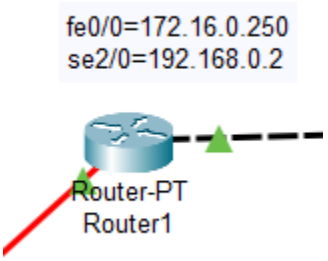




Step 3: Similarly for Router 1

FastEthernet0/0 → IP address : 172.16.0.250 & hit enter & tick the On checkbox

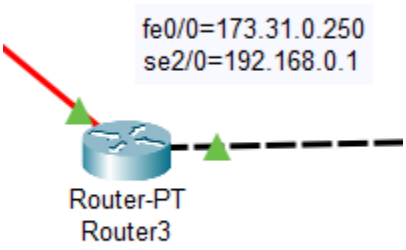
Serial2/0 → IP address : 192.168.0.2 & hit enter & tick the On checkbox



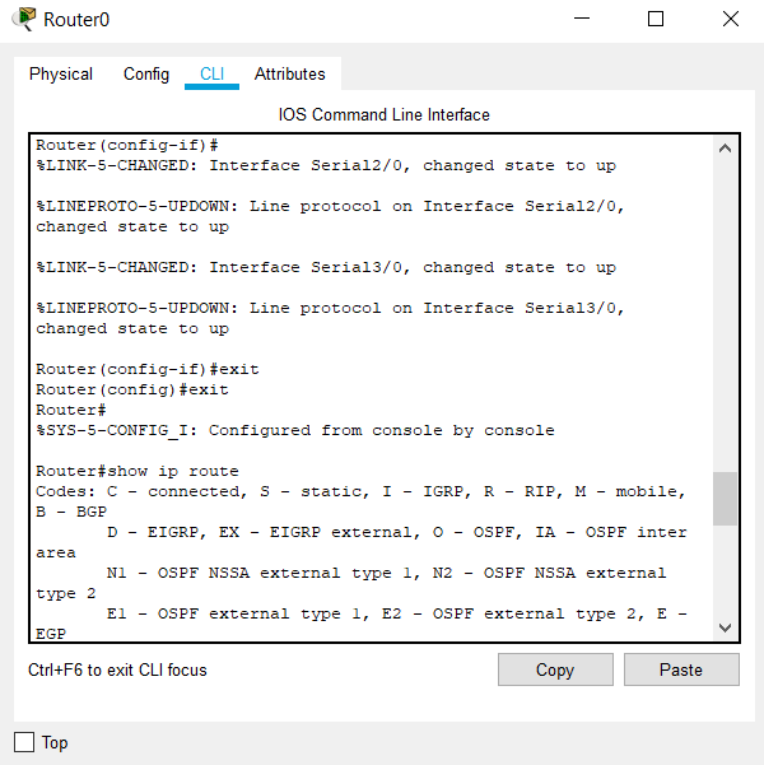
Step 4: Similarly for Router 3

FastEthernet0/0 → IP address : 173.31.0.250 & hit enter & tick the On checkbox

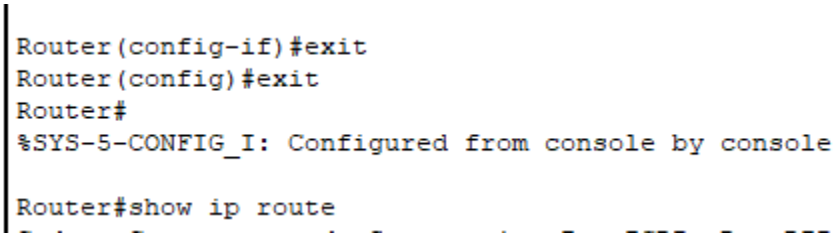
Serial2/0 → IP address : 192.168.0.1 & hit enter & tick the On checkbox



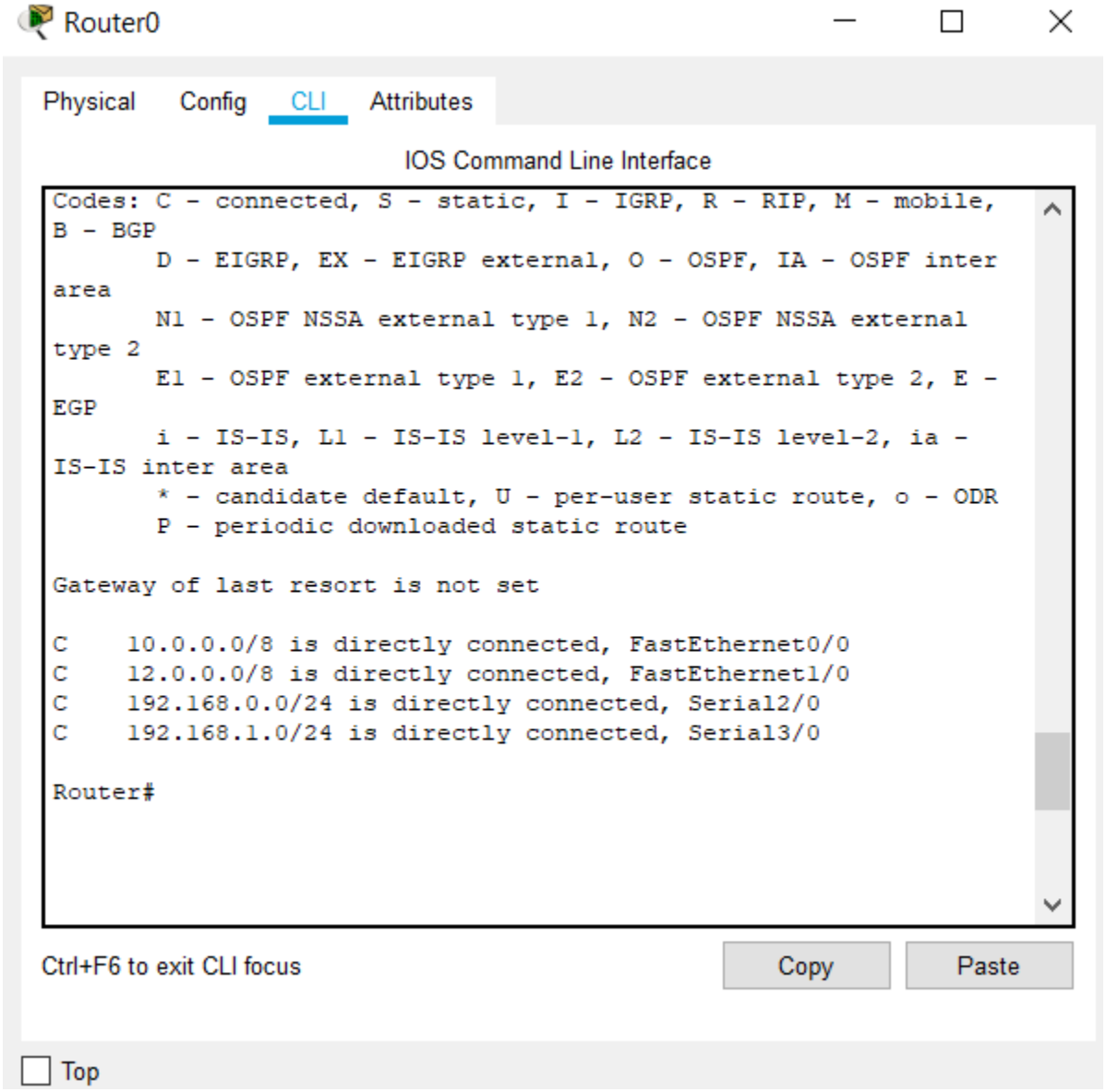
Step 5: Now click on Router 0 and go to CLI



In CLI write **exit** until you come to the **Router#** (Here we have written exit twice and then pressed enter)
Then write the command: show ip route



You will get the output like this:



MANET implementation simulation

Aim:

To create a basic MANET implementation simulation for Packet animation and Packet Trace

Software Used:

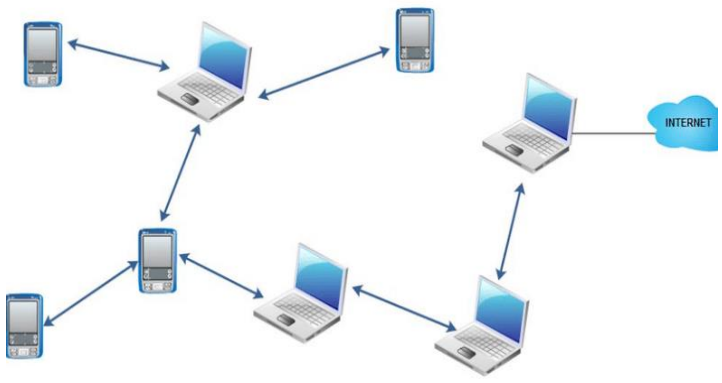
Omnet++ 5.7, INET 4.3.6 framework

Theory:

1) MANET (Mobile Adhoc NETwork) also called a wireless Adhoc network or Adhoc wireless network that usually has a routable networking environment on top of a Link Layer ad hoc network
2) They consist of a set of mobile nodes connected wirelessly in a self-configured, self-healing network without having a fixed infrastructure.
3) MANET nodes are free to move randomly as the network topology changes frequently.
4) Each node behaves as a router as they forward traffic to other specified nodes in the network.
5) MANET may operate a standalone fashion or they can be part of larger internet.
6) They form a highly dynamic autonomous topology with the presence of one or multiple different transceivers between nodes.
7) The main challenge for the MANET is to equip each device to continuously maintain the information required to properly route traffic.

The following are some of the important characteristics of MANET

1) Dynamic Topologies
2) Bandwidth constrained, variable capacity links:
3) Autonomous Behavior:
4) Energy Constrained Operation:
5) Limited Security:
6) Less Human Intervention:



A Typical example of MANET

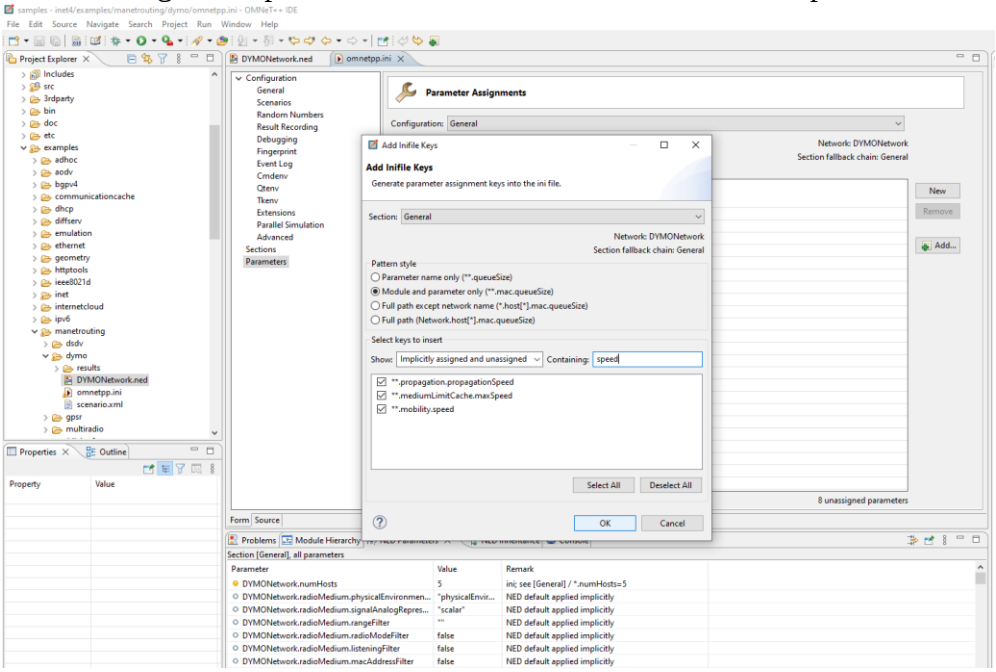
Simple steps:

1.open omnet++ then click on -

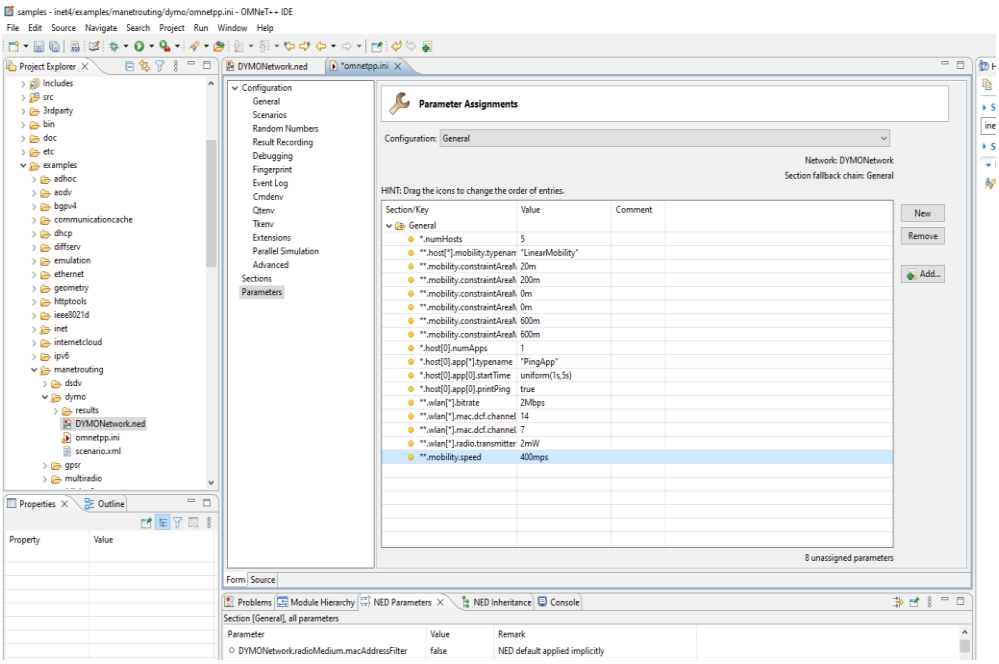
INET → examples → manetrouting → dymo and then load the DYMONetwork.ned and omnetpp.ini files by double clicking on them

2. Select omnetpp.ini file and click on parameters, we need to add mobility to the nodes

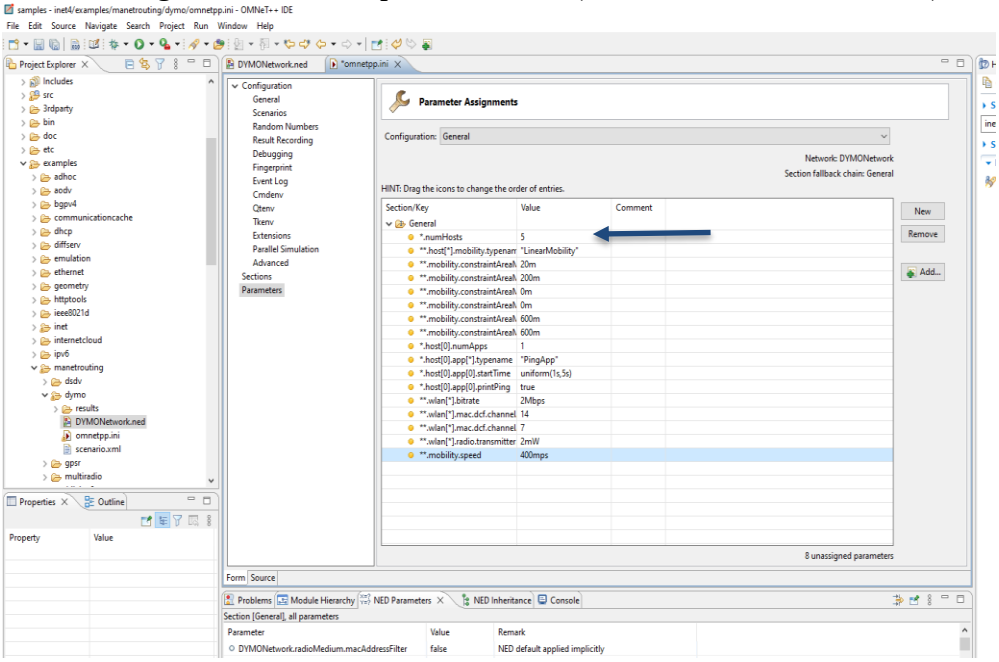
3. For adding a new parameter click on add button and add the parameter **.mobility.speed



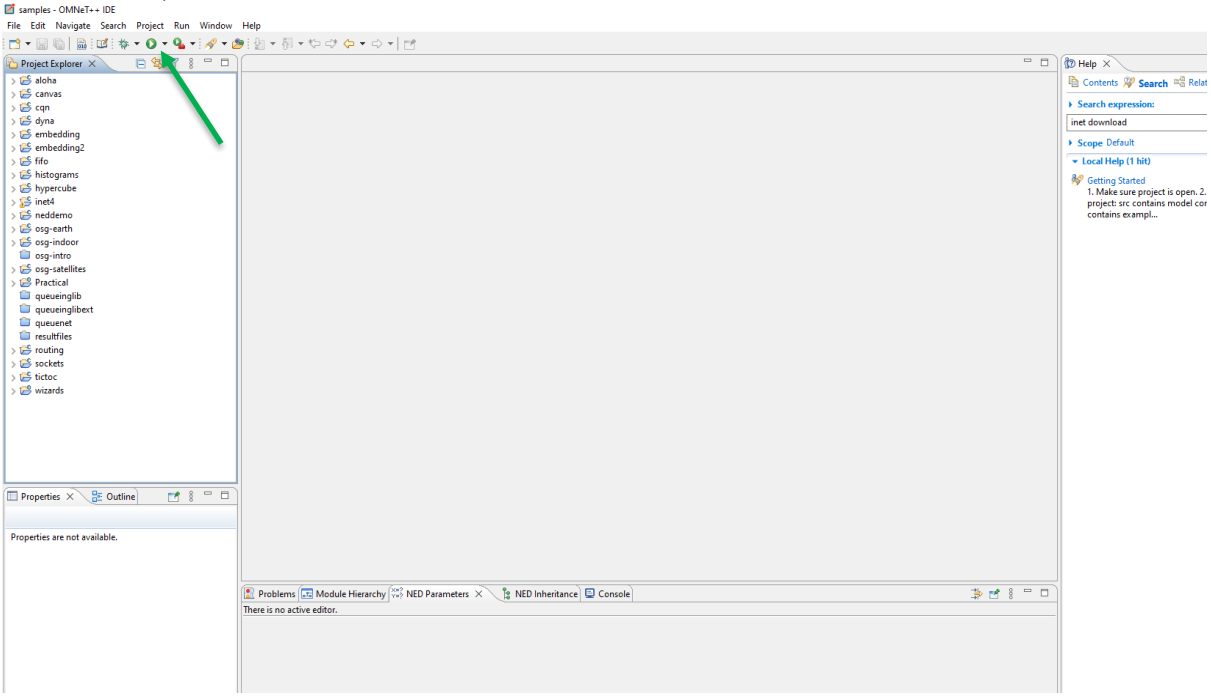
4.If you don't get the speed parameter, then click on new and Set the value for **.mobility.speed = 400mps



5.Now change the numHosts parameter to 5.(shown in blue arrow below)



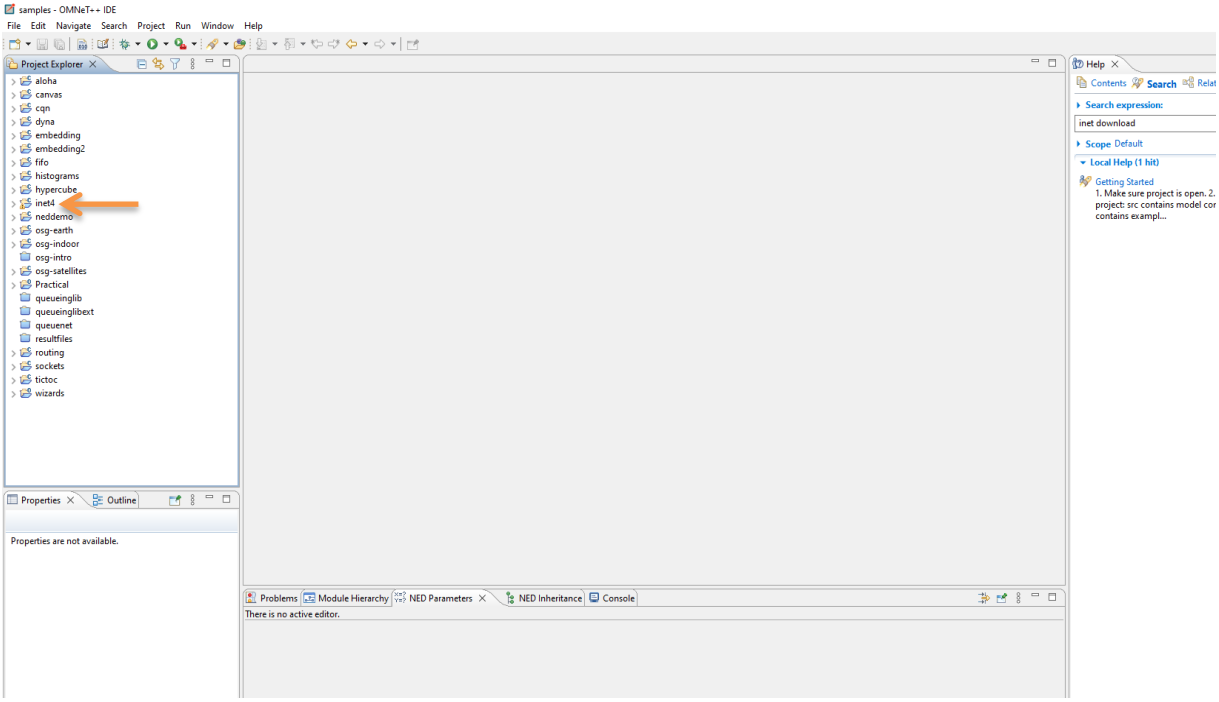
6.Now the run the simulation by going to DYMONetwork.ned (run button is shown in green arrow below)



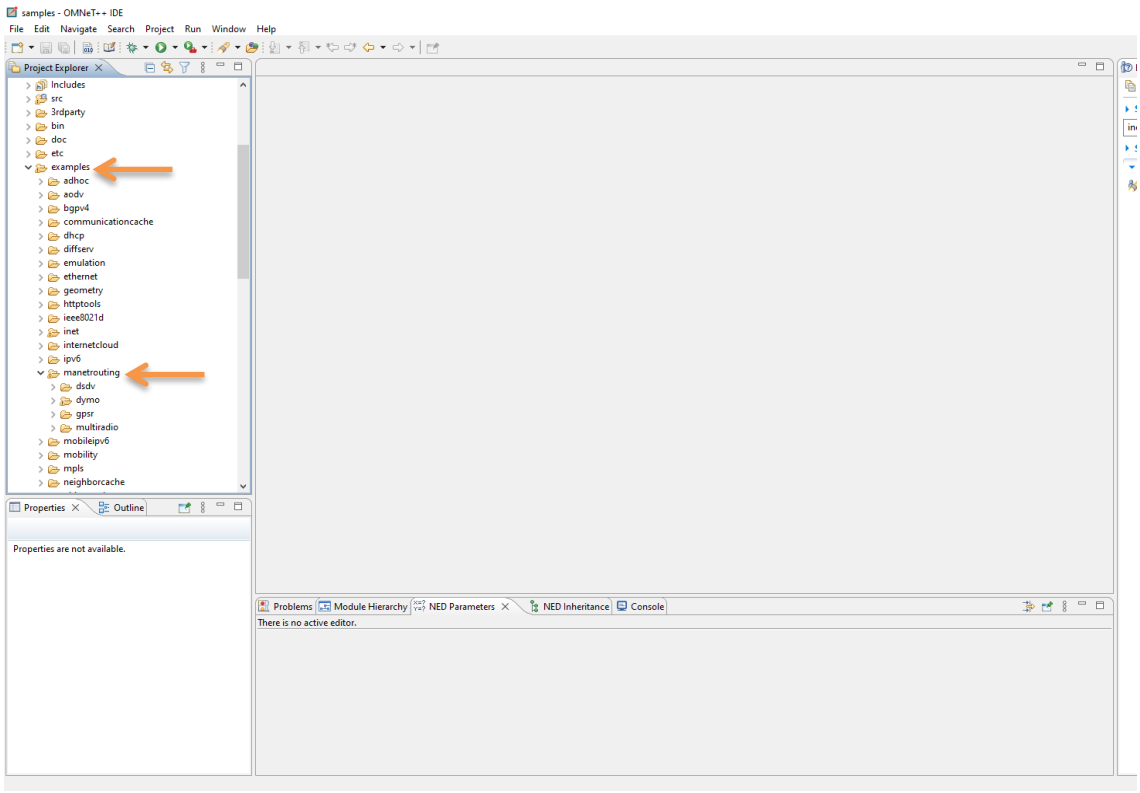
Also pause the simulation so that you can show the external your output.
Below the steps are given in more detail.

We create an MANET using Omnet++ and INET through the following steps

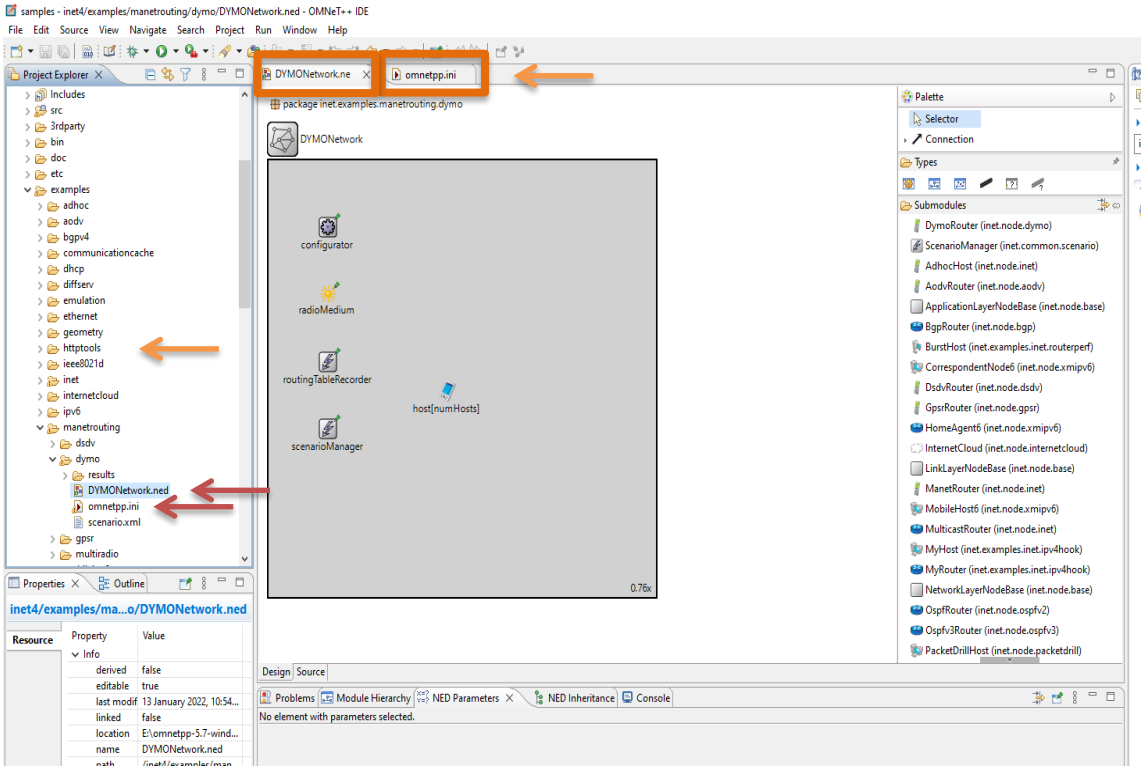
Step 1: Open the Omnet++ software and click on inet4 folder



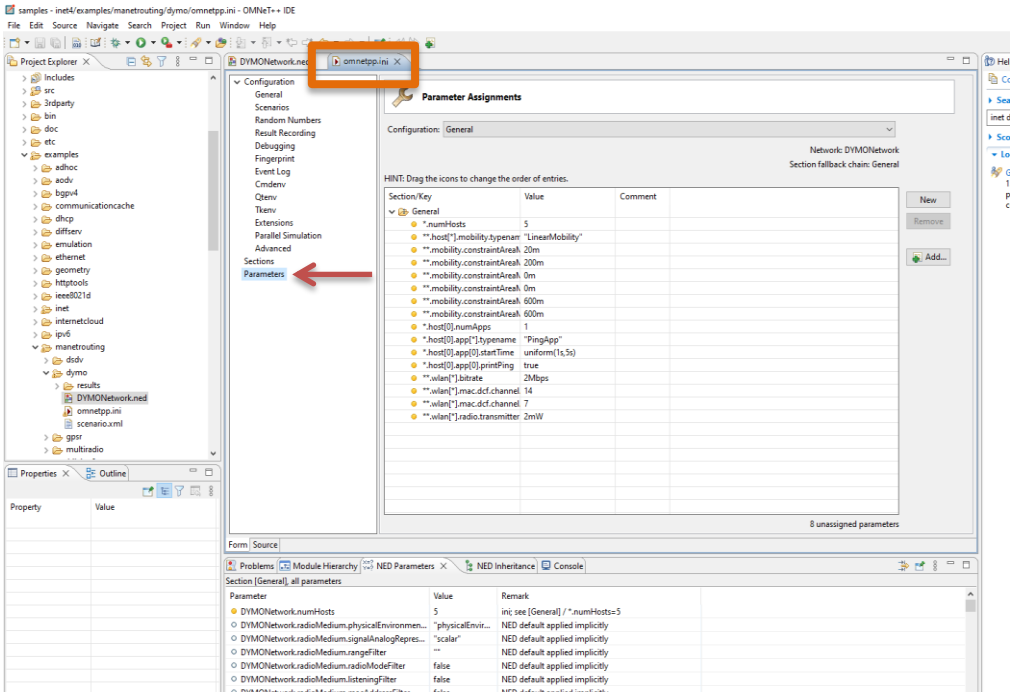
Step 2: Now select the examples folder and then in that folder select manetrouting folder



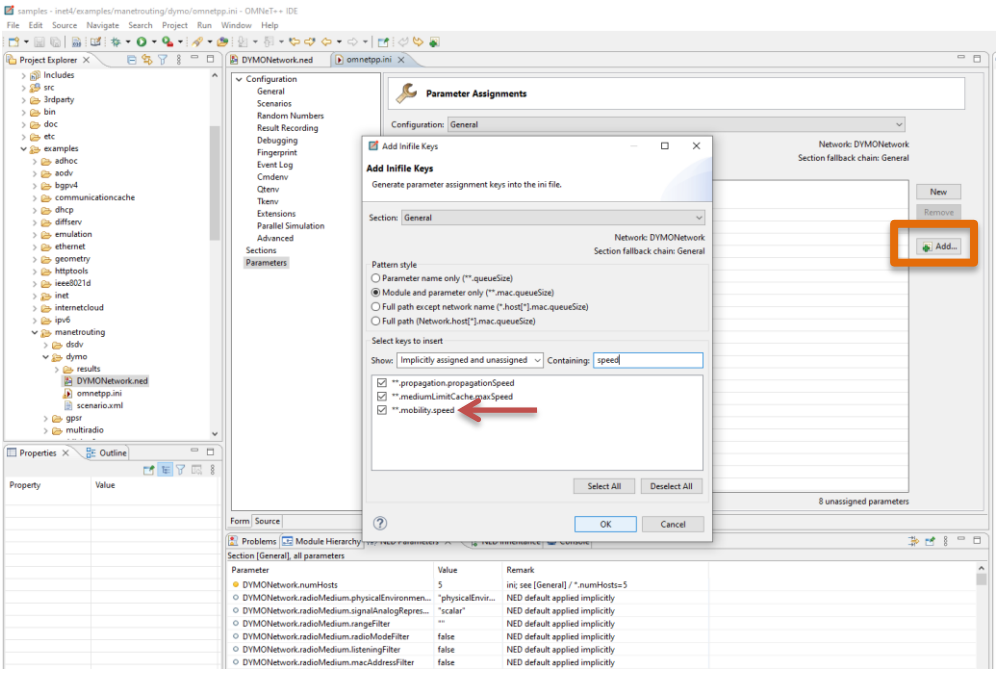
Step 3 : In manetrouting folder click dymo folder and then load the DYMONetwork.ned and omnetpp.ini files by double clicking



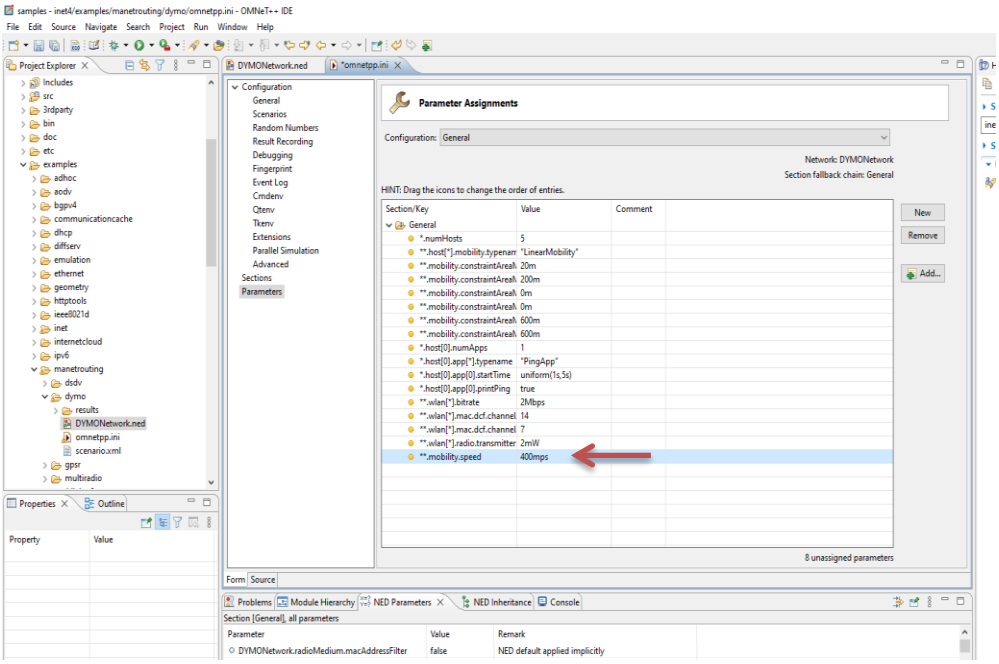
Step 4: Select omnetpp.ini file and click on parameters, we need to add mobility to the nodes



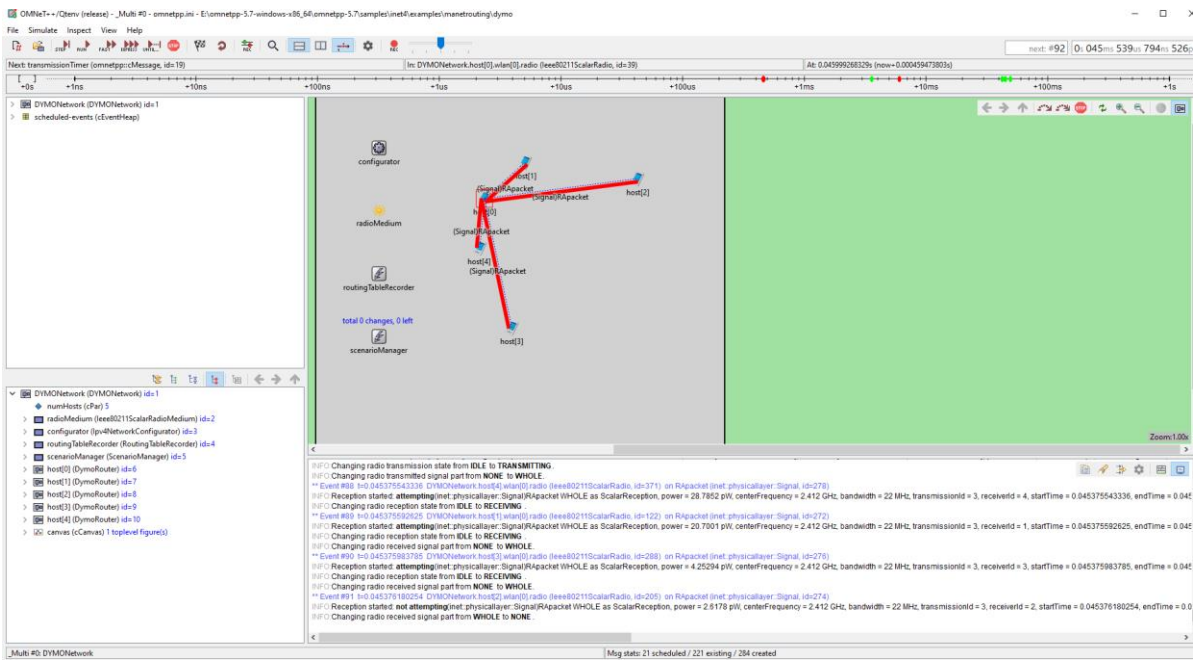
Step 5: For adding a new parameter click on add button and add the parameter **.mobility.speed



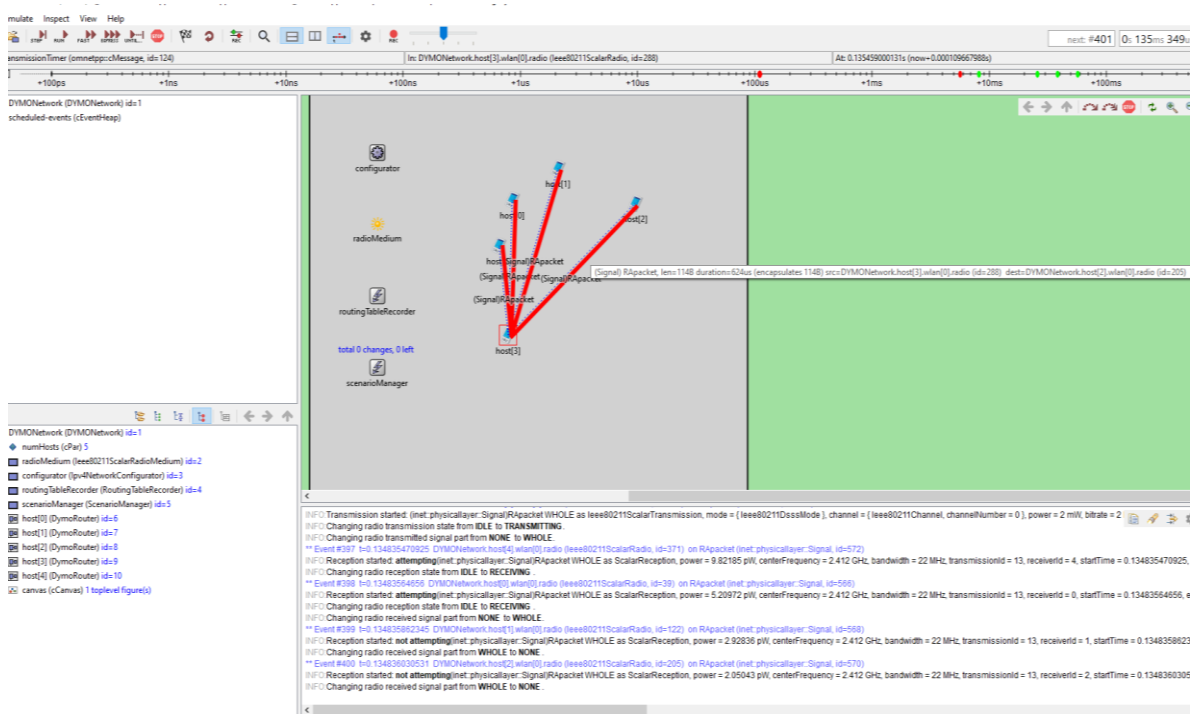
Step 6: Set the value for **.mobility.speed = 400mps



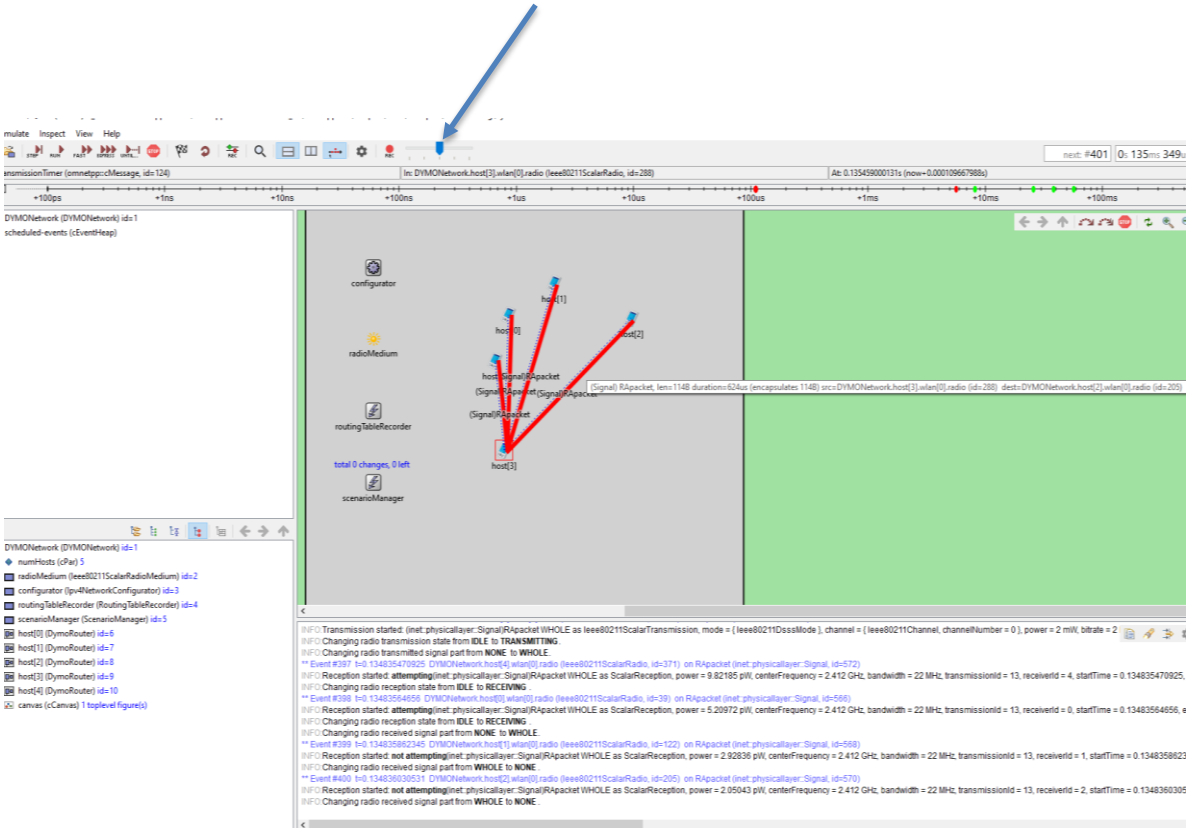
Step 7: Now we run the simulation with 5 mobile hosts forming MANET and get the following output



Since the nodes have mobility, after sometime their positions would change and we get



You can control the speed of simulation using that blue pin. It is shown using blue arrow below.

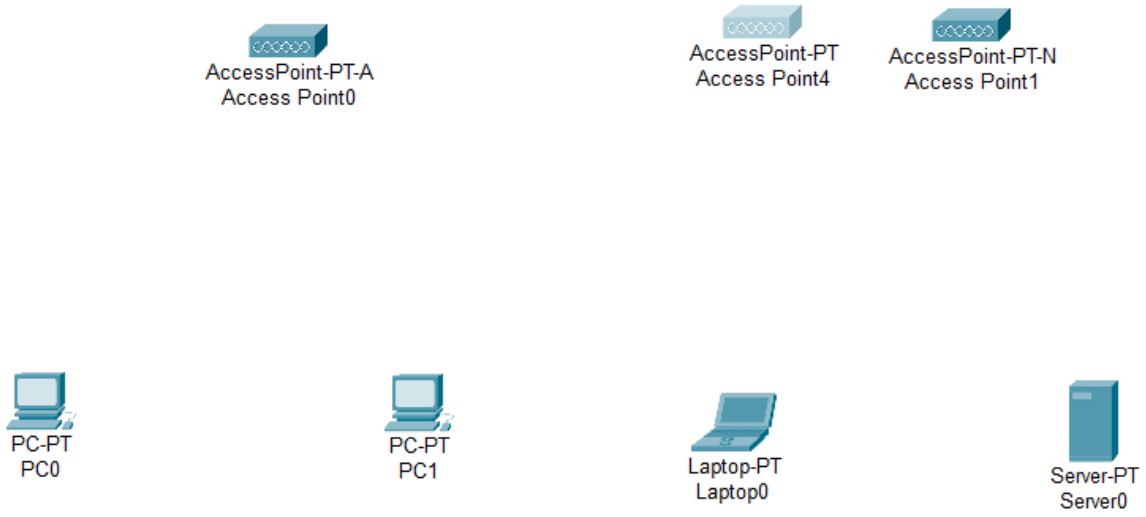


Hence the given MANET has been simulated with 5 hosts

Practical -7

Aim: Implement a Wireless sensor network simulation.

Basic setup

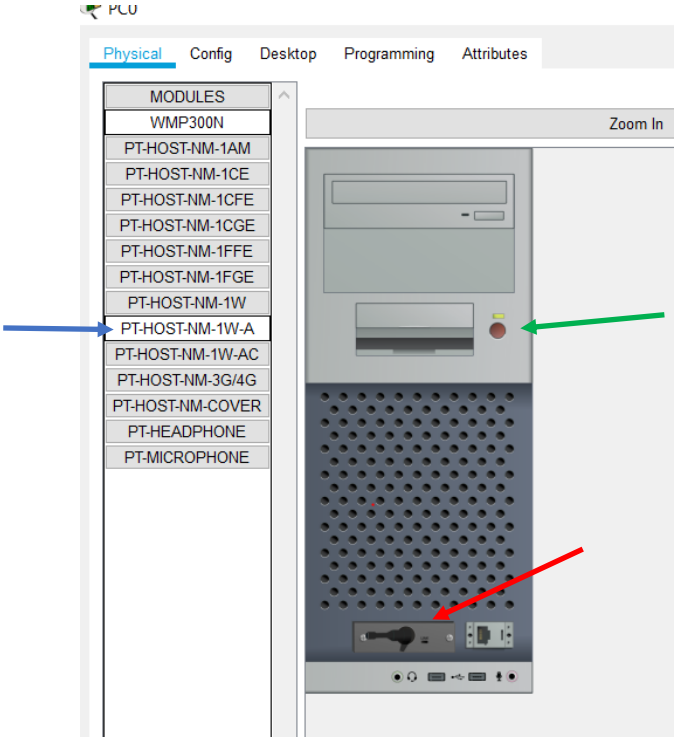


1: Create the following network using AccessPoint-PT-A and PC-PT.

2: Click on PC0 → Physical tab.

3: Turn off the CPU (off - green arrow) and remove the FastEthernet module (removing place in red arrow) and install PT-HOST-NM-1W-A (blue arrow) and turn On the CPU.

Note: when ON yellow light will glow above red on/off button.



4. A connection will be made between Accesspoint and PC0

5. Click on PC1 and click on physical tab.

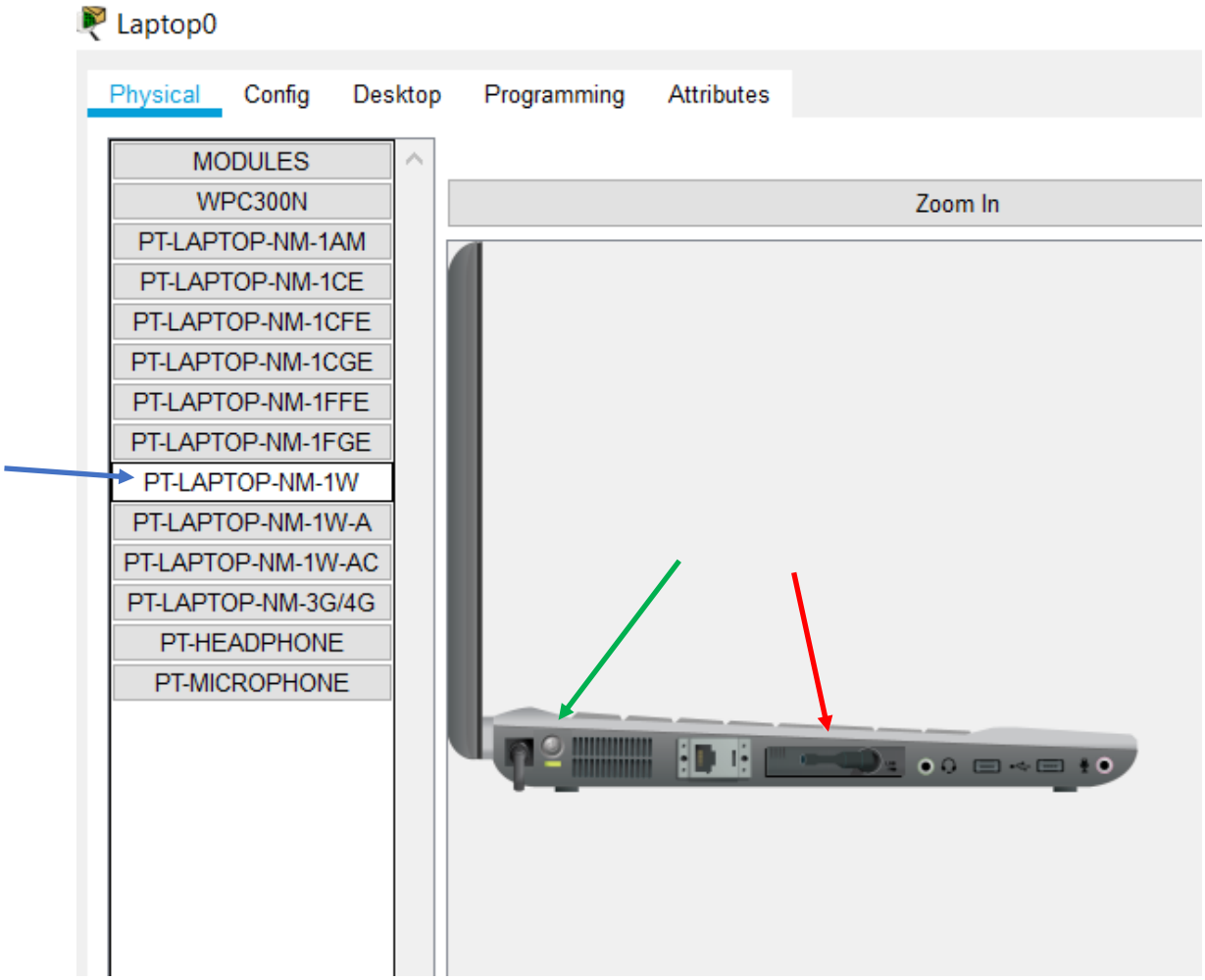
6. Repeat step 3 and see if the connection is done between PC1 and Accesspoint.

Now similarly for laptop

7. Click on Laptop → Physical tab.

8. Turn off the Laptop (off - green arrow) and remove the PT-LAPTOP-NM-1CFE module (removing place in red arrow) and install PT-LAPTOP-NM-1W (blue arrow) and turn On the Laptop.

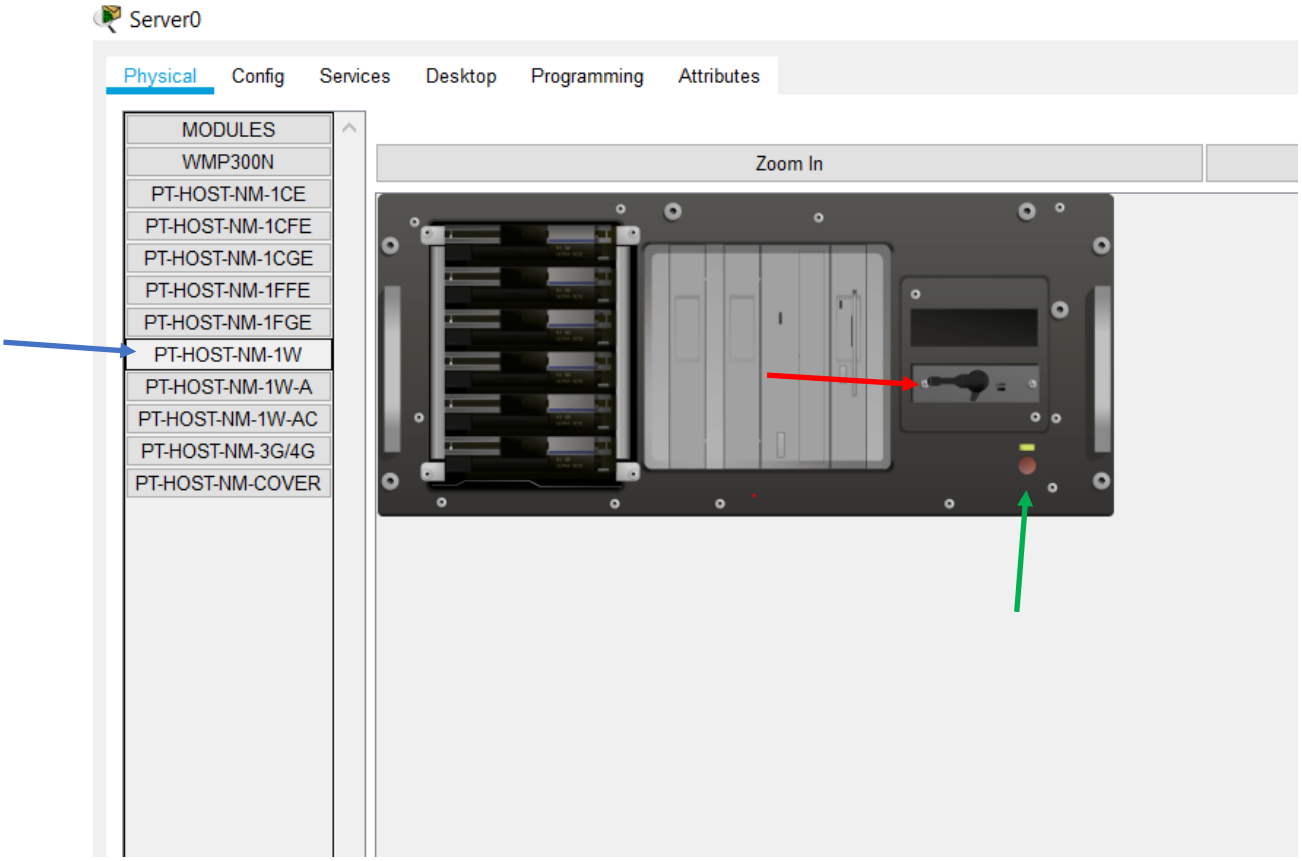
A connection will be made between Accesspoint-PT-N and Server



Now similarly for server

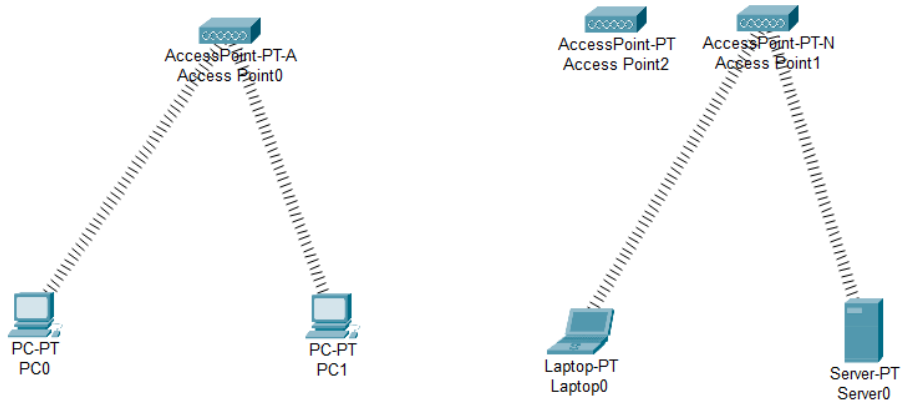
9. Click on Server and click on Physical tab.

10. Turn off the Server (off - green arrow) and remove the PT-HOST-NM-1CFE module (removing place in red arrow) and install PT-HOST-NM-1W (blue arrow) and turn On the Server.



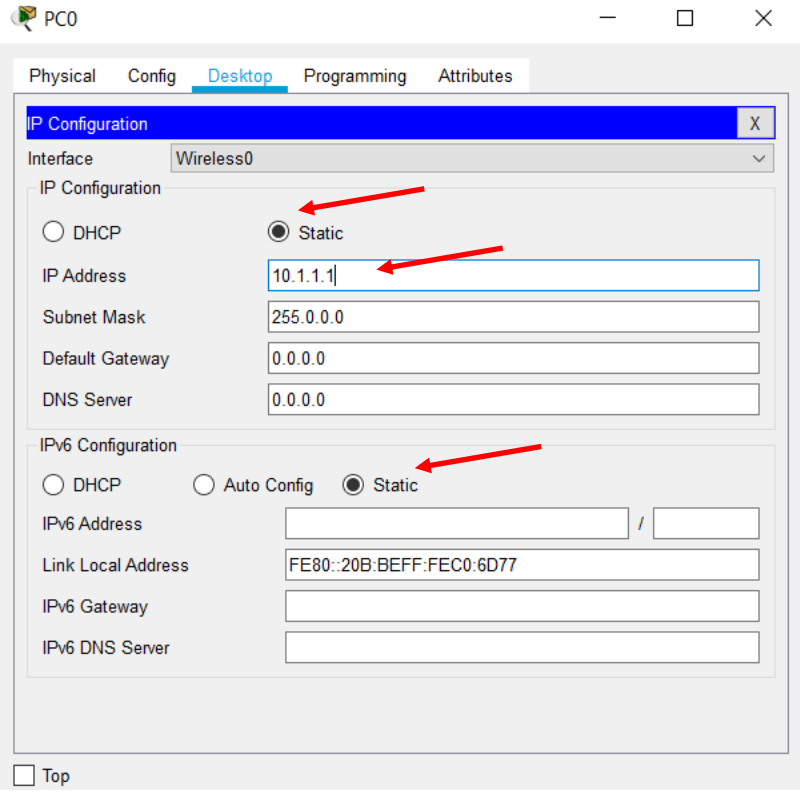
A connection will be made between Accesspoint-PT-N and Server.

After these stes the setup looks like:



11. Click on PC0 → Desktop → IP Configuration & set the IP config:

Select both the static options & give IP Address as 10.1.1.1



Similarly for **PC1**

12. Click on PC1 → Desktop → IP Configuration & set the IP config:

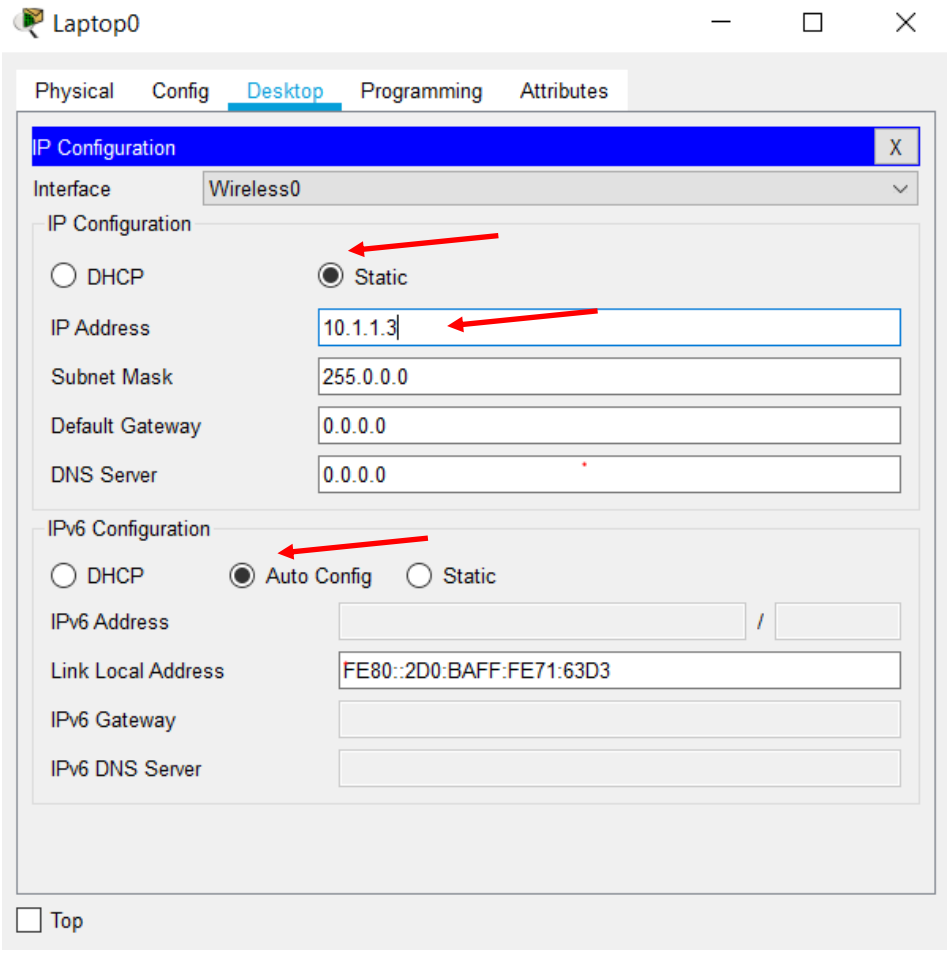
Select both the static options & give IP Address as 10.1.1.2

Similaly for **laptop**

13. Click on Laptop → Desktop → IP Configuration & set the IP config:

Select first option as **static** for IP configuration & give IP Address as 10.1.1.3\

Select second option **Auto Config** for Ipv6 Configuration.



Similarly for **server**

14. Click on server → Desktop → IP Configuration & set the IP config:

Select both the static options & give IP Address as 10.1.1.4

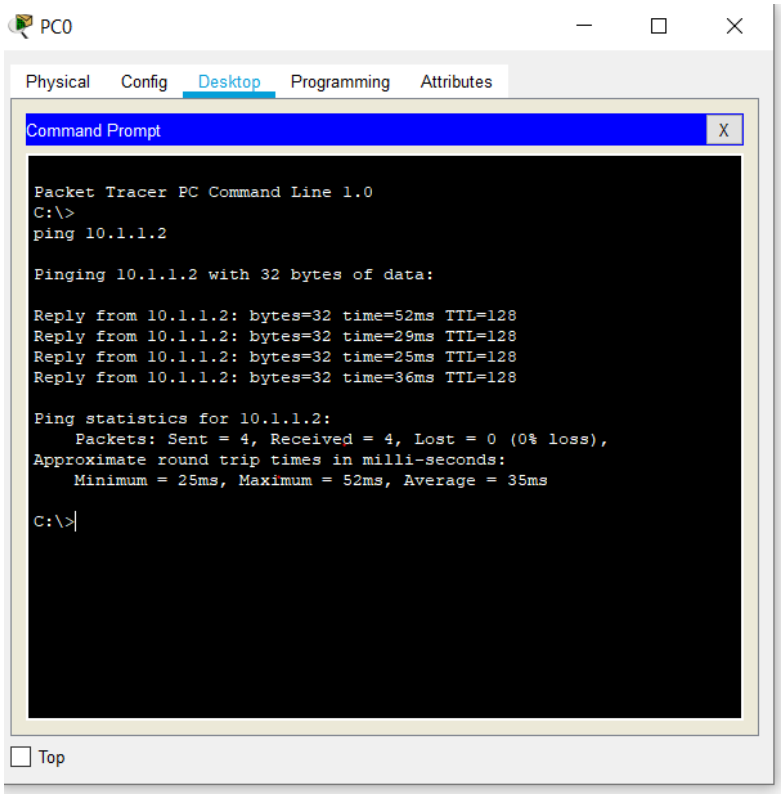
15.Testing connections:

Test Access PointA

a. Ping PC1 (10.1.1.2) from PC0. The ping should succeed.

To Ping click on PC0 → Desktop → Command prompt

Write command: ping 10.1.1.2

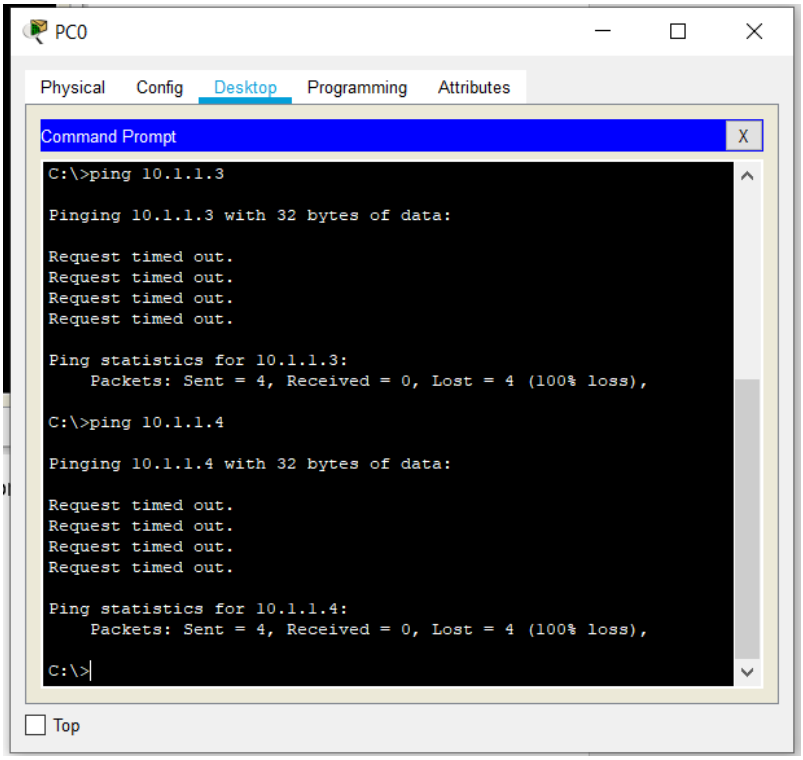


b. Ping Laptop0(10.1.1.3) and Server0 (10.1.1.4) from PC0. The pings should fail.

Similarly on the same terminal write the commands:

ping 10.1.1.3

ping 10.1.1.4



16.Test Access PointN

a.Ping Server0 (10.1.1.4) from Laptop0. The ping should succeed.

To Ping click on Laptop → Desktop → Command prompt

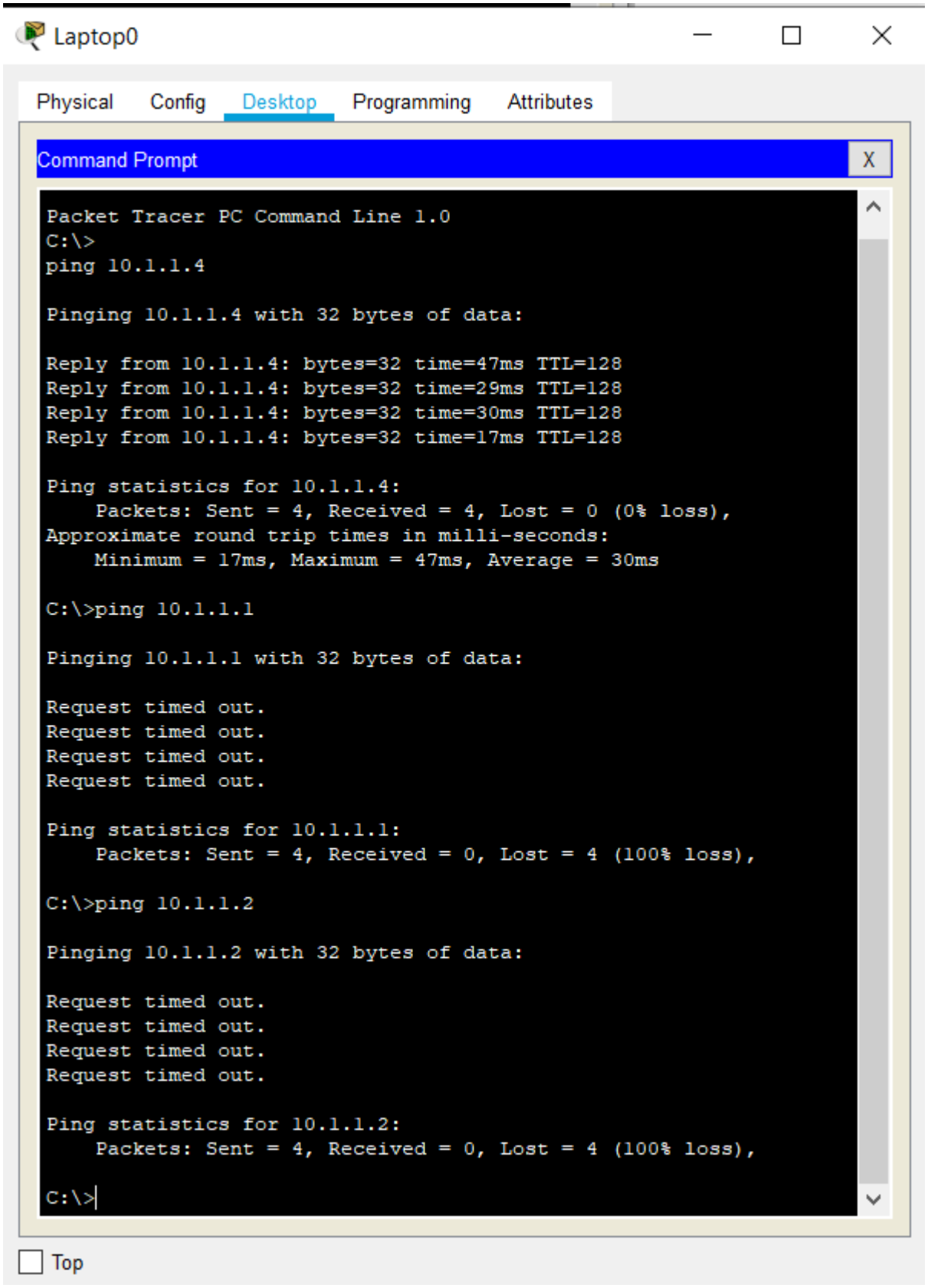
Write command: ping 10.1.1.4

b. Ping PC0 (10.1.1.1) and PC1 (10.1.1.2) from Laptop0. The pings should fail.

Similarly on the same terminal write the commands:

ping 10.1.1.1

ping 10.1.1.2

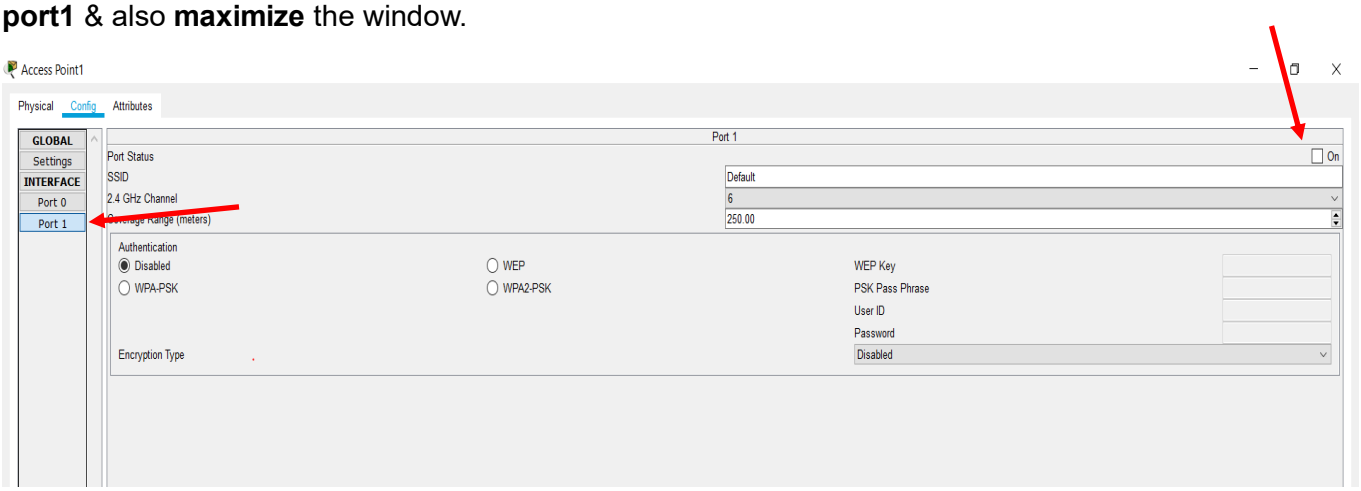

Laptop0
Physical
Config
Desktop
Programming
Attributes
Command Prompt
X
Packet Tracer PC Command Line 1.0
C:\>
ping 10.1.1.4
Pinging 10.1.1.4 with 32 bytes of data:
Reply from 10.1.1.4: bytes=32 time=47ms TTL=128
Reply from 10.1.1.4: bytes=32 time=29ms TTL=128
Reply from 10.1.1.4: bytes=32 time=30ms TTL=128
Reply from 10.1.1.4: bytes=32 time=17ms TTL=128
Ping statistics for 10.1.1.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 17ms, Maximum = 47ms, Average = 30ms
C:\>ping 10.1.1.1
Pinging 10.1.1.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 10.1.1.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ping 10.1.1.2
Pinging 10.1.1.2 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 10.1.1.2:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>
Top

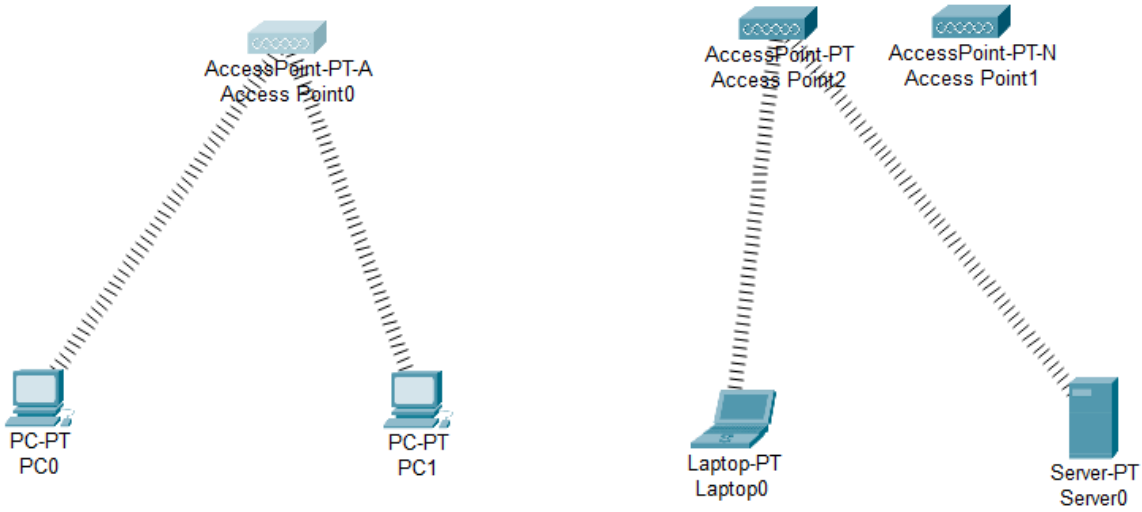
17. Now Turn off the port of AccesspointN

a. Turn on Port1 on Access Point3 and turn off Port1 on Access PointN. Laptop0 and

Server0 should associate with Access Point3.

To **turn off/on** you need to **select the access point** & then **go to config** then **click on port1** & also **maximize** the window.





Lastly ping server (10.1.1.4) from laptop using command:

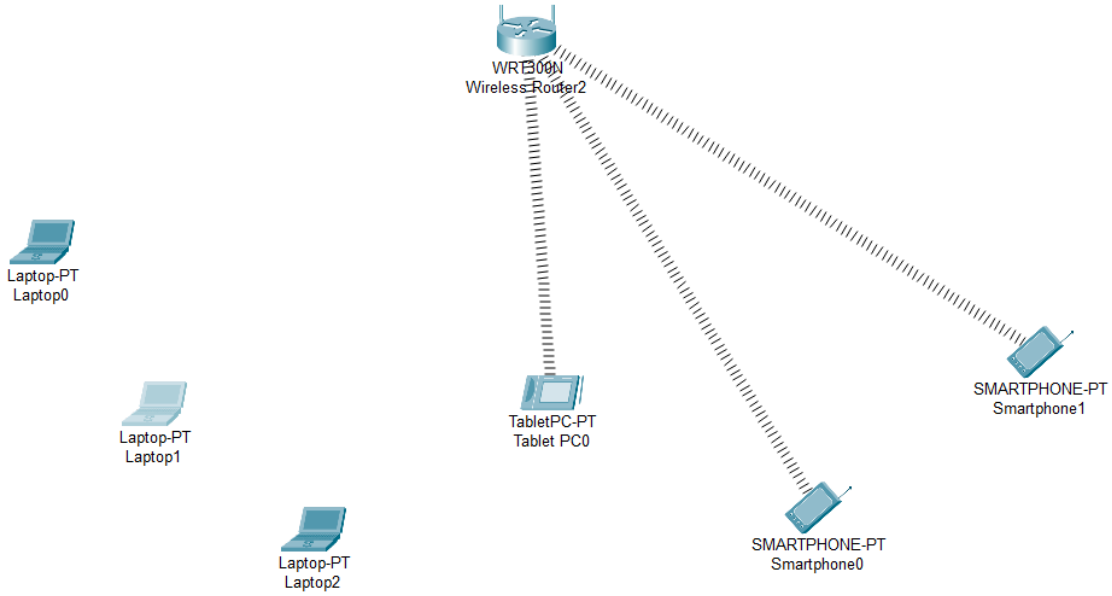
Ping 10.1.1.4

Ping will be successful

Practical 8

Aim: Create MAC protocol simulation implementation for wireless sensor Network

Basic Setup:

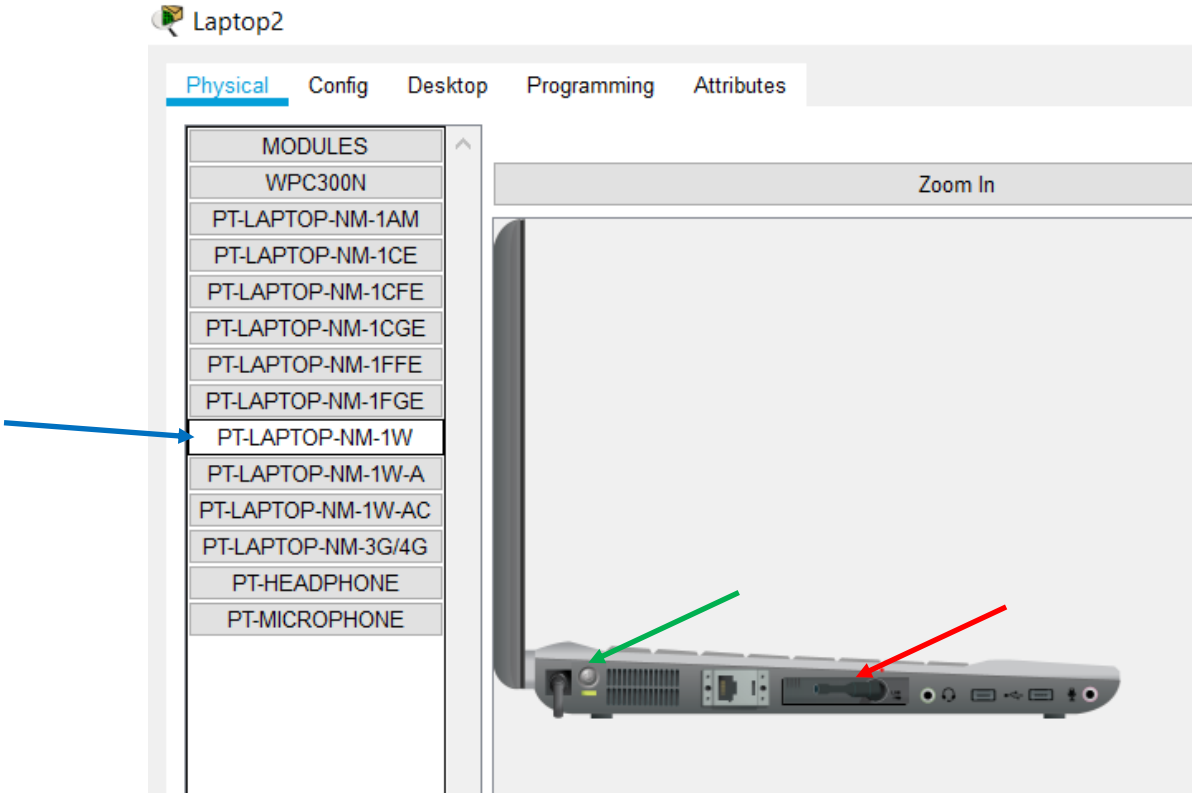


1. Click on every laptop & change the interference interface to PT-LAPTOP-NM-1W

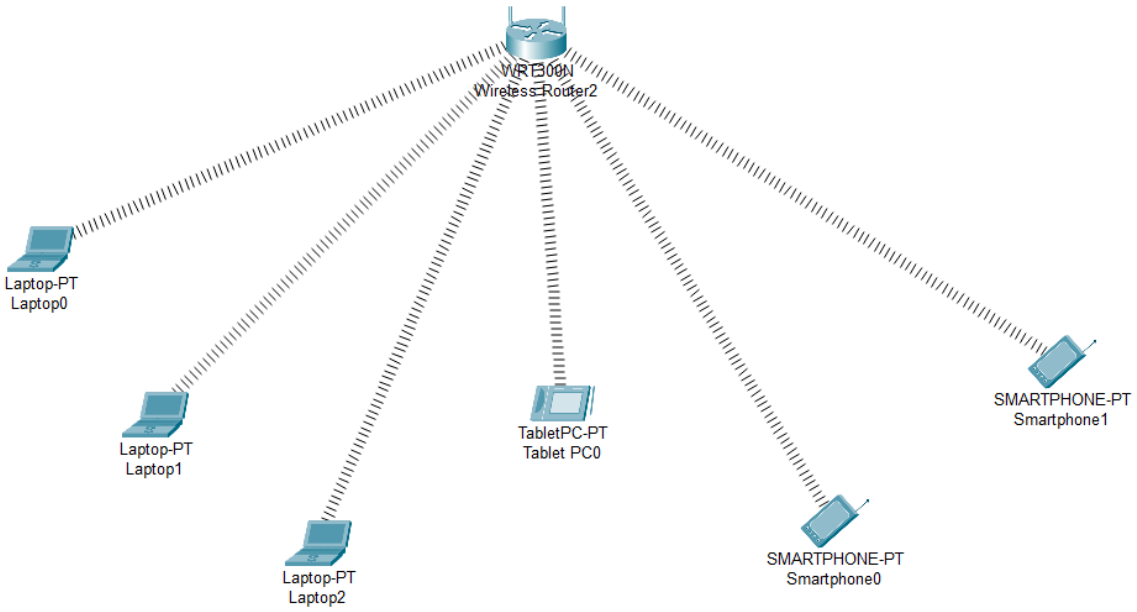
Turn off the PC(green arrow)

Remove the existing interface(red arrow)

Put the new interface PT-LAPTOP-NM-1W(blue arrow) & turn on the Laptop



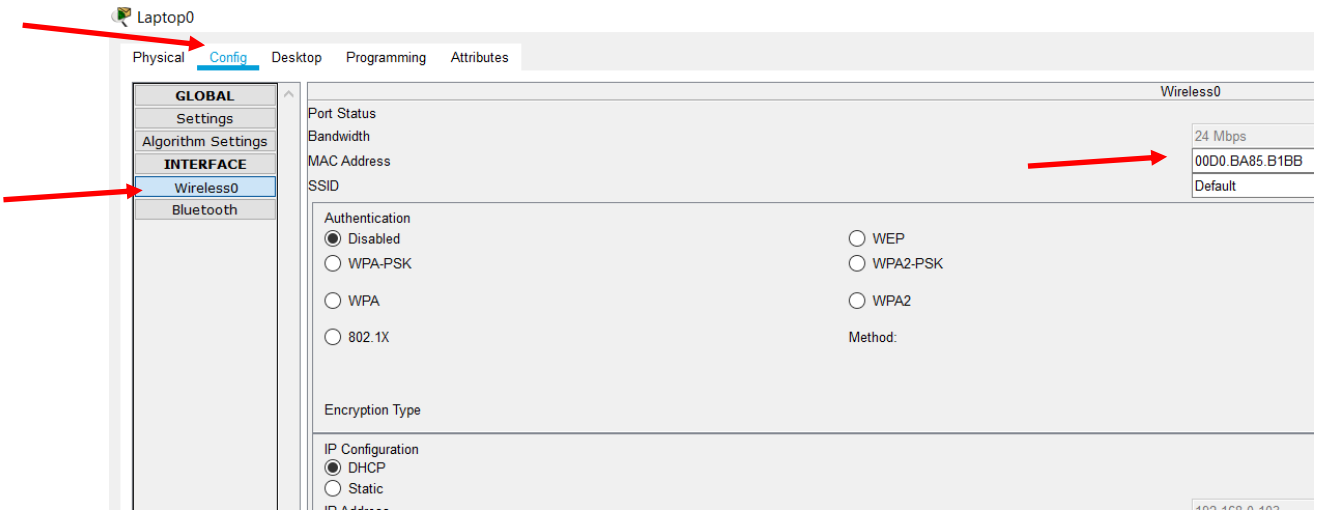
After step 1 the setup looks like:



2. Copy the MAC address of each component as follows

Click on component → Config → Wireless0 then,

Copy the MAC Address & paste it somewhere (eg.Notepad)

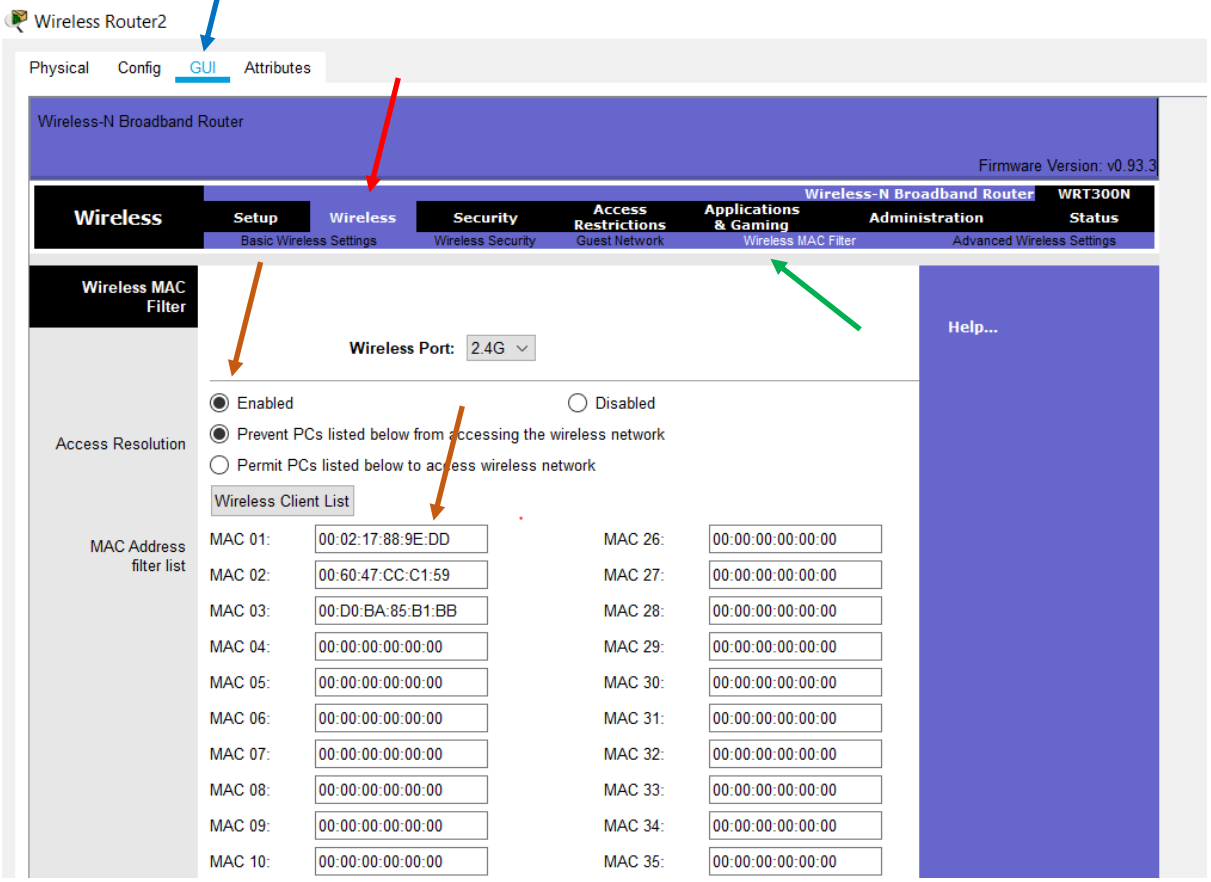


3.Now we add few addresses in the wireless MAC filter of the Wireless Router and then use the given options for either allow or deny the Wireless access

To do so:

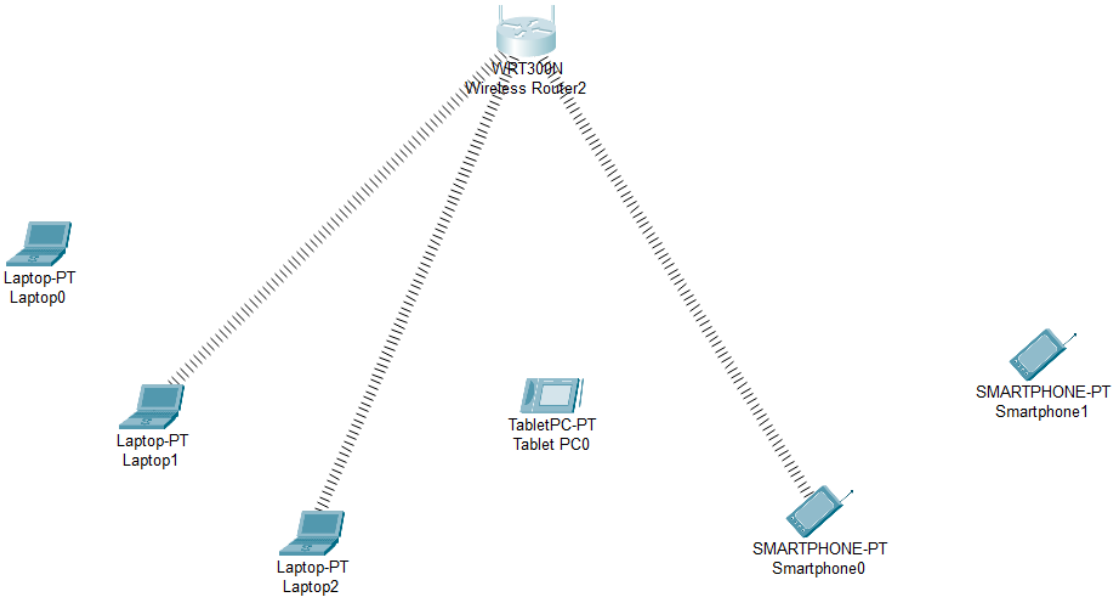
Click on Router → GUI → Wireless → Wireless MAC filter

Then click on enabled (brown arrow) & give the MAC addresses as 00:D0:BA:85:B1:BB(put " : " colon after each 2 digits in MAC address) & hit enter button



Then scroll down & click on Save settings

Output: depends on how many MAC addresses you put



Practical 9

Aim: Simulate Mobile Adhoc Network with Directional Antenna

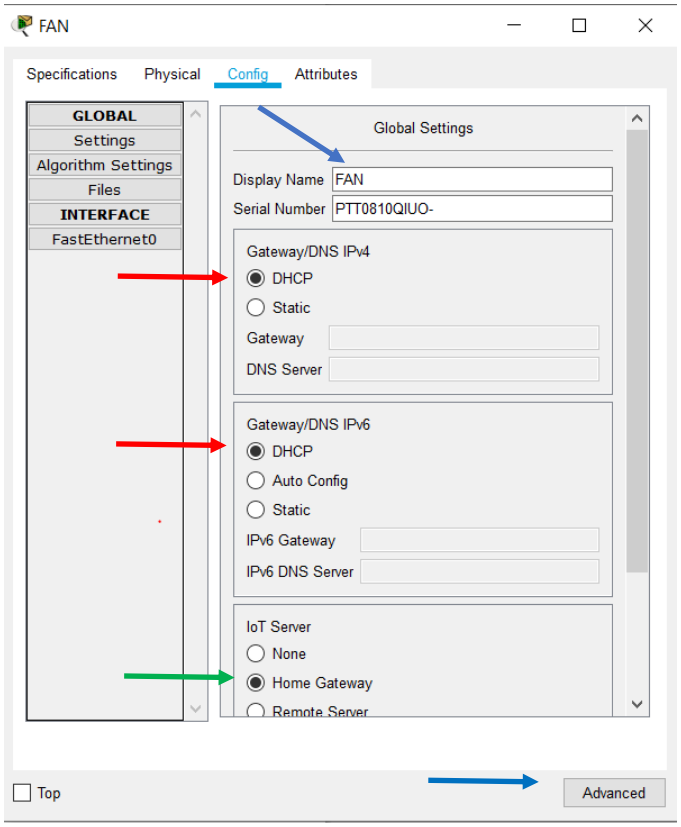
Basic Setup:

1. Click on Fan → Config then,

Change Display name (blue arrow)

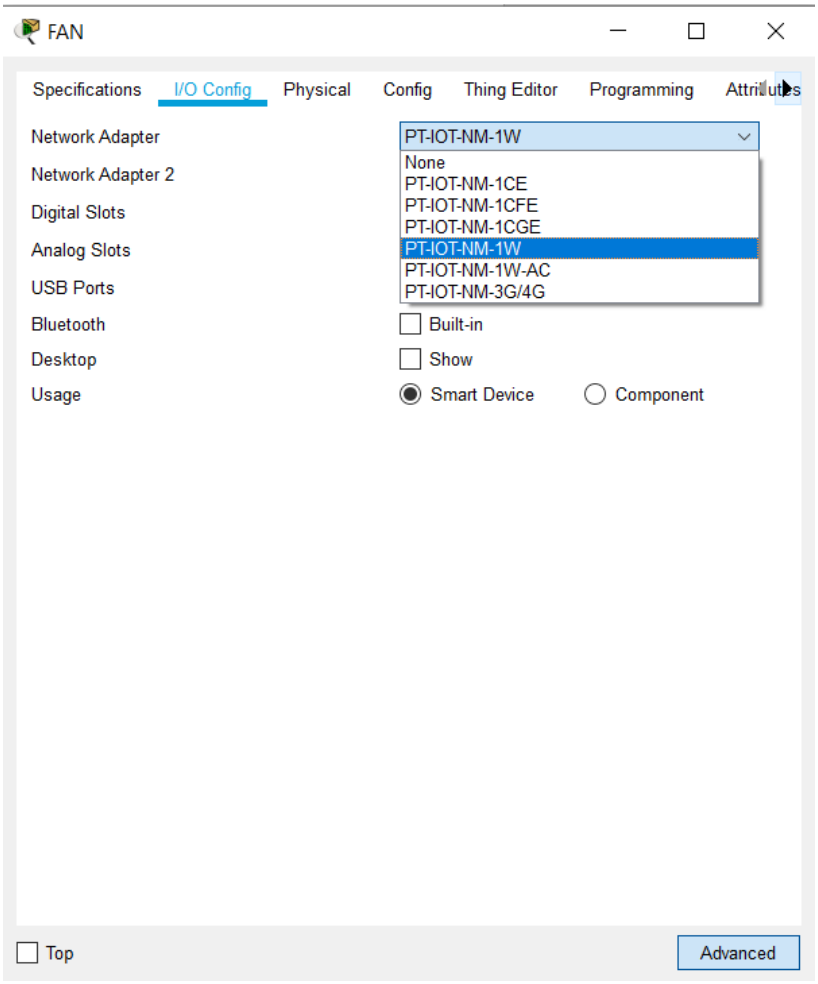
Select DHCP for IPv4 & IPv6 (red arrow)

Select Home Gateway (green arrow)



2. Then click on Advanced → I/O Config

Select the PT-IOT-NM-1W option from the dropdown



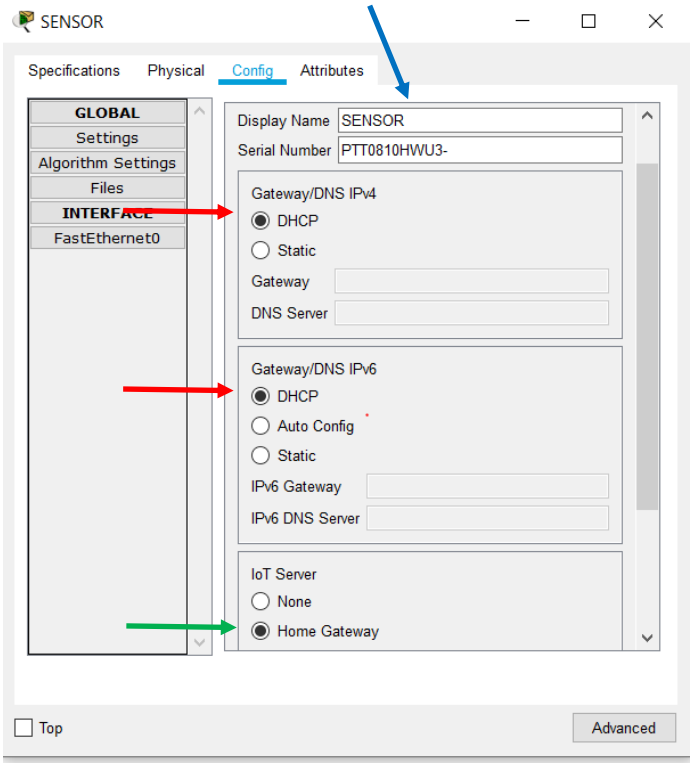
Similarly for motion sensor:

3. Click on Sensor → Config then,

Change Display name (blue arrow)

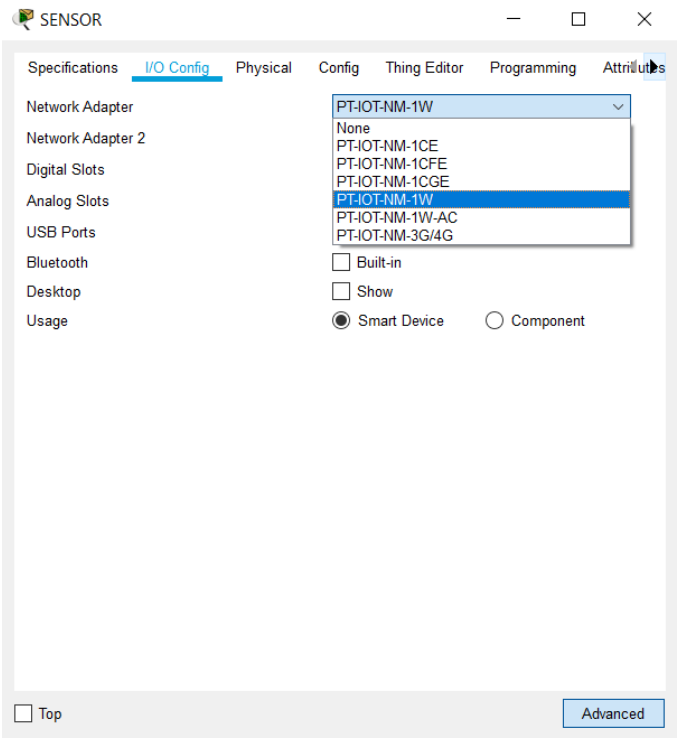
Select DHCP for IPv4 & IPv6 (red arrow)

Select Home Gateway (green arrow)



4.Then click on Advanced → I/O Config

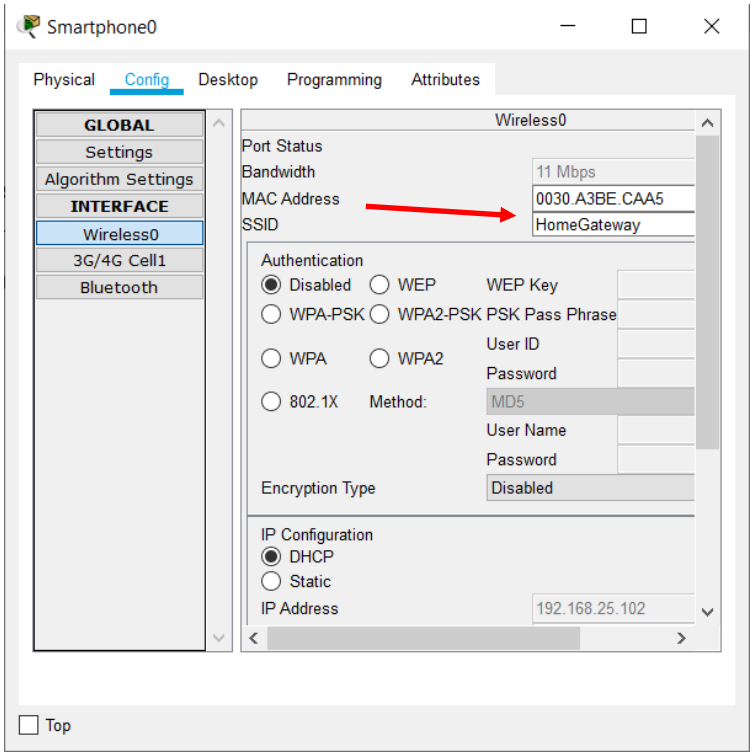
Select the PT-IOT-NM-1W option from the dropdown



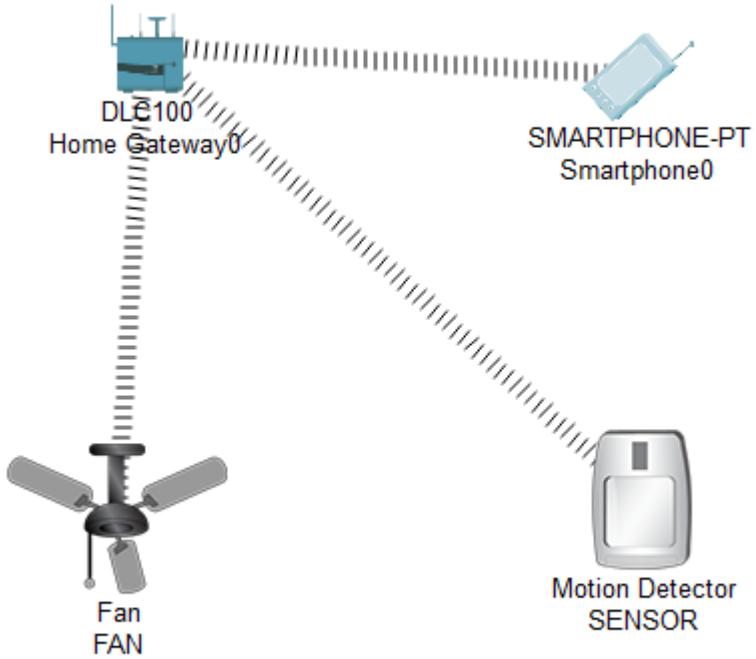
5. For the smartphone change the SSID to the SSID in the Home Gateway0

Click on smartphone → Config → Wireless0

Write HomeGateway instead of Default in SSID



All the devices are now connected to the Home Gateway



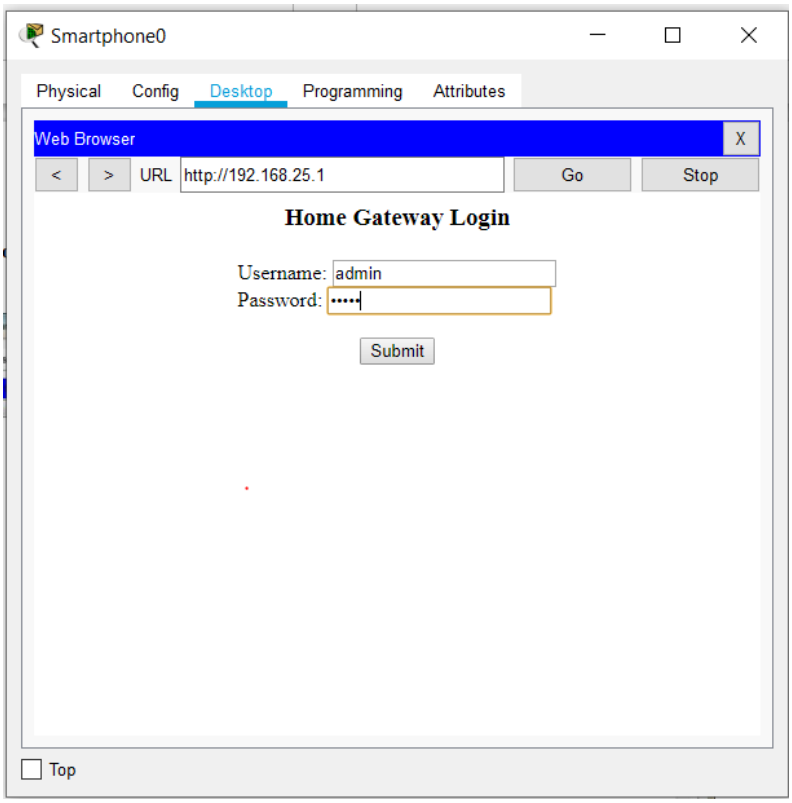
6. Now open the Web browser of the SmartPhone and type the IP address of the HomeGateway

Click on Smartphone → Desktop → Web browser

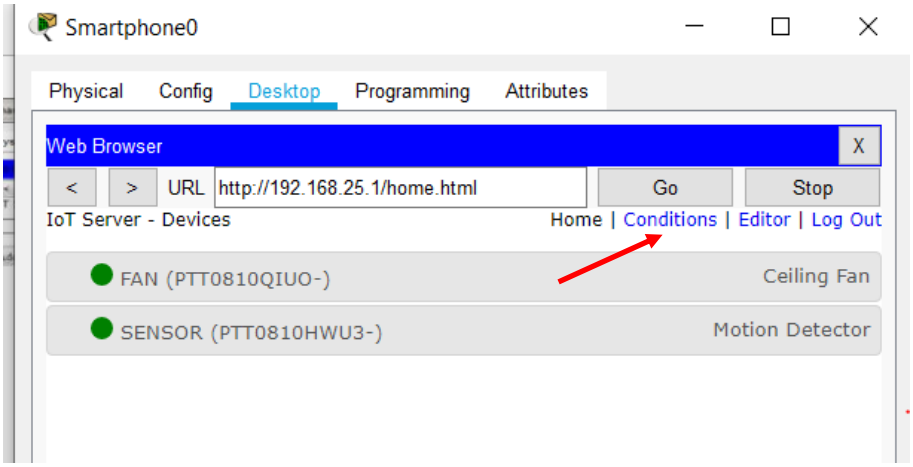
Enter the Url as **192.168.25.1** then click on Go

Then enter **username : admin** & **password : admin**

Then click submit

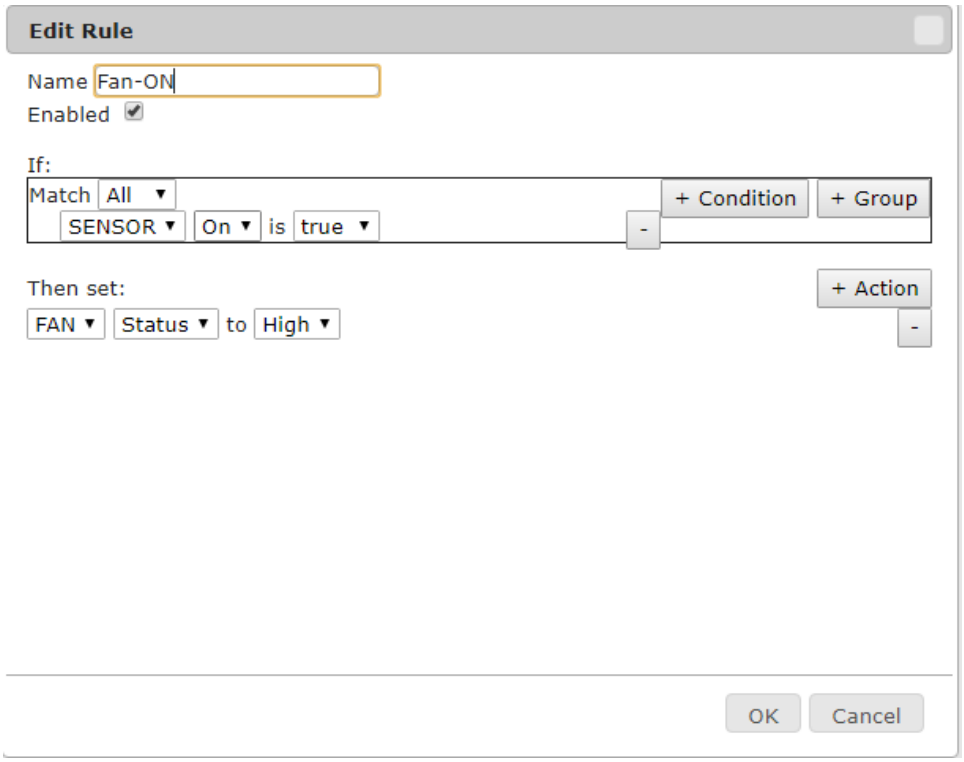


7.After logging click on conditions

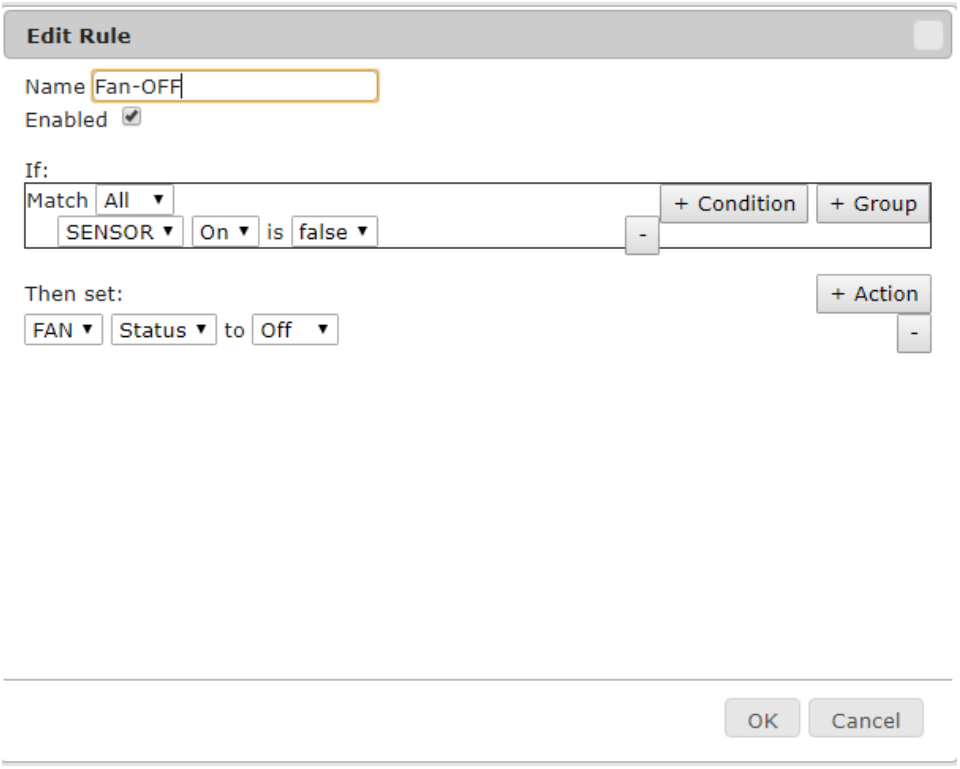


Then click on Add & add conditions :

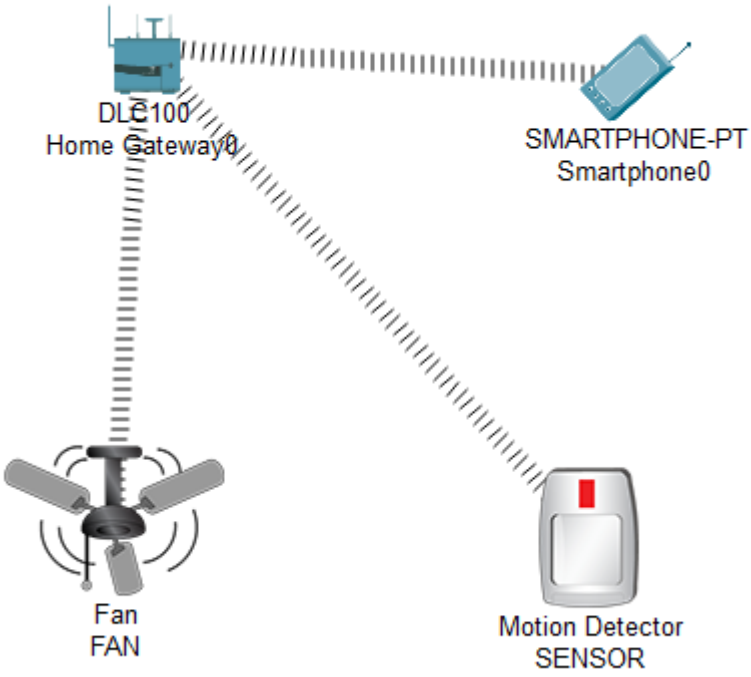
First condition



Second condition



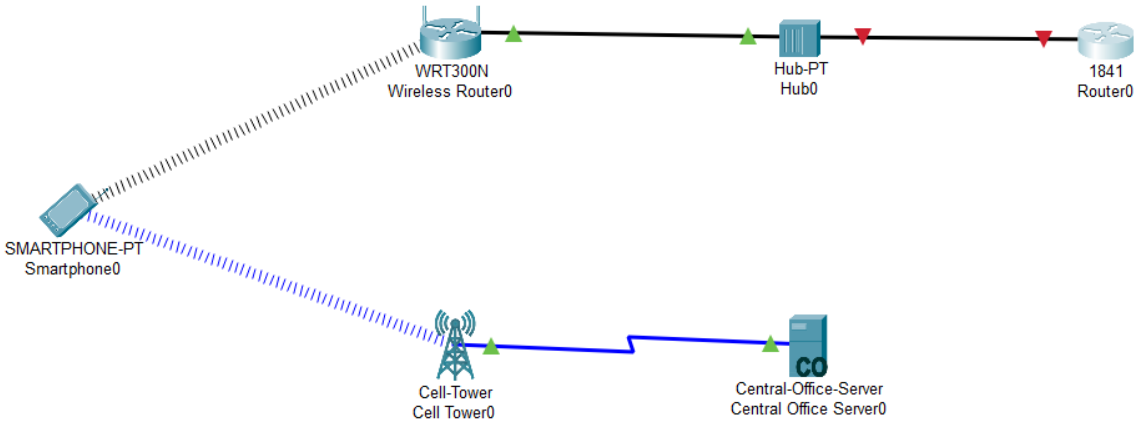
Now bring the arrow on the sensor , then press alt & left-click



Practical 10

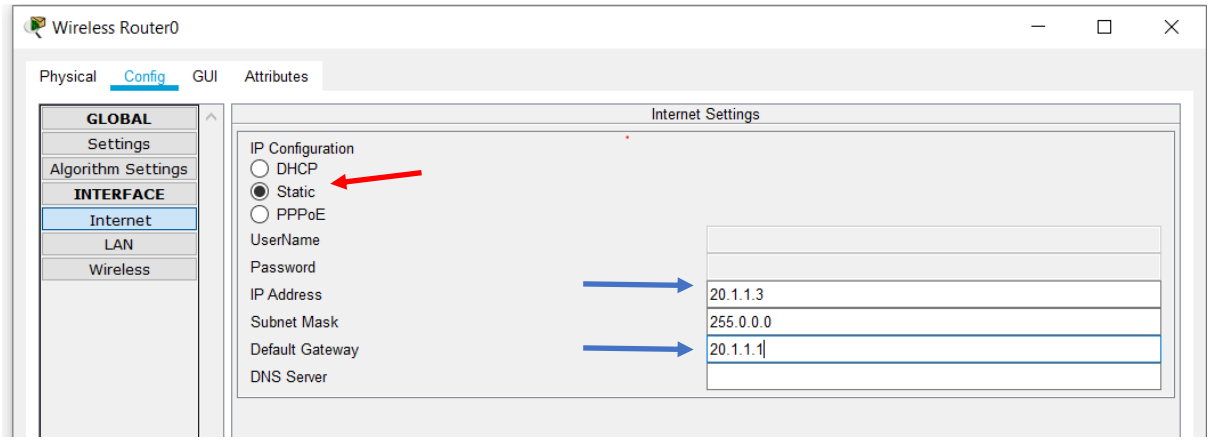
Aim: Create a mobile network using Cell Tower, Central Office Server, Web browser and Web Server. Simulate connection between them.

Basic Setup:



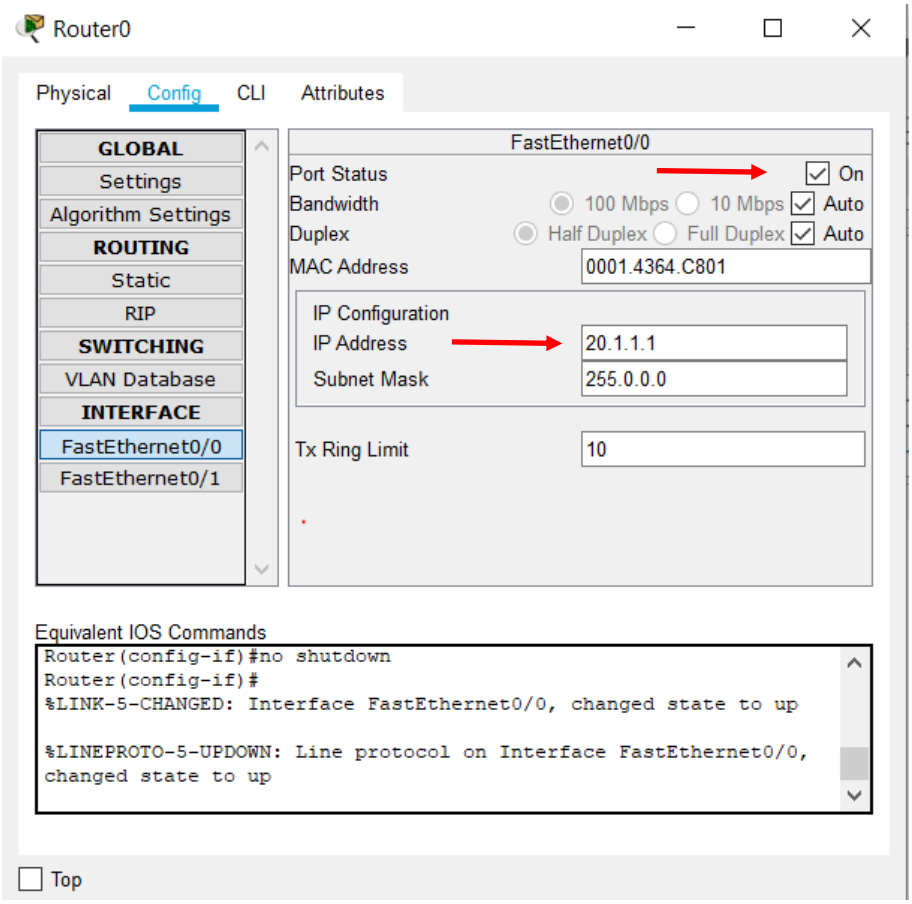
1. Click on Wireless Router(WRT300N) → Config → Internet

Then choose static option(red arrow) & give IP address as 20.1.1.3 & hit enter, then give default gateway as 20.1.1.1



2.Click on router 1841 → Config → FastEthernet0/0

There give IP Address as 20.1.1.1 & click the ☑ On checkbox

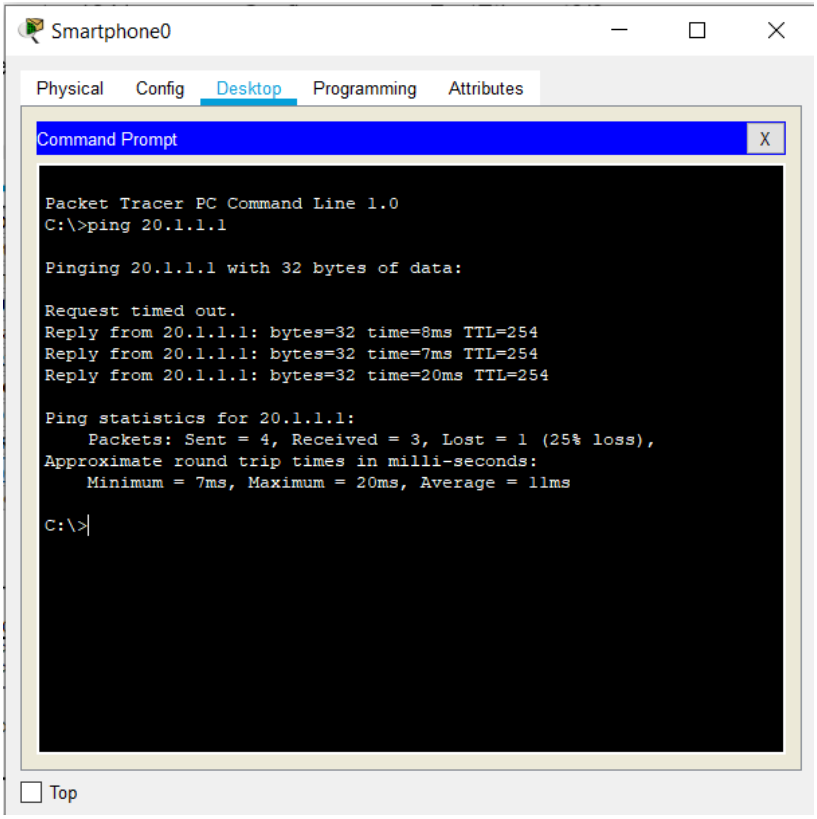


3. Click on Smartphone → Desktop → Command Prompt

In cmd ping router 1841 using command:

Ping 20.1.1.1

On first ping there will be 25% loss



On second ping there will be no loss.