Wireless Sensor Networks Practical

Practical - 1

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

Wireless sensor networks (WSNs) have the power of distributed communication, computing, and sensing features. They are characterized as infrastructure less, fault tolerant and self-organizing networks which provide opportunities for low-cost, easy- to-apply, rapid and flexible installations in an environment for various applications.

The components of sensor node hardware are as follows:

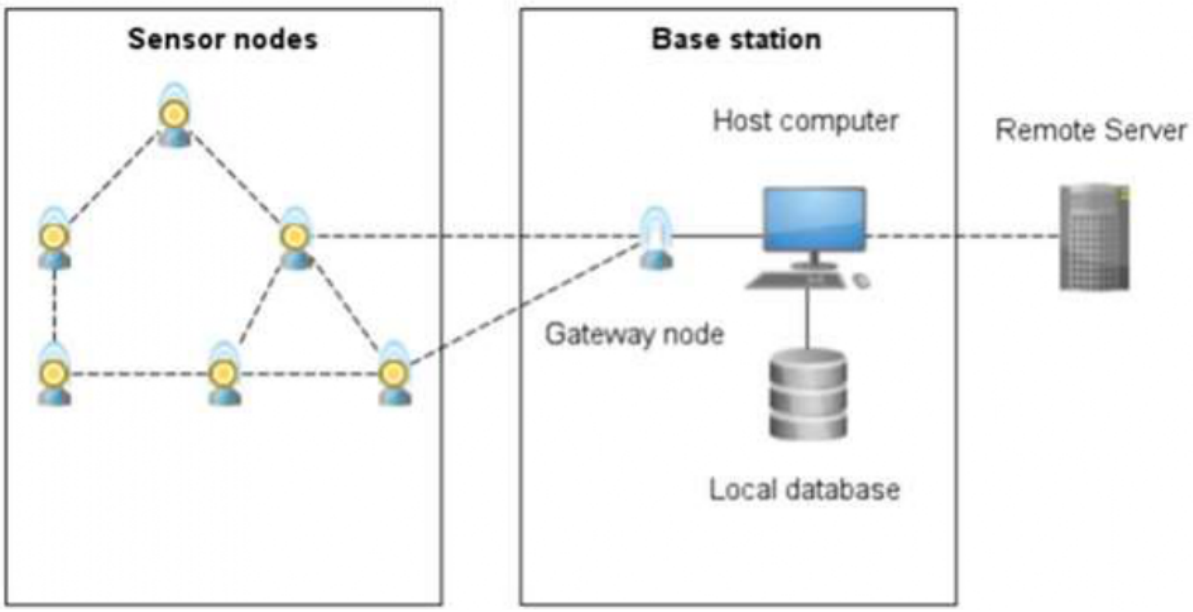
1. Sensors: Sensors in WSN are used to capture the environmental variables and which is used for data acquisition. Sensor signals are converted into electrical signals. They are capable of executing data processing, data gathering and communicating with additional associated nodes in the network. A distinctive sensor node capability is about 4-8 MHz, having 4 KB of RAM, 128 KB flash and preferably 916 MHz of radio frequency.

2. Nodes: They are used to enhance the network reliability. A relay node is a special type of field device that does not have process sensor or control equipment and as such does not interface with the 7 process itself. A distinctive rely node processor speed is about 8 MHz, having 8 KB of RAM, 128 KB flash and preferably 916 MHz of radio frequency.

3. Base Station: Besides sensor nodes, another fundamental item - a base station can be found in WSN. In contrast to sensor nodes, the base station possesses much more computational power, larger memory and is often connected to better energy source than batteries (like power grid). One can look at the base station as an entry point to the WSN where the base station's primary goal is to gather sensed data from sensor nodes in WSN.

4. Graphical User Interface: GUI is the web-based software package, that allows the data collected by sensors to be viewed. The software is also used to set irrigation parameters.

Example of a Wireless Sensor Network (WSN)


Sensor nodes
Base station
Host computer
Remote Server
Gateway node
Local database

Practical - 2

Aim: Exploring and understanding TinyOS computational concepts: - Events, Commands and Task. - nesC model - nesC Components

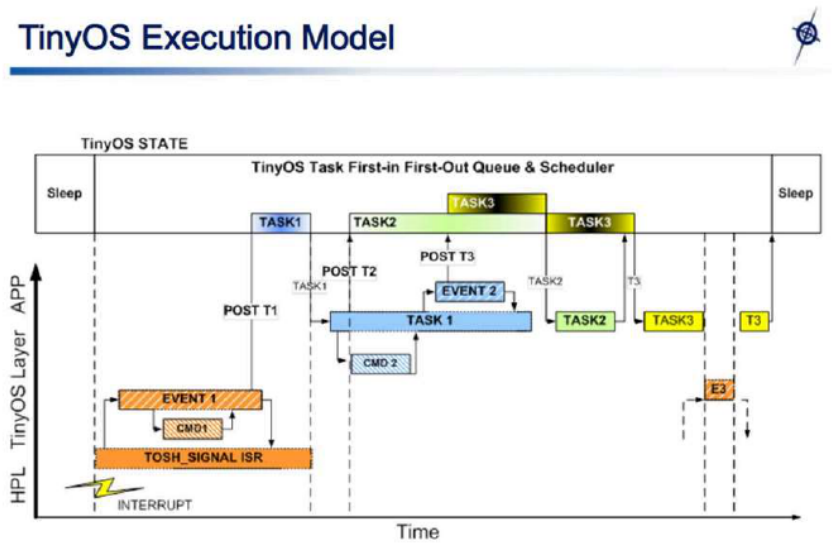
TinyOS has a component-based programming model, codified by the NesC language, a dialect of C. TinyOS is not an OS in the traditional sense; it is a programming framework for embedded systems and set of components that enable building an application-specific OS into each application. A typical application is about 15K in size, of which the base OS is about 400 bytes; the largest application, a database-like query system, is about 64 K bytes.

The computational concepts of TinyOS include: Events, Commands and Task.

1. Events: When a command is requested, event signals the completion of that service. Events may also be signaled asynchronously, for example, due to hardware interrupts or message arrival. From a traditional OS perspective, events are analogous to upcalls.

2. Commands: A command is typically a request to a component to perform some service, such as initiating a sensor reading. Commands are analogous to downcalls. The command returns immediately and the event signals completion at a later time.

3. Task: Rather than performing a computation immediately, commands and event handlers may post a task, a function executed by the TinyOS scheduler at a later time. This allows commands and events to be responsive, returning immediately while deferring extensive computation to tasks. While tasks may perform significant computation, their basic execution model is run-to- completion, rather than to run indefinitely; this allows tasks to be much lighter-weight than threads.



nesC Model

1. nesC is a component-based, event-driven programming language used to build applications for the TinyOS platform. The name nesC is an abbreviation of "network embedded systems C".

2. TinyOS is an operating environment designed to run on embedded devices used in distributed wireless sensor networks.

3. nesC is built as an extension to the C programming language with components "wired" together to run applications on TinyOS.

4. nesC is based on the concept of components, and directly supports TinyOS's eventbased concurrency model.

5. nesC programs are subject to whole program analysis (for safety) and optimization (for performance).

nesC Components

1. nesC applications are built by writing and assembling components.

2. A component provides and uses interfaces. These interfaces are the only point of access to the component.

3. An interface generally models some service (e.g., sending a message) and is specified by an interface type.

4. Interfaces in nesC are bidirectional: they contain commands and events, both of which are essentially functions. The providers or an interface implement the commands, while the users implement the events.

5. The separation of interface type definitions from their use in components promotes the definition of standard interfaces, making components more reusable and flexible.

6. A component can provide and use the same interface type (e.g., when interposing a component between a client and service), or provide the same interface multiple times. In these cases, the component must give each interface instance a separate name.

7. Components are also a clean way to abstract the boundary between hardware and software.

8. A subtle but important point is that bidirectional interfaces make it very easy to support hardware interrupts.

9. In contrast, one-way interfaces based on procedure calls force hardware polling or having two separate interfaces for hardware operations and the corresponding interrupts.

10. There are two types of components in nesC: modules and configurations. Modules provide application code, implementing one or more interfaces. Configurations are used to wire other components together, connecting interfaces used by components to interfaces provided by others.

11. Every nesC application is described by a top-level configuration that wires together the components used.

12. Most components in TinyOS represent services or pieces of hardware (such as the LEDs) and therefore exist only in a single instance. However, it is sometimes useful to create several instances of a component.

13. In nesC, this is achieved by declaring an abstract component with optional parameters; abstract components are created at compile-time in configurations.

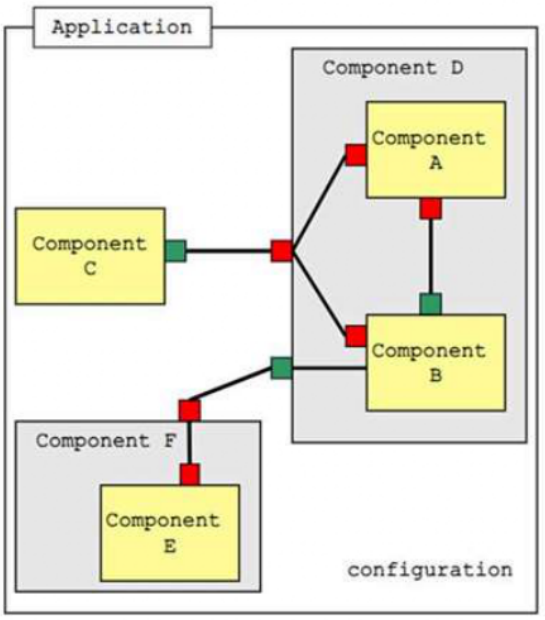
nesC model

Two key concepts

1. **Interfaces**
   - uses
   - provides
2. **Components**
   - modules
   - configurations

Application: graph of components

Interfaces: only point of access to components



Practical - 3

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

TOSSIM is a discrete event simulator, or more precisely emulator, of TinyOS at the bit level. An event is generated for each transmitted or received bit rather than one per packet. TOSSIM is a simulator for the execution of nesC model on TinyOS/MICA hardware. TOSSIM further works as an emulator of actual hardware through mapping hardware interrupts to discrete events. TOSSIM also has a built-in simulated radio model. In conclusion, TOSSIM is a high-fidelity emulator of a network of TinyOS/MICA motes.

TOSSIM is conceived to study the behaviour of TinyOS and its applications: it is not intended to profile the performance of new protocols. Accordingly, there is one significant drawback and that is the fact that every mote must run the same code. Consequently, the usability of TOSSIM for applications involving heterogeneous sensors is limited at best. As such, TOSSIM is not appropriate for heterogeneous applications. TOSSIM has been shown to handle simulations of a sensor network with around a thousand motes. Its ability to simulate larger mote counts is primarily limited by its bit-level granularity: its performance degrades as the traffic increases. Channel sampling is also simulated at the bit level and consequently the use of a CSMA protocol causes significant overhead particularly when compared to a TDMA-based MAC protocol.

The simulator engine provides a set of communication services for interacting with external applications. These services allow programs to connect to TOSSIM over a TCP socket to monitor or actuate a running simulation. Details of the ADC and radio models, such as readings and loss rates, can be both queried and set. Programs can also receive higher level information, such as packet transmissions and receptions or application-level events. TOSSIM supports the TinyOS tool-chain, making the transitions between simulated and real networks easy. Compiling to native code allows developers to use traditional tools such as debuggers in TOSSIM. As it is a discrete event simulation, users can set debugger breakpoints and step through what is normally real-time code (such as packet reception) without disrupting operation. It also provides mechanisms for other programs to interact and monitor a running simulation; by keeping monitoring and interaction external to TOSSIM, the core simulator engine remains very simple and efficient.

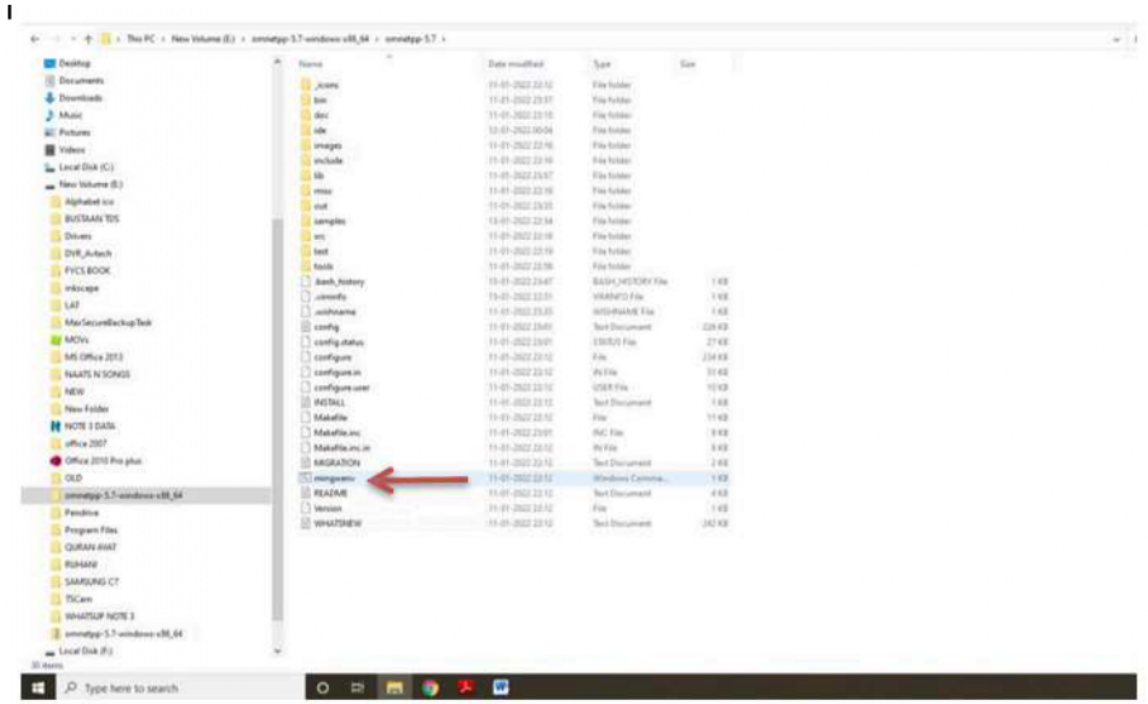
Practical - 4

Aim: Create and simulate a simple adhoc network

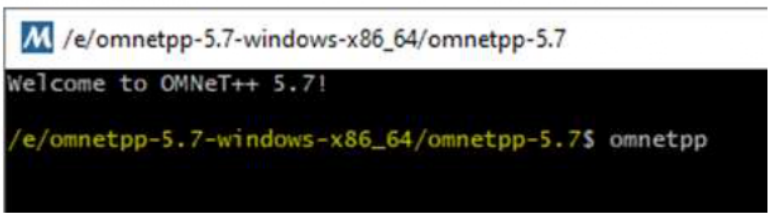
Software Used: Omnetpp 5.7, INET 4.3.6 framework

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

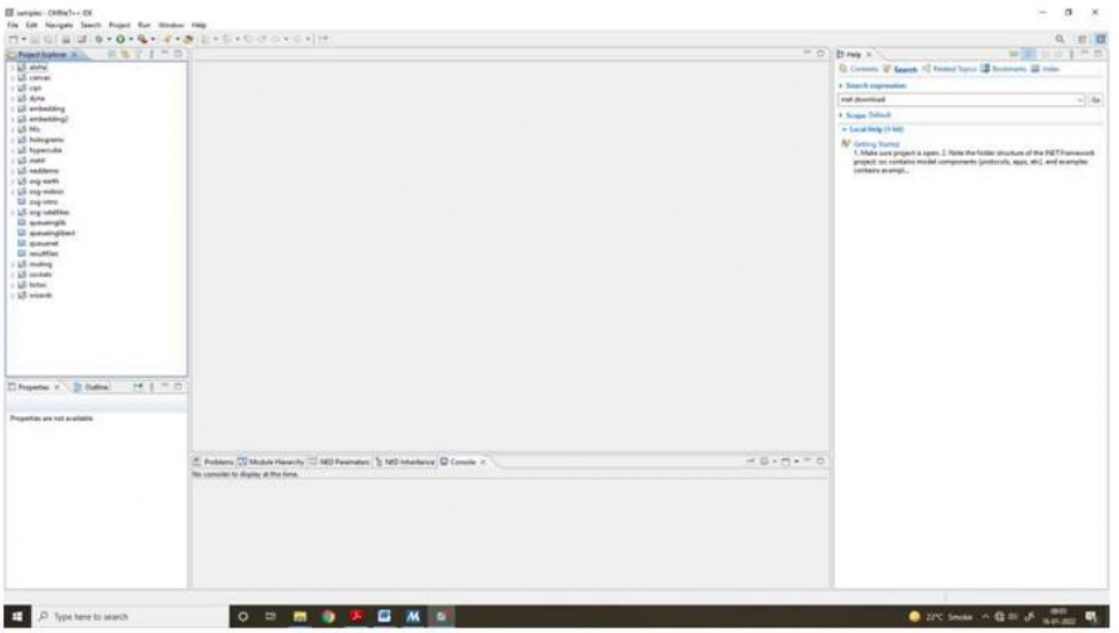
Step 1: Start the Omnetpp simulator through the following procedure Click on the file mingwenv



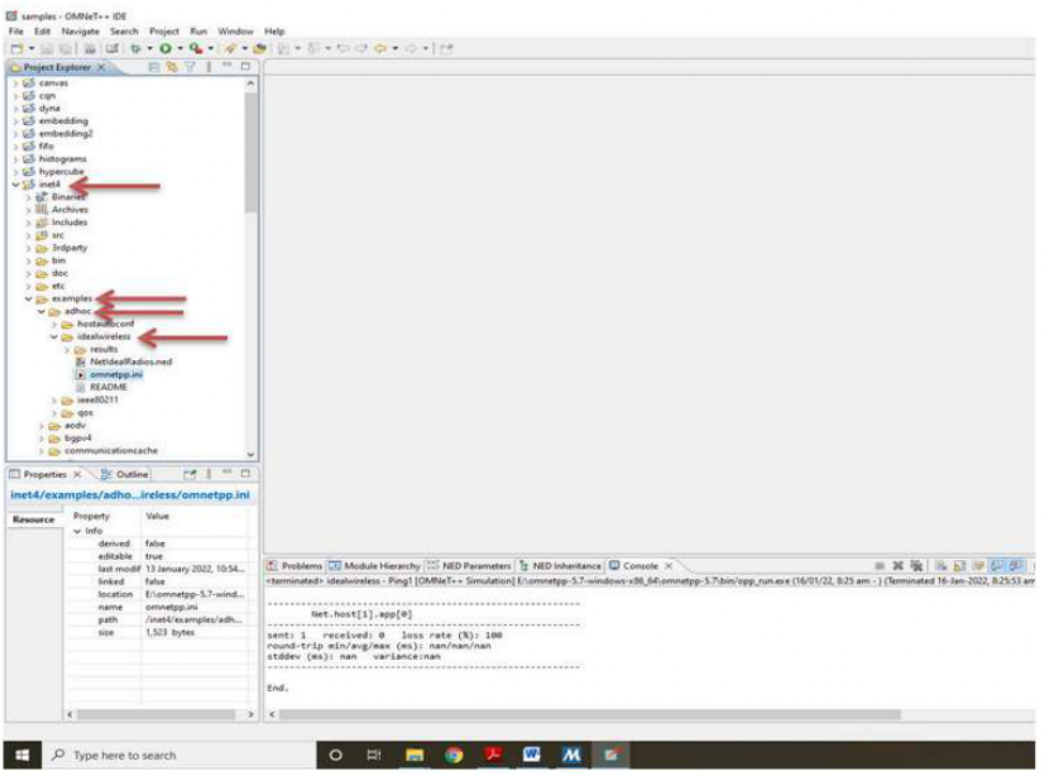
We will get the following $ prompt, type omnetpp and enter



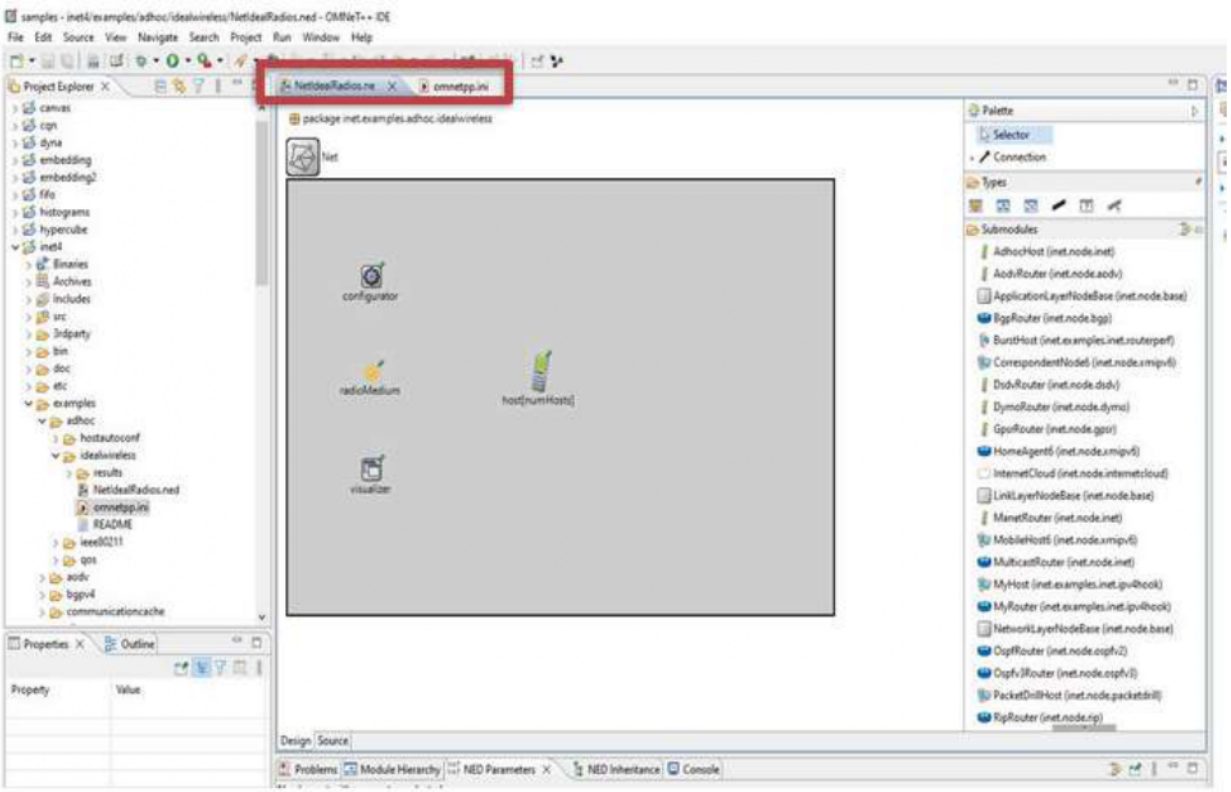
Step 2: The omnetpp simulator in now ready with the following user interface



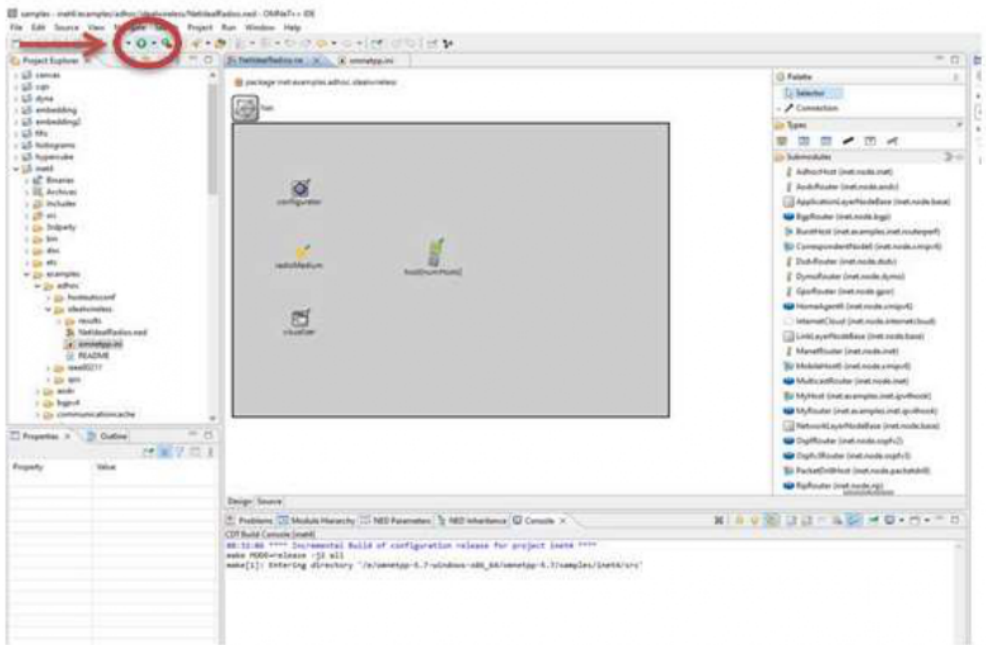
Click on inet folder, then in it click on examples, then on adhoc and then on idealwireless as given



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

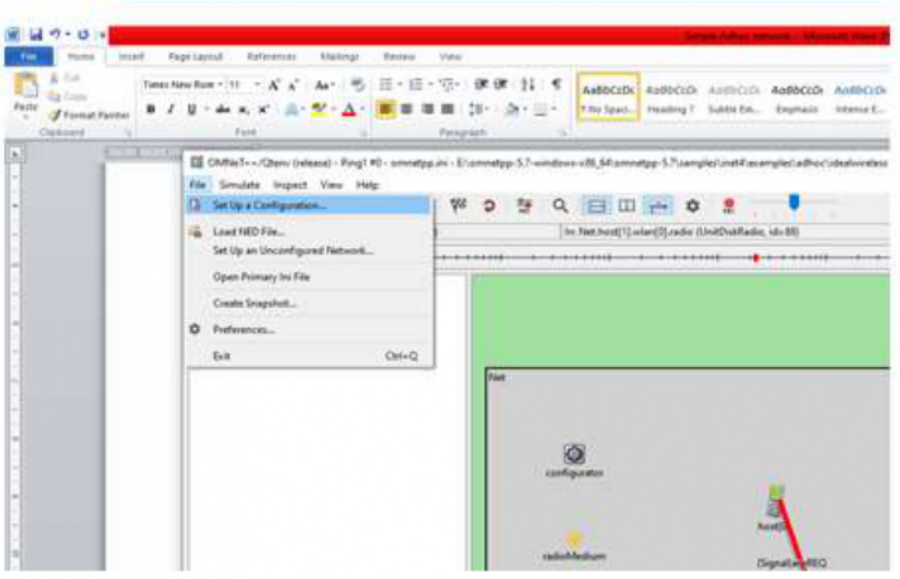


Step 4: Now we run the simulation



Step 5: After running the simulation we get the following

The number of hosts can be increased by the following

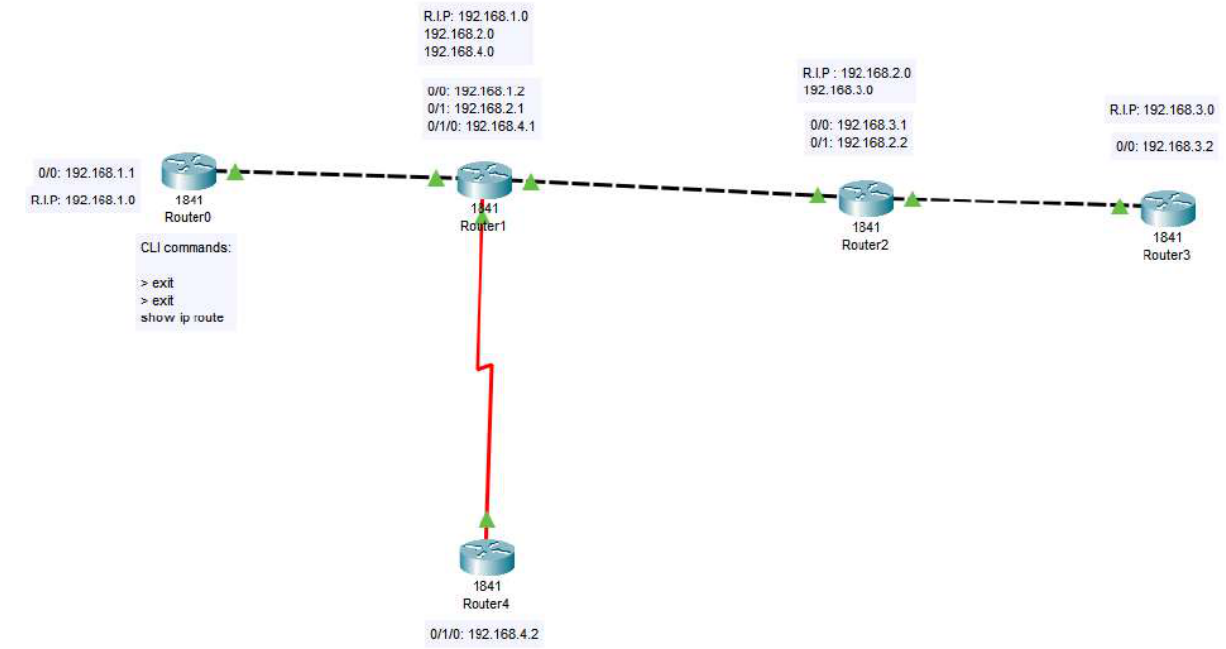


In this we get a dropdown menu, select n host option and enter the required hosts The following simulation has 7 hosts


flat
configurator
radioMedium
visualizer
host[0]
host[1]
host[2]
host[3]
host[4]
host[5]
host[6]

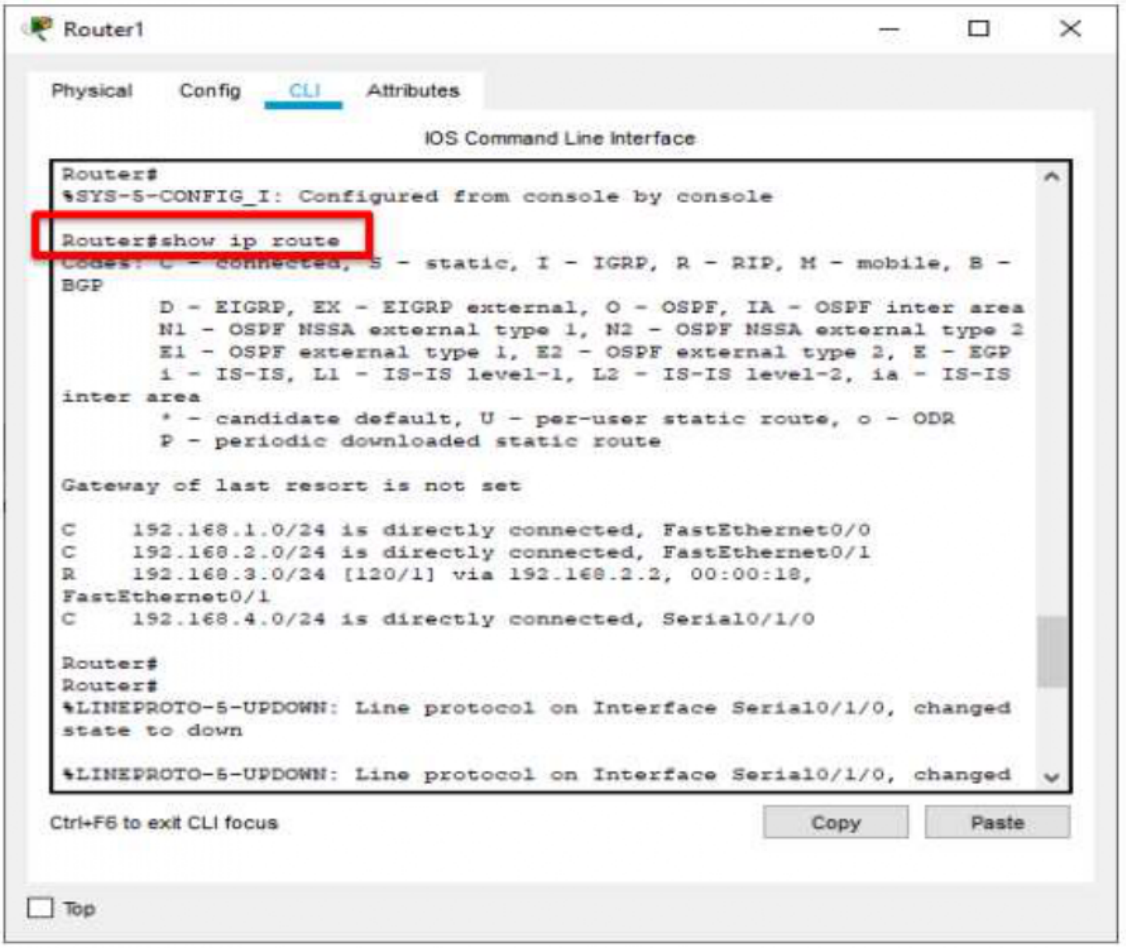
Practical - 5

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



,

The ip addresses are configured on the given interfaces of the Routers.
The Routing path is also set using RIP.
The following command is executed in the CLI mode of Router1



,

We get the following Routing information from Router1, which is the required output.

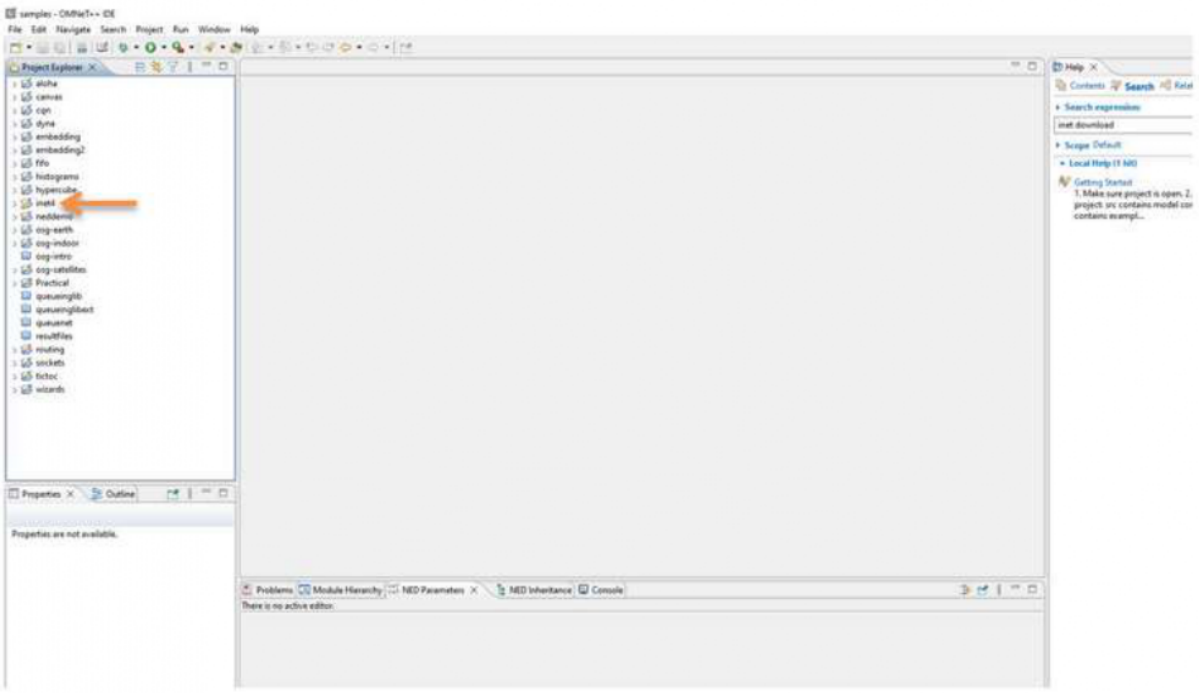
C 192.168.1.0/24 is directly connected, FastEthernet0/0
C 192.168.2.0/24 is directly connected, FastEthernet0/1
R 192.168.3.0/24 [120/1] via 192.168.2.2, 00:00:18, FastEthernet0/1
C 192.168.4.0/24 is directly connected, Serial0/1/0

Practical - 6

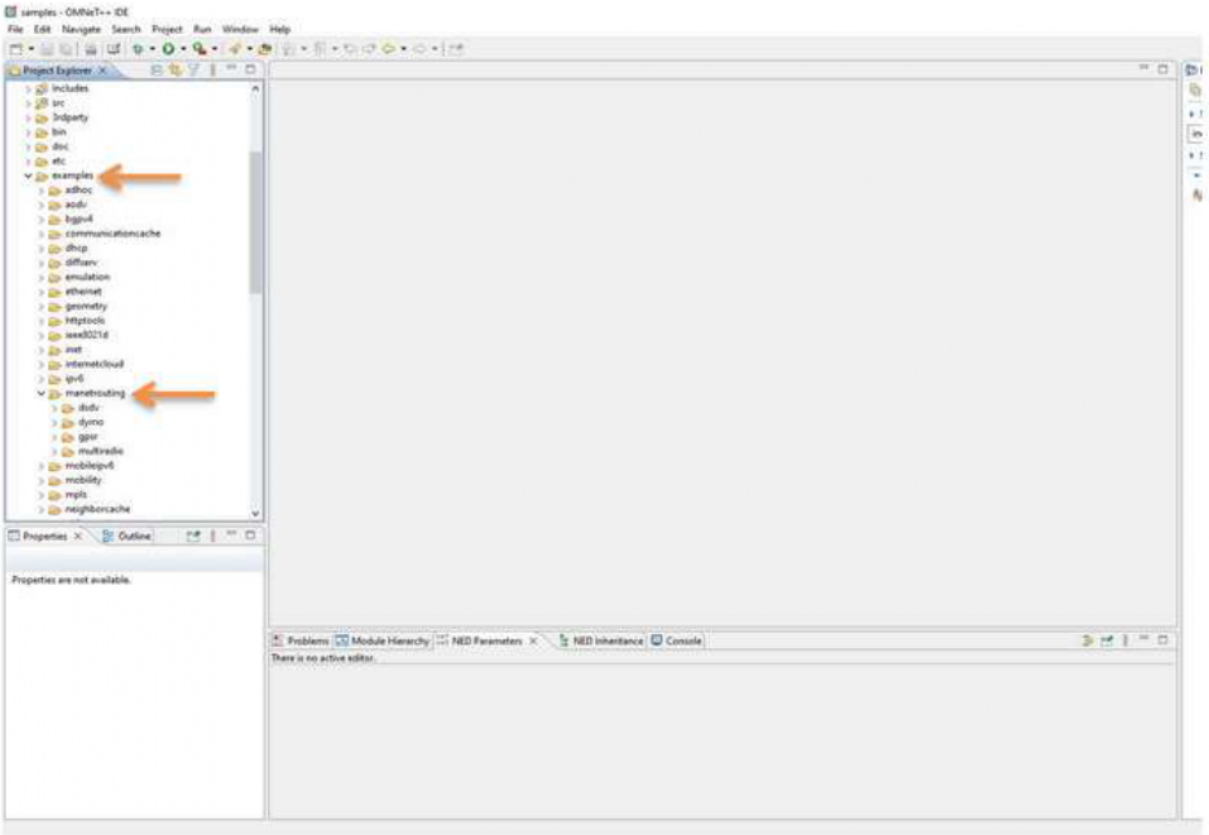
Aim: Create a basic MANET implementation simulation for Packet animation and Packet Trace
Software Used: Omnet++ 5.7, INET 4.3.6 framework

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

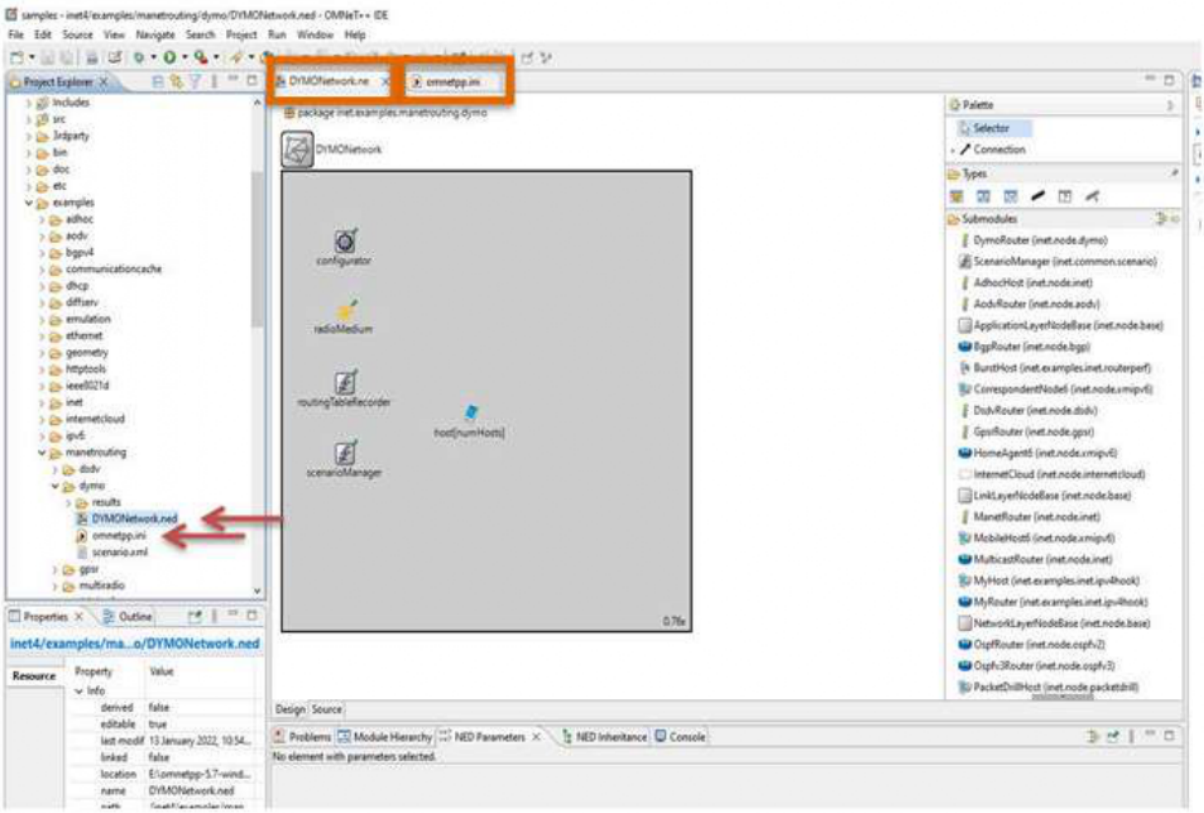
Step 1: Open the Omnet++ software and click on inet4 folder



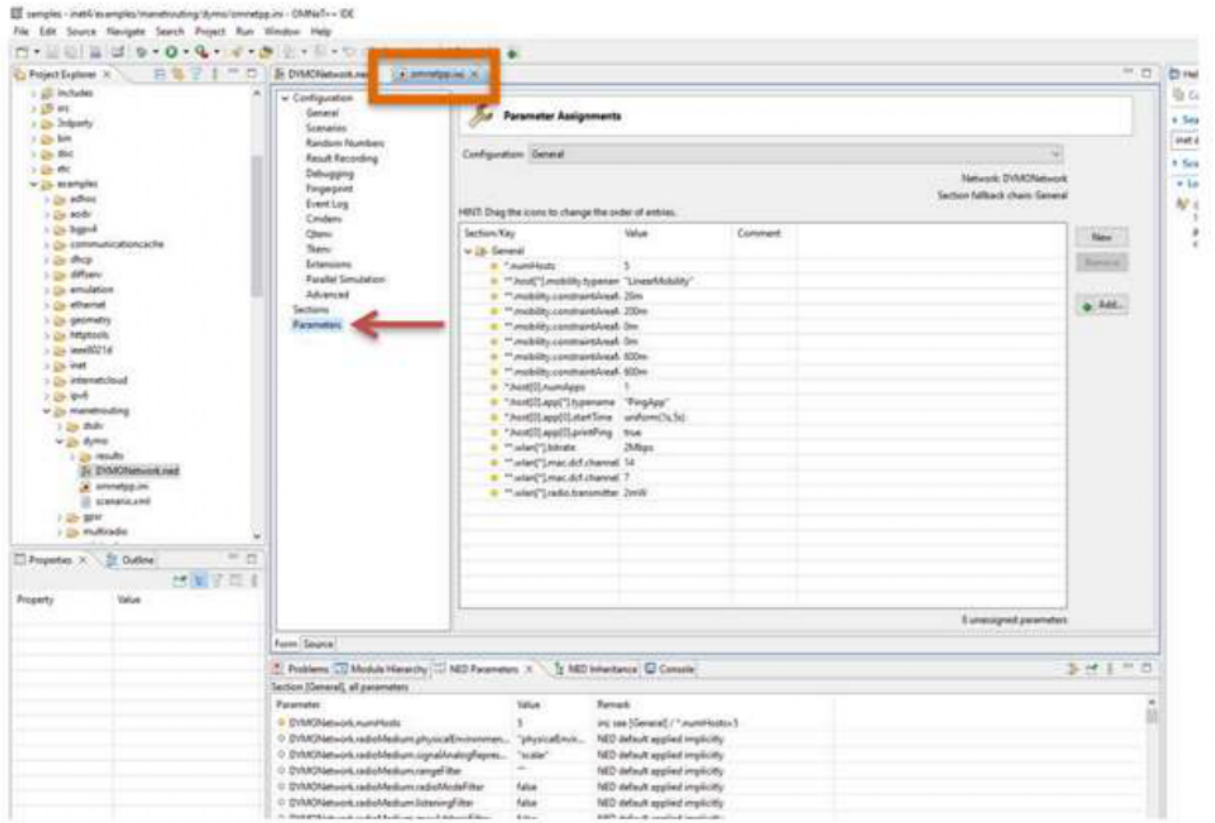
Step 2: Now select the examples folder and then in that folder select manetrouting folder



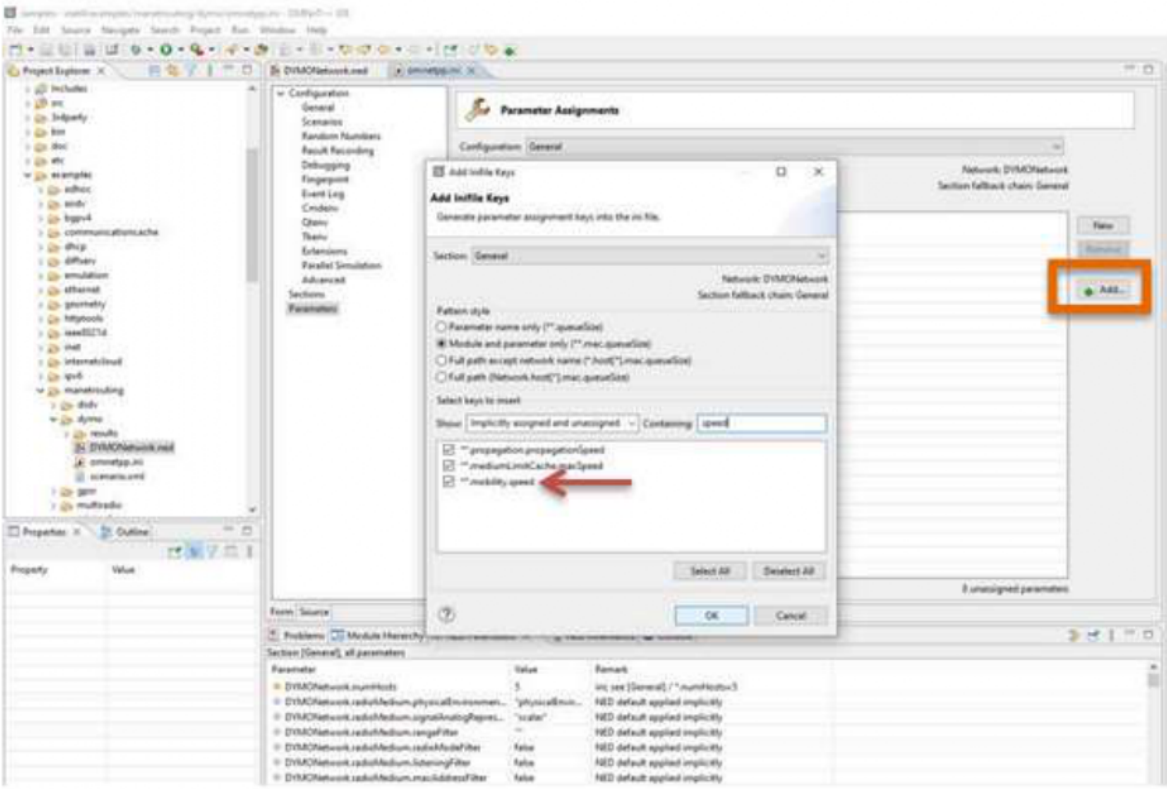
Step 3 : In manetrouting folder click dymo folder and then load the DYMONetwork.ned and omnetpp.ini files by double clicking



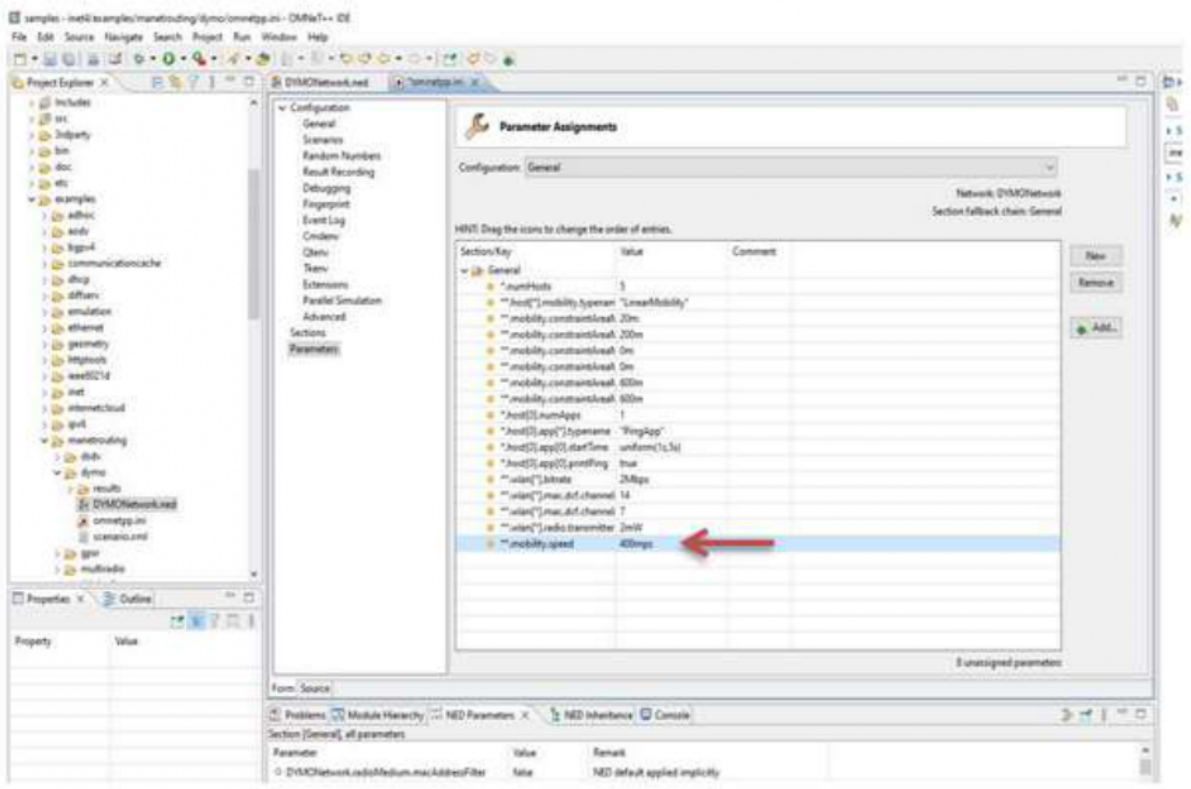
Step 4: Select omnetpp.ini file and click on parameters, we need to add mobility to the nodes



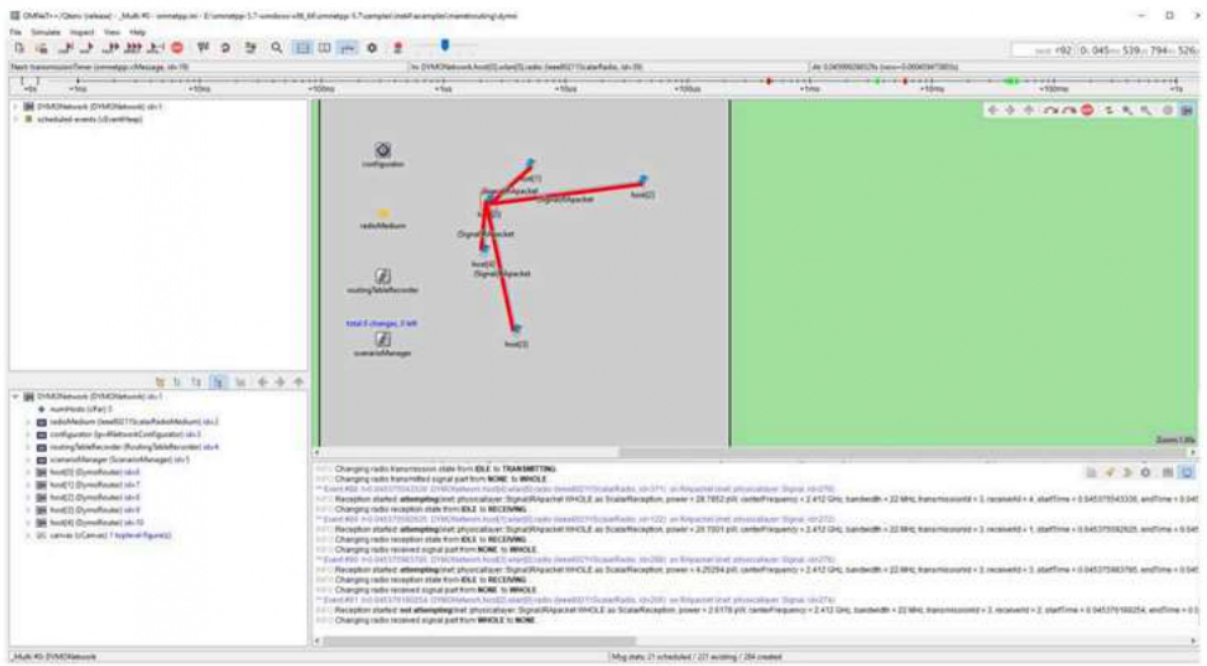
Step 5: For adding a new parameter click on add button and add the parameter **.mobility.speed



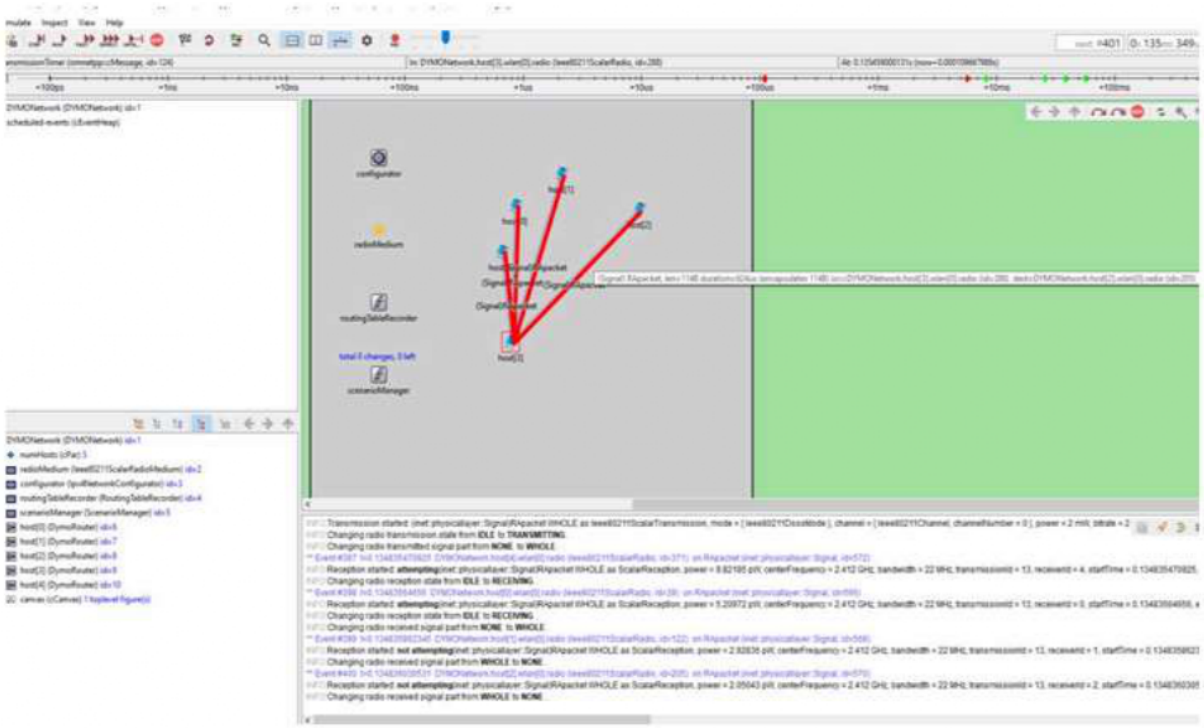
Step 6: Set the value for **.mobility.speed = 400mps



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



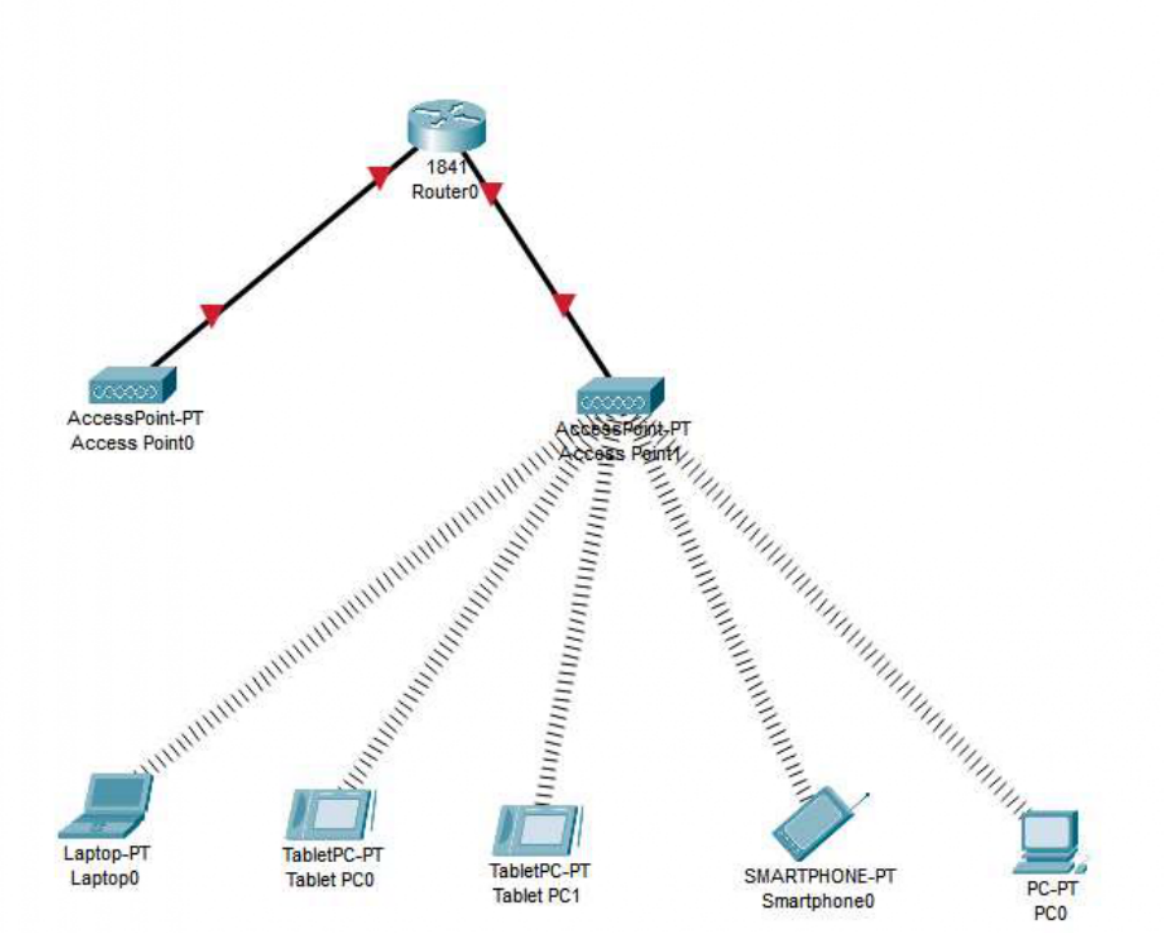
Hence the given MANET has been simulated with 5 hosts

Practical - 7

Aim : Implement a Wireless Sensor network simulation

STEPS:

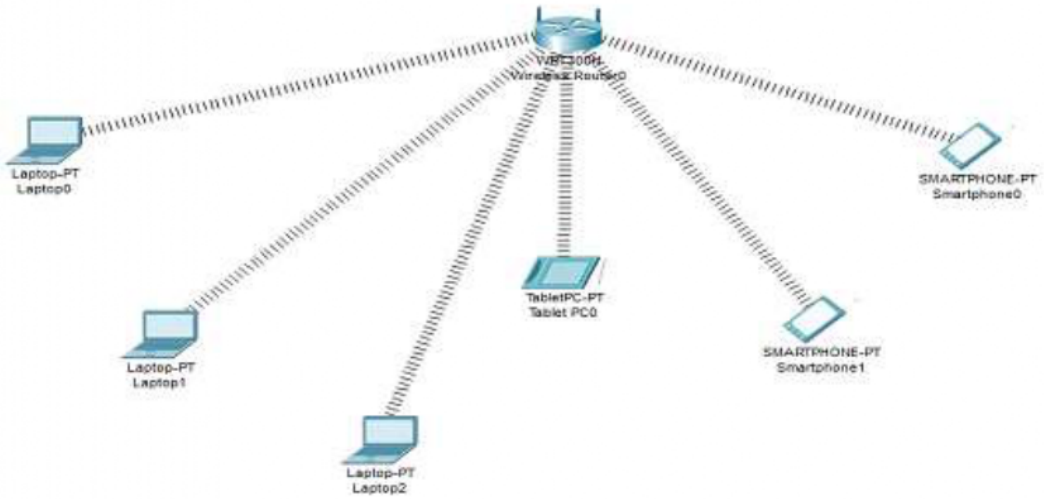
- Create a network using 1 Router-PT, 2 AccessPoint-PT-N and some end devices like tablets, smartphones and laptop.
- Connect access points and router using coaxial cable.
- In laptop connect Linksys WPC300N (The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.).
- if connecting PC then connect PT-HOST-NM-1W (The PT-HOST-NM-1W module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.)



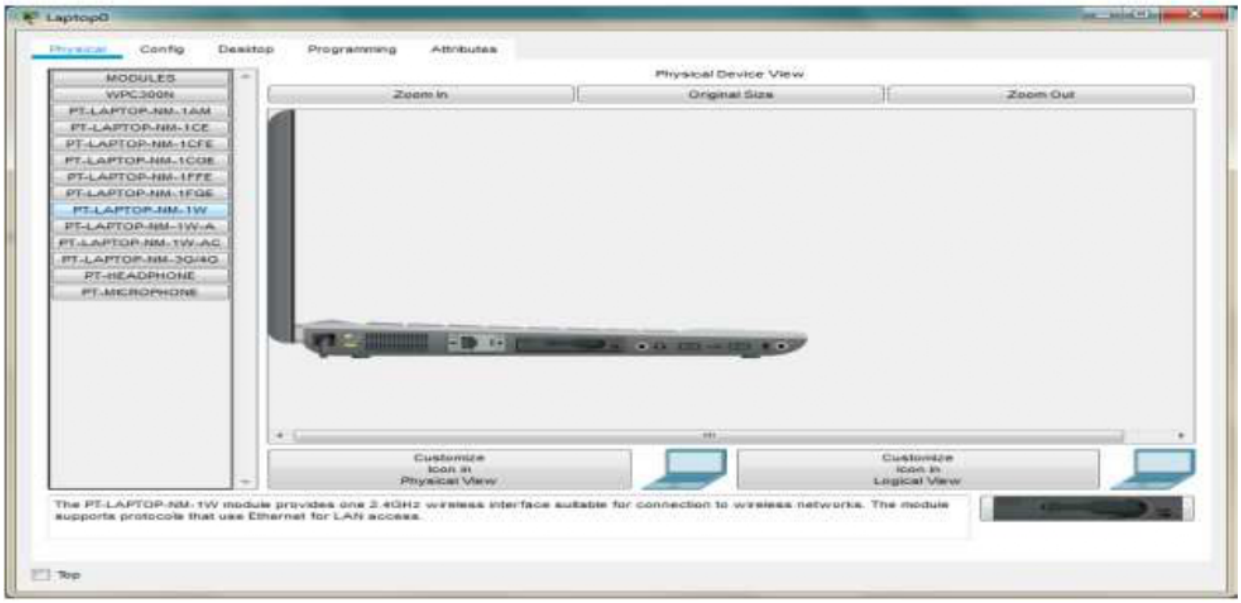
Practical - 8

Aim: Create MAC protocol simulation implementation for wireless sensor Network.

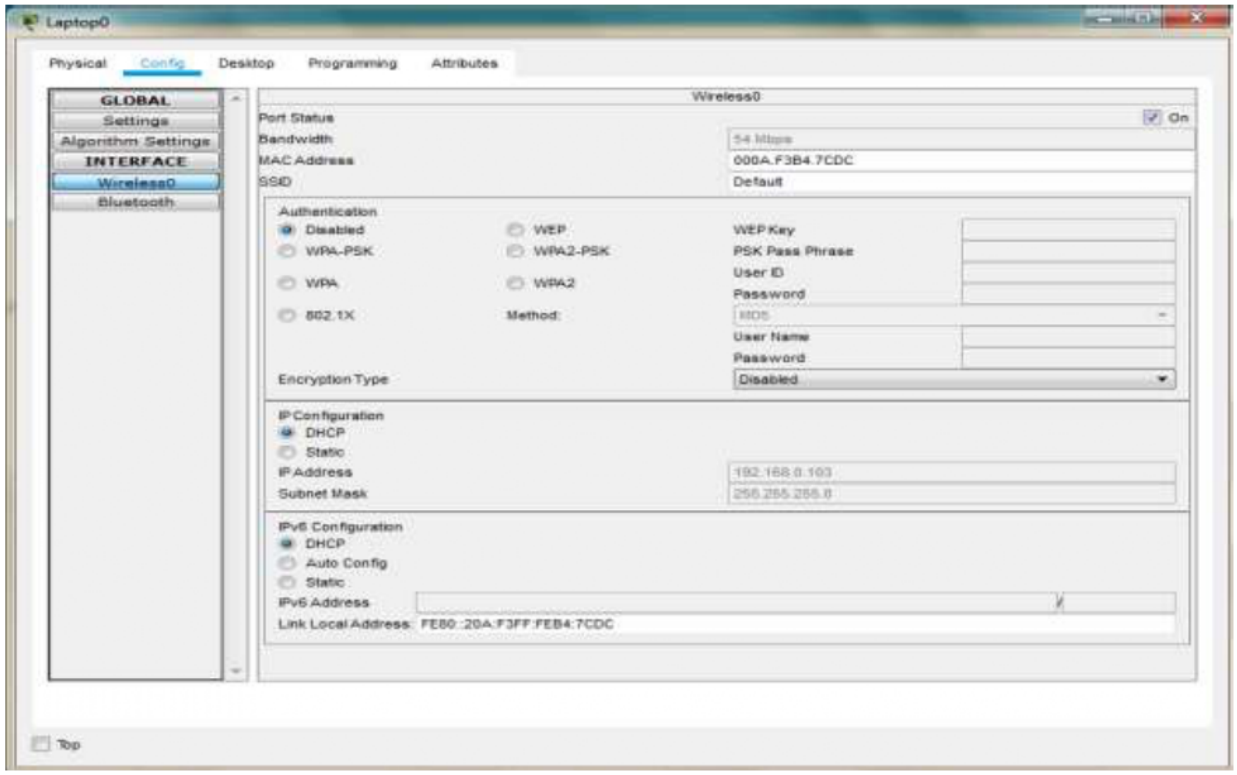
Software Used:
Cisco Packet Tracer 7.2.0.0226

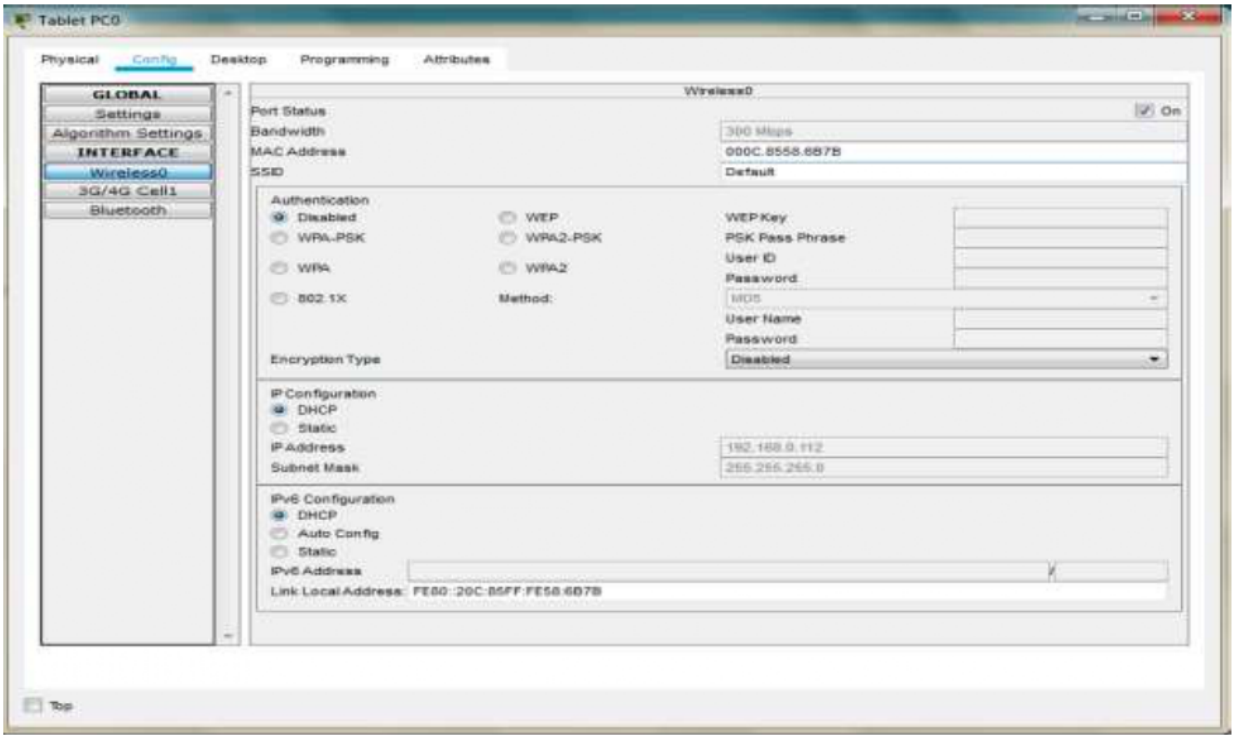


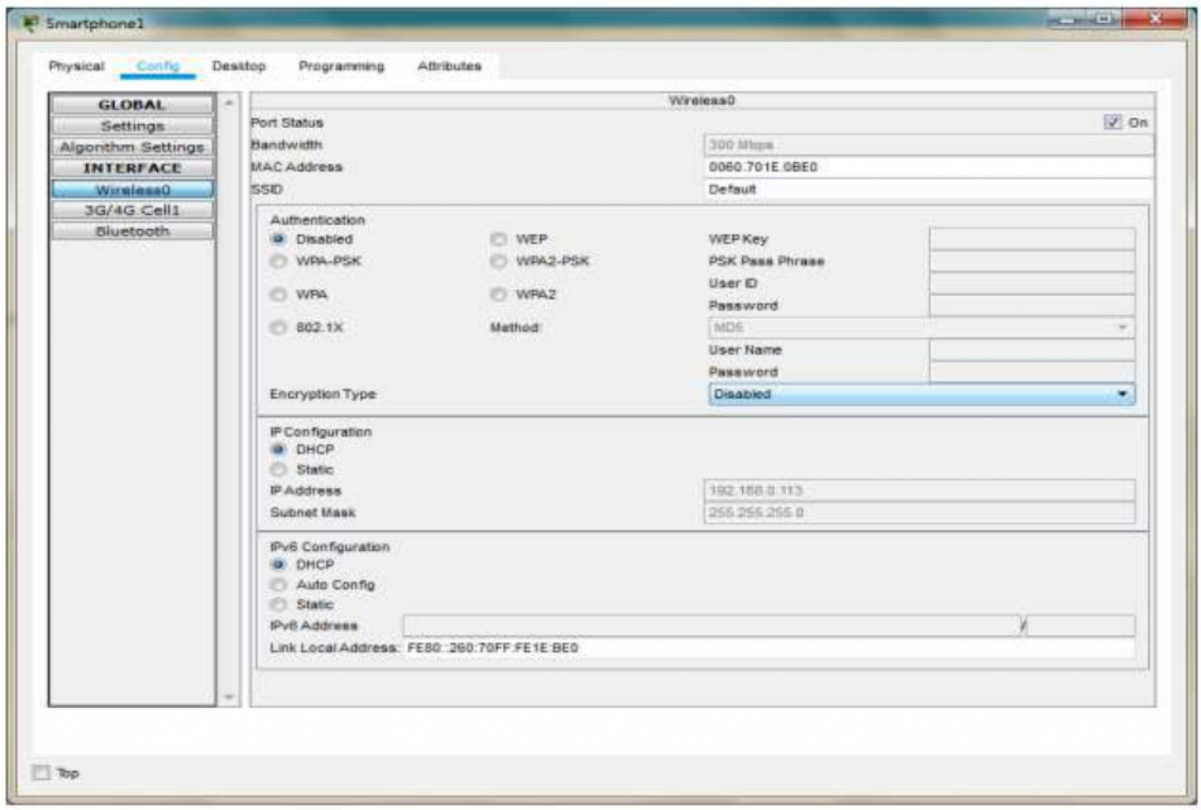
Smart phones and Tablet have a wireless interface by default, while the laptop does not has a wireless interface, we need to add the interface in all the laptops.
Adding the wireless interface to each Laptops as follows



Copy the MAC address of each component as follows



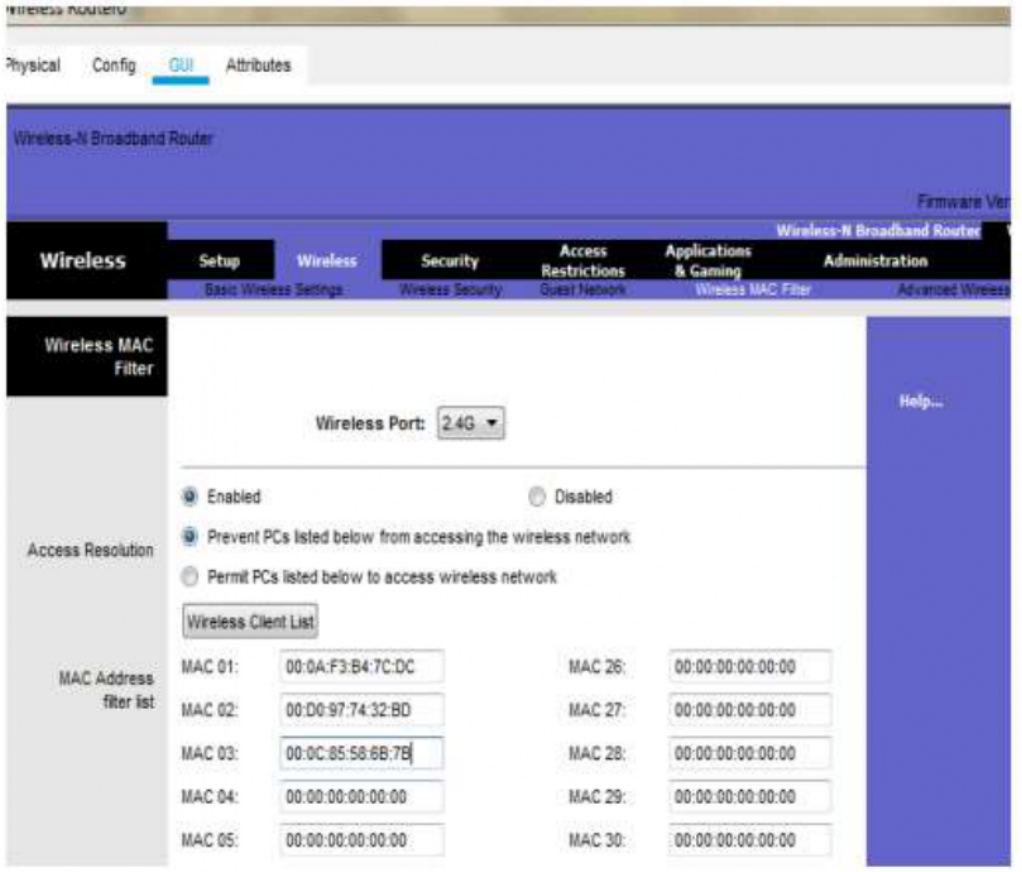




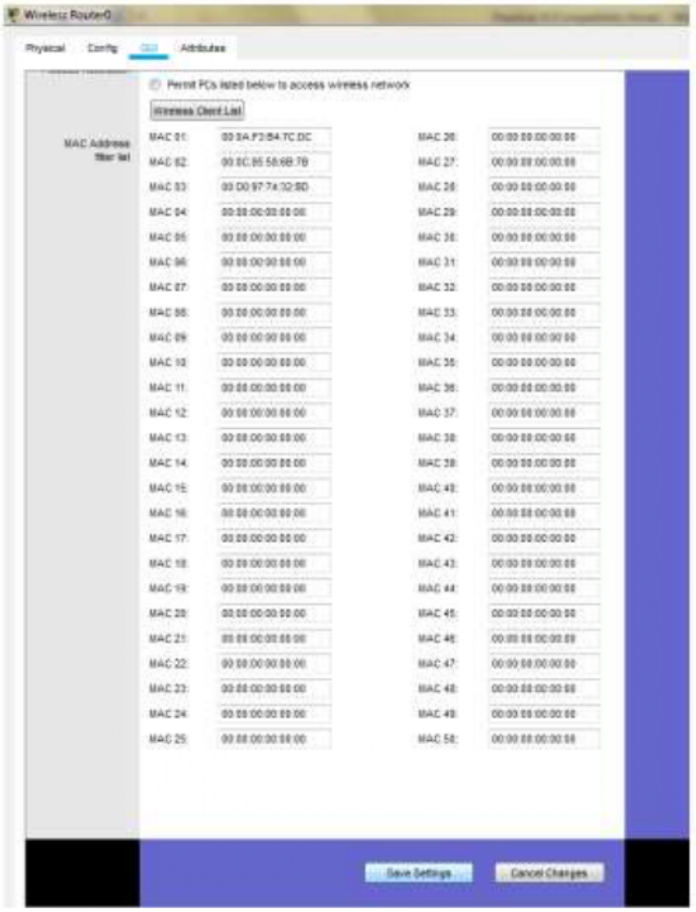
We note the following MAC addresses and convert them to the following form

Component	MAC Address	Converted MAC address
Laptop0	000A.F3B4.7CDC	00:0A:F3:B4:7C:DC
Laptop1	0001.4269.6539	00:01:42:69:65:39
Laptop2	0060.5CB8.B919	00:60:5C:B8:B9:19
TabletPC	000C.8558.6B7B	00:0C:85:58:6B:7B
SmartPhone0	00D0.9774.32BD	00:D0:97:74:32:BD
SmartPhone1	0060.701E.0BE0	00:60:70:1E:0B:E0

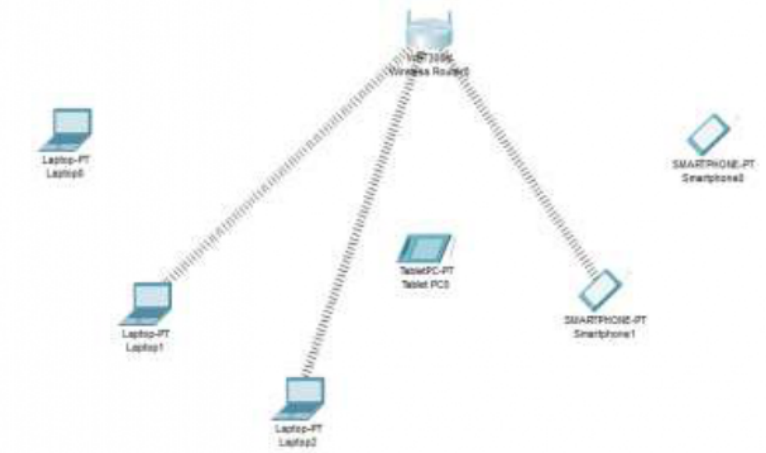
Now we add few addresses in the wireless MAC filter of the Wireless Router and then use the given options for either allow or deny the Wireless access



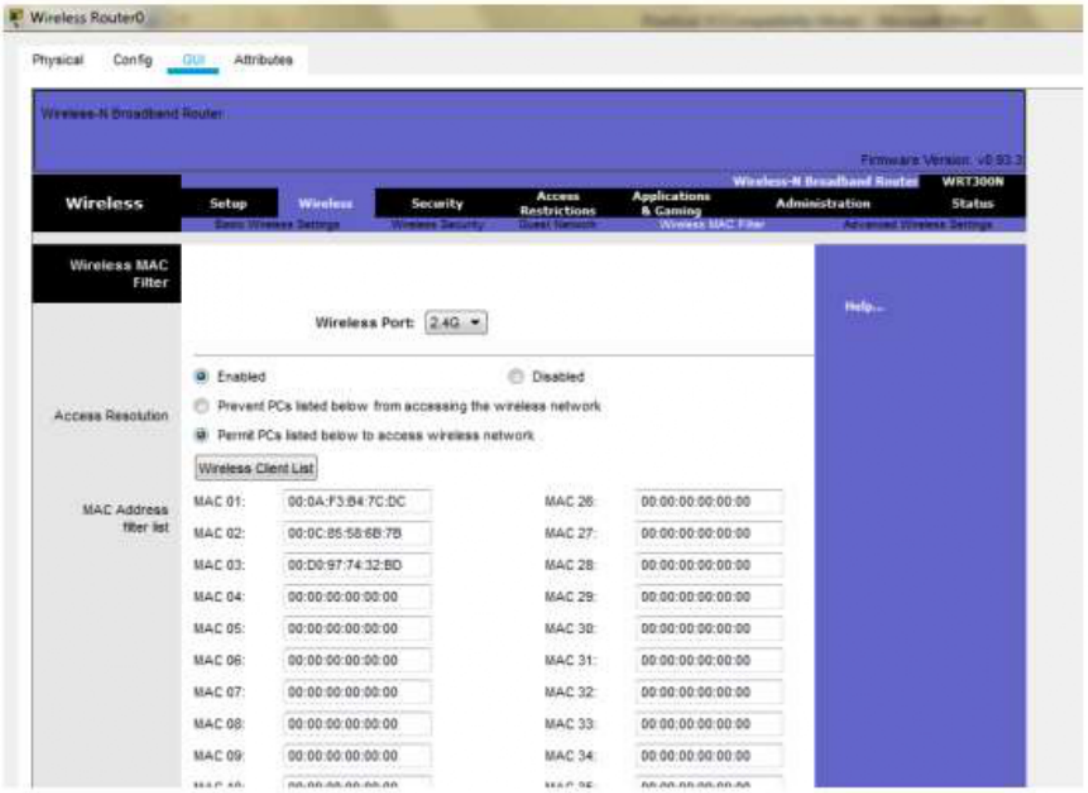
As seen in above screen shot we add the MAC address of Laptop0, TabletPC SmartPhone0 in the list so as to deny them accessing the Wireless network and then save the settings



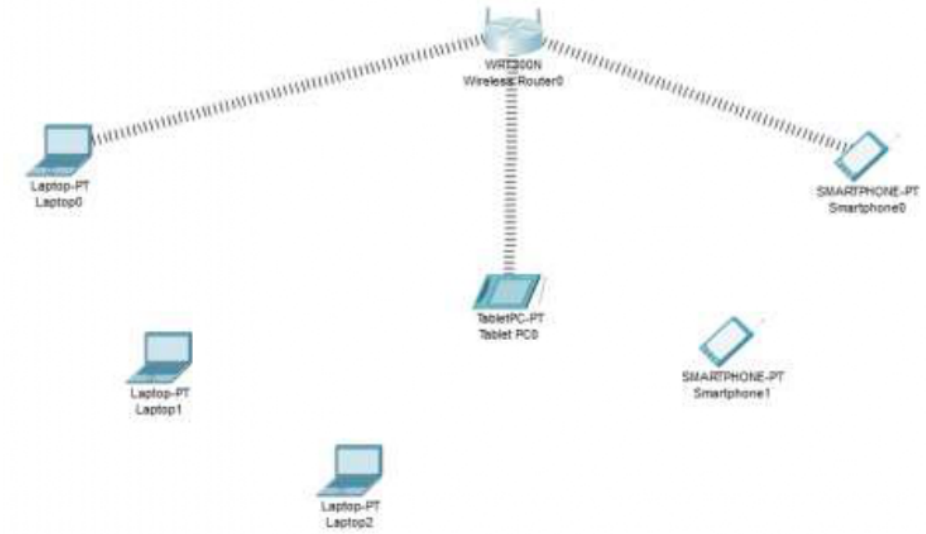
The result so obtained is as shown; the three devices denied any wireless connectivity



Similarly we can change the setting so that the above devices get wireless connectivity and the remaining devices do not get the wireless connectivity



And save the setting and get the following

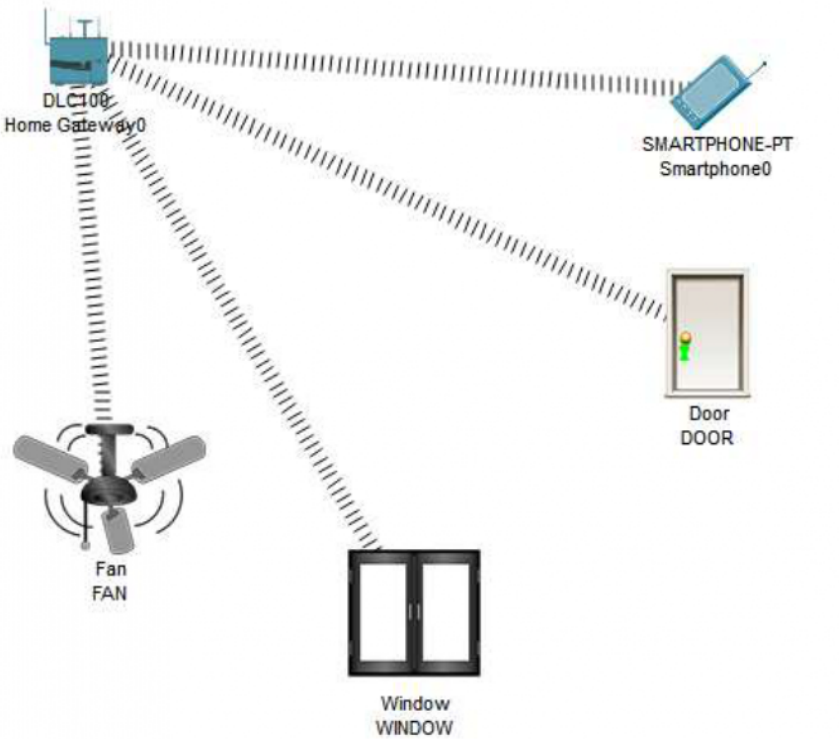


Practical - 9

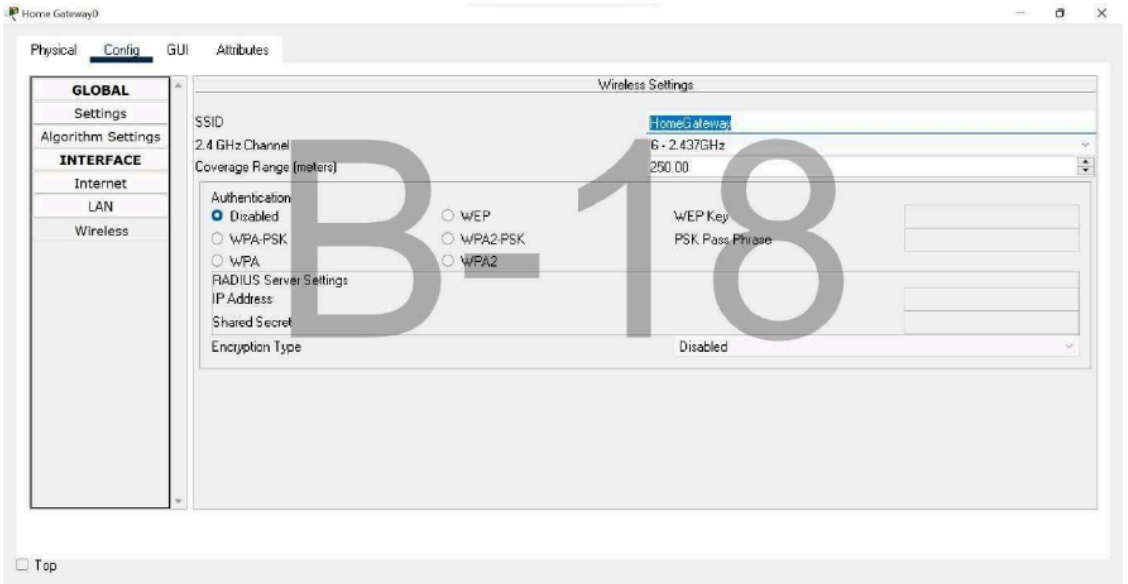
Aim: Simulate Mobile Adhoc Network with Directional Antenna

STEPS:

1. Create a network using Home gateway, and IOT devices
2. The Home Gateway provides 4 ethernet ports as well as a wireless access point configured with the "HomeGateway" ssid on channel 6. WEP / WPA-PSK / WPA2 enterprise can be configured to secure wireless connections. The picture below shows 4 IOE Things attached to a Home Gateway The Home Gateway is connected to the Internet through it's Internet WAN ethernet port.
3. For fan go to config -> Advance -> IO config -> network adapter:1W
4. Copy the SSID of home gateway (config -> wireless -> SSID)
5. In fan and window select home gateway in IO config
6. Go to smartphone -> desktop -> IOT monitor -> control the IOT devices




IoT0
Specifications
I/O Config
Physical
Config
Thing Editor
Programming
Attributes
Network Adapter
PT-IOT-NM-1W
Network Adapter 2
None
Digital Slots
1
Analog Slots
0
USB Ports
0
Bluetooth
Built-in
Desktop
Show
Usage
Smart Device
Component
Top
Advanced

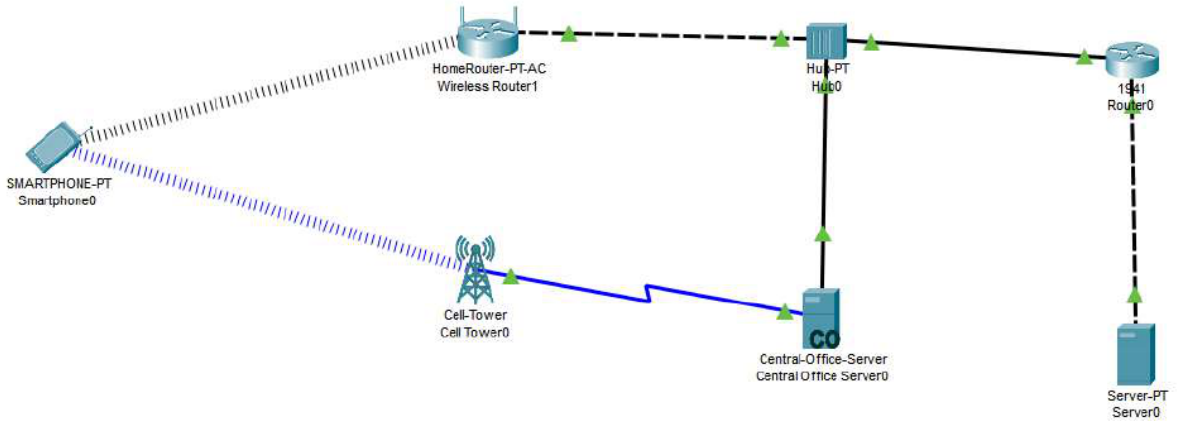

Home Gateway0
Physical
Config
GUI
Attributes
GLOBAL
Settings
Algorithm Settings
INTERFACE
Internet
LAN
Wireless
Wireless Settings
SSID
HomeGateway
2.4 GHz Channel
6 - 2.437GHz
Coverage Range (meters)
250.00
Authentication
Disabled
WEP
WEP Key
WPA-PSK
WPA2-PSK
PSK Pass Phrase
WPA
WPA2
RADIUS Server Settings
IP Address
Shared Secret
Encryption Type
Disabled
Top

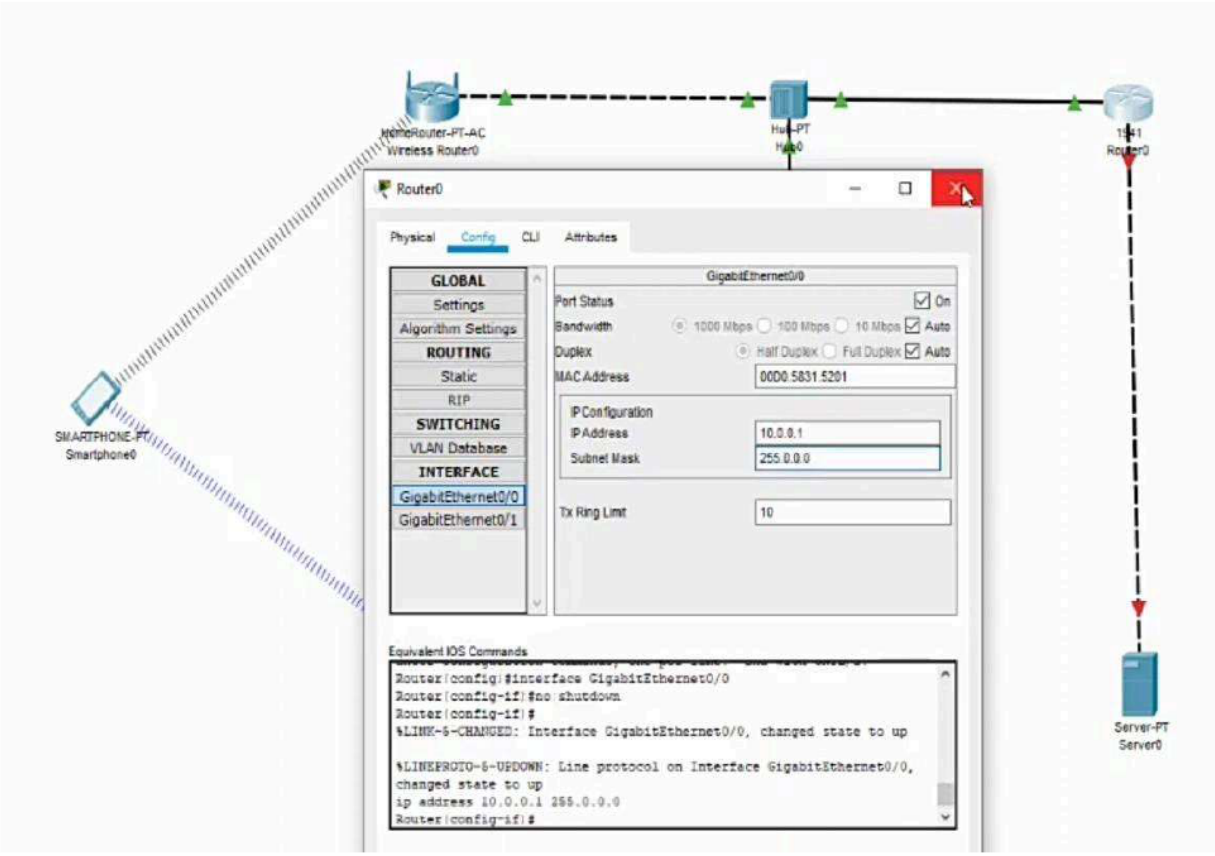

IoT0
Specifications
Physical
Config
Attributes
GLOBAL
Settings
Algorithm Settings
Files
INTERFACE
Wireless0
Bluetooth
Gateway/DNS IPv6
Automatic
Static
Default Gateway
DNS Server
IoT Server
None
Home Gateway
Remote Server
Server Address
User Name
Password
Refresh
Top
Advanced

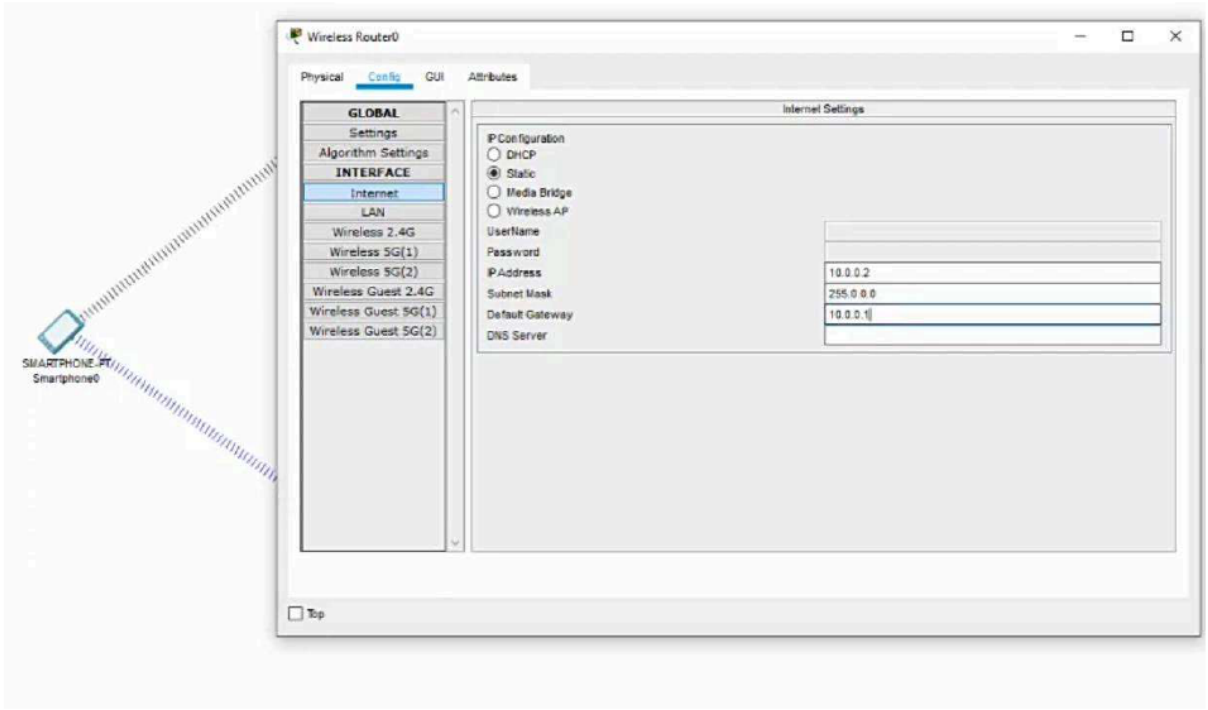

Smartphone0
Physical
Config
Desktop
Programming
Attributes
IoT Monitor
IoT Server - Devices
Home | Conditions | Editor | Log Out
IoT0 (PTT0810823I-)
Ceiling Fan
Status
Off
Low
High
IoT1 (PTT0810WOAX-)
Window
On
IoT2 (PTT0810P1PZ-)
Door
Open
Lock
Unlock
Lock
Top

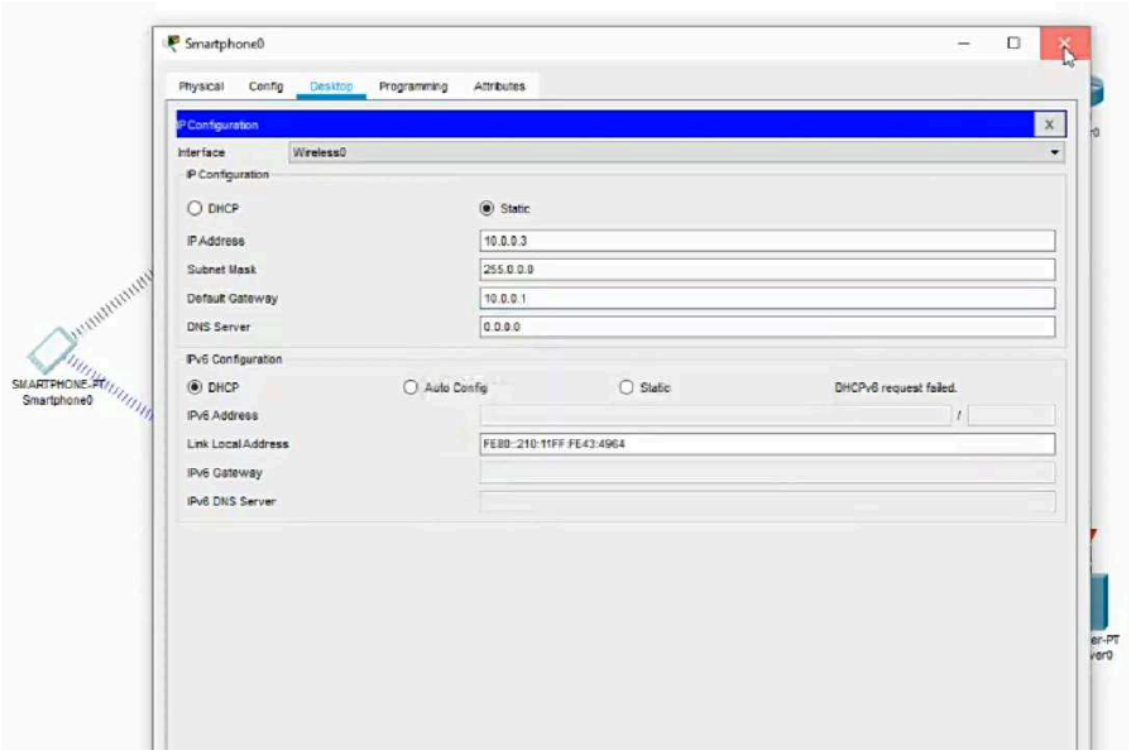
Practical - 10

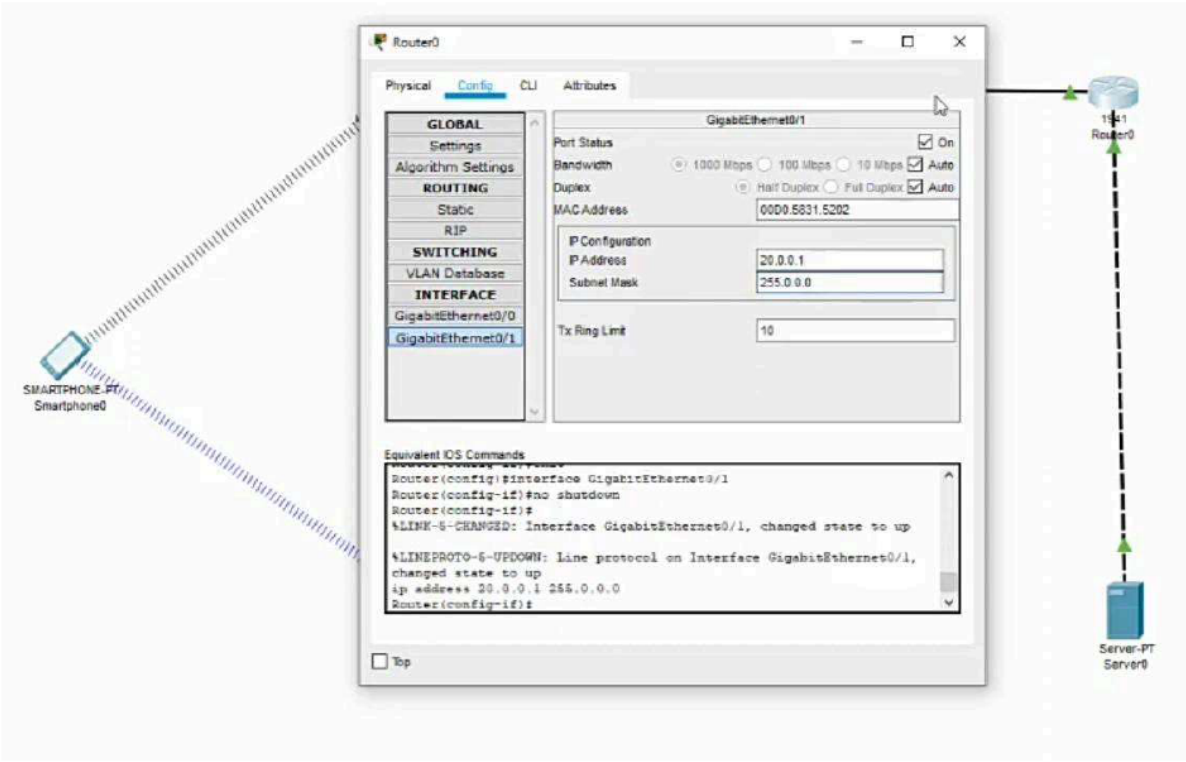
Aim: Create a mobile network using Cell Tower, Central Office Server, Web browser and Web Server. Simulate connection between them

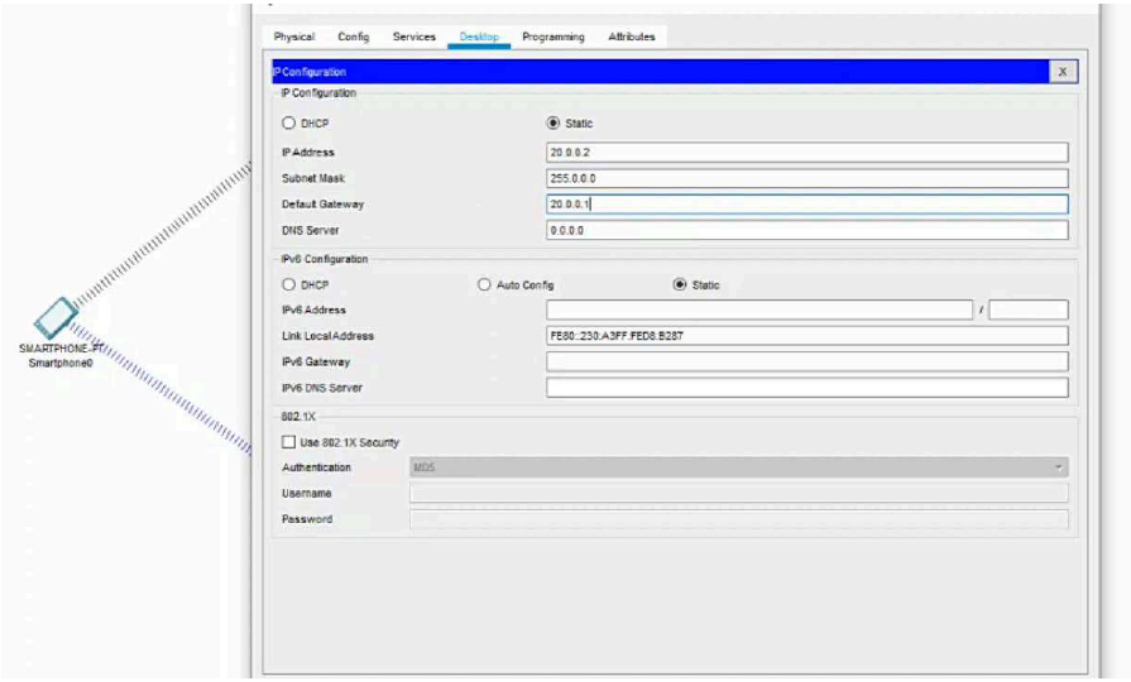


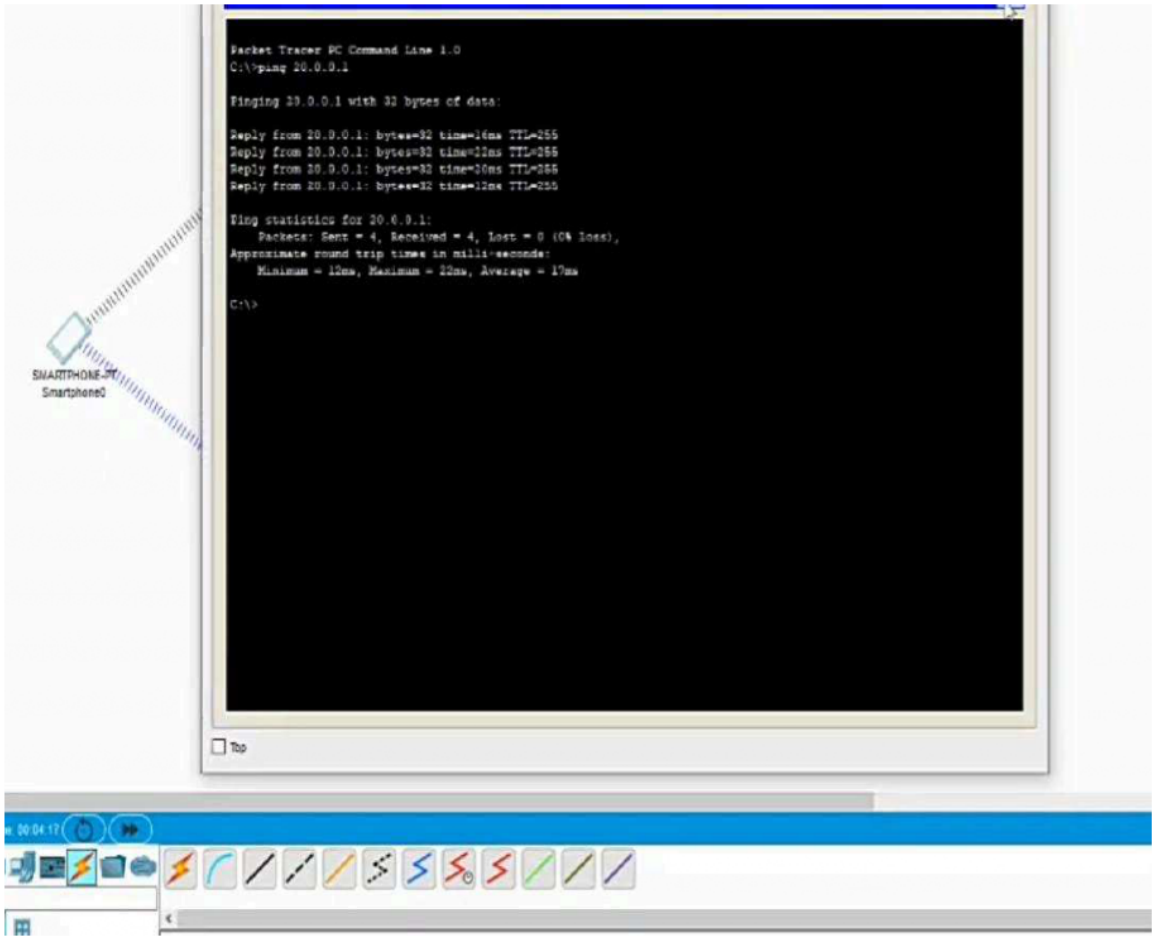


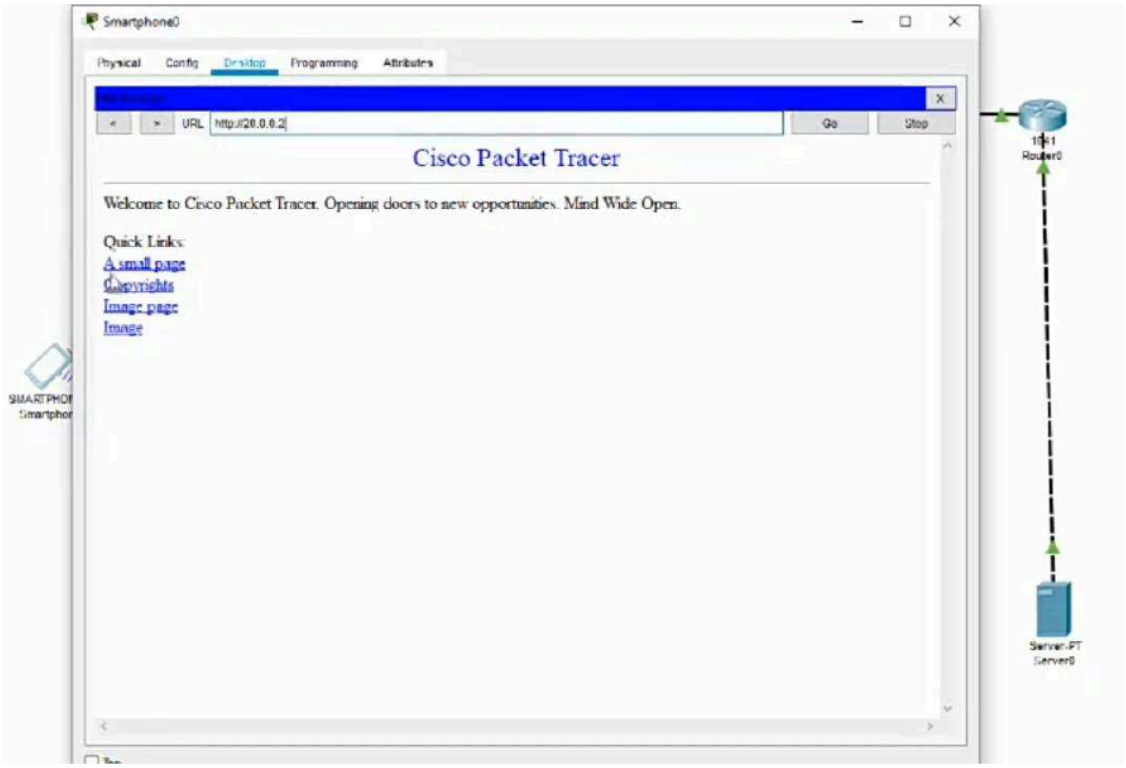

Wireless Router0
Physical
Config
GUI
Attributes
GLOBAL
Settings
Algorithm Settings
INTERFACE
Internet
LAN
Wireless 2.4G
Wireless 5G(1)
Wireless 5G(2)
Wireless Guest 2.4G
Wireless Guest 5G(1)
Wireless Guest 5G(2)
Internet Settings
IP Configuration
DHCP
Static
Media Bridge
Wireless AP
UserName
Password
IP Address
10.0.0.2
Subnet Mask
255.0.0.0
Default Gateway
10.0.0.1
DNS Server
Top
SMARTPHONE-PT
Smartphone0

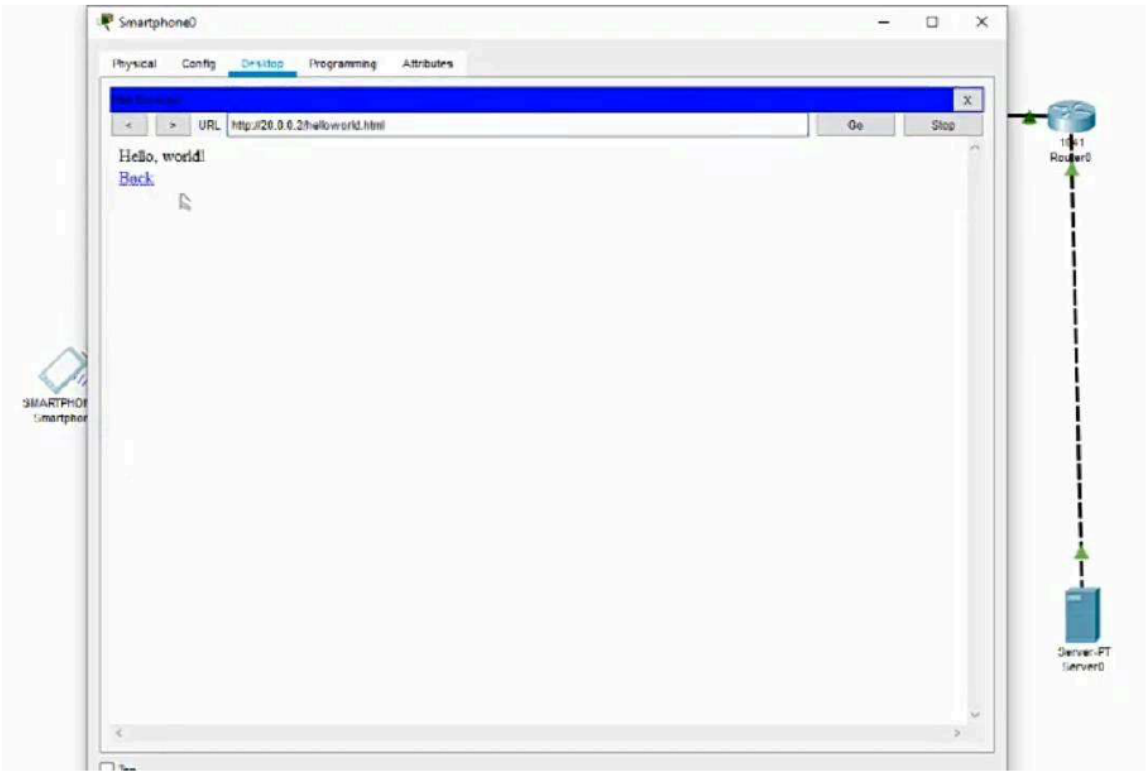

Smartphone0
Physical
Config
Desktop
Programming
Attributes
IP Configuration
Interface
Wireless0
IP Configuration
DHCP
Static
IP Address
10.0.0.3
Subnet Mask
255.0.0.0
Default Gateway
10.0.0.1
DNS Server
0.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
DHCPv6 request failed.
IPv6 Address
Link Local Address
FE80::210:11FF:FE43:4964
IPv6 Gateway
IPv6 DNS Server
SMARTPHONE-PT
Smartphone0


Router0
Physical
Config
CLI
Attributes
GLOBAL
Settings
Algorithm Settings
ROUTING
Static
RIP
SWITCHING
VLAN Database
INTERFACE
GigabitEthernet0/0
GigabitEthernet0/1
GigabitEthernet0/1
Port Status
On
Bandwidth
1000 Mbps
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
00D0.5831.5202
IP Configuration
IP Address
20.0.0.1
Subnet Mask
255.0.0.0
Tx Ring Limit
10
Equivalent IOS Commands
Router(config)#interface GigabitEthernet0/1
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up
ip address 20.0.0.1 255.0.0.0
Router(config-if)#
Top
SMARTPHONE-PT
Smartphone0
1941
Router0
Server-PT
Server0


Physical
Config
Services
Desktop
Programming
Attributes
IP Configuration
IP Configuration
DHCP
Static
IP Address
20.0.0.2
Subnet Mask
255.0.0.0
Default Gateway
20.0.0.1
DNS Server
0.0.0.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address
FE80::230:A3FF:FED8:B287
IPv6 Gateway
IPv6 DNS Server
802.1X
Use 802.1X Security
Authentication
MD5
Username
Password
SMARTPHONE-PT
Smartphone0


Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Reply from 20.0.0.1: bytes=32 time=14ms TTL=255
Reply from 20.0.0.1: bytes=32 time=11ms TTL=255
Reply from 20.0.0.1: bytes=32 time=10ms TTL=255
Reply from 20.0.0.1: bytes=32 time=12ms TTL=255
Ping statistics for 20.0.0.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 10ms, Maximum = 14ms, Average = 11ms
C:\>
Top
Smartphone0


Smartphone0
Physical
Config
Desktop
Programming
Attributes
URL
http://20.0.0.2
Go
Stop
Cisco Packet Tracer
Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.
Quick Links:
A small page
Copyrights
Image page
Image
Router0
Server-PT
Server0


Smartphone0
Physical
Config
Desktop
Programming
Attributes
URL
http://20.0.0.2/helloworld.html
Go
Stop
Hello, world!
Back
Router0
Server-PT
Server0