**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Title** | **Page no** |
| 1 | Create a network with three routers with RIPv2 and each router associated network will have minimum three PC. Show connectivity |  |
| 2 | Create a network with three routers with OSPF and each router associated network will have minimum three PC. Show connectivity |  |
| 3 | Create a network with three routers with BGP and each router associated network will have minimum three PC. Show connectivity. |  |
| 4 | Configure DHCP server and client for DNS service. |  |
| 5 | Create virtual PC based network using virtualization software and virtual NIC |  |
| 6 | Create network cloud and hosts |  |
| 7 | Create simple Adhoc network |  |
| 8 | Create MANET simulation for AODVUU Network |  |
| 9 | Create Single mobile network |  |
| 10 | Create wireless network in OMNET++ |  |

**PRACTICAL 1**

**Routing Information Protocol (RIP)**

**Aim –** Create a network with three routers with RIPv2 and each router associated network will have minimum three PC. Show connectivity.

**Theory –**

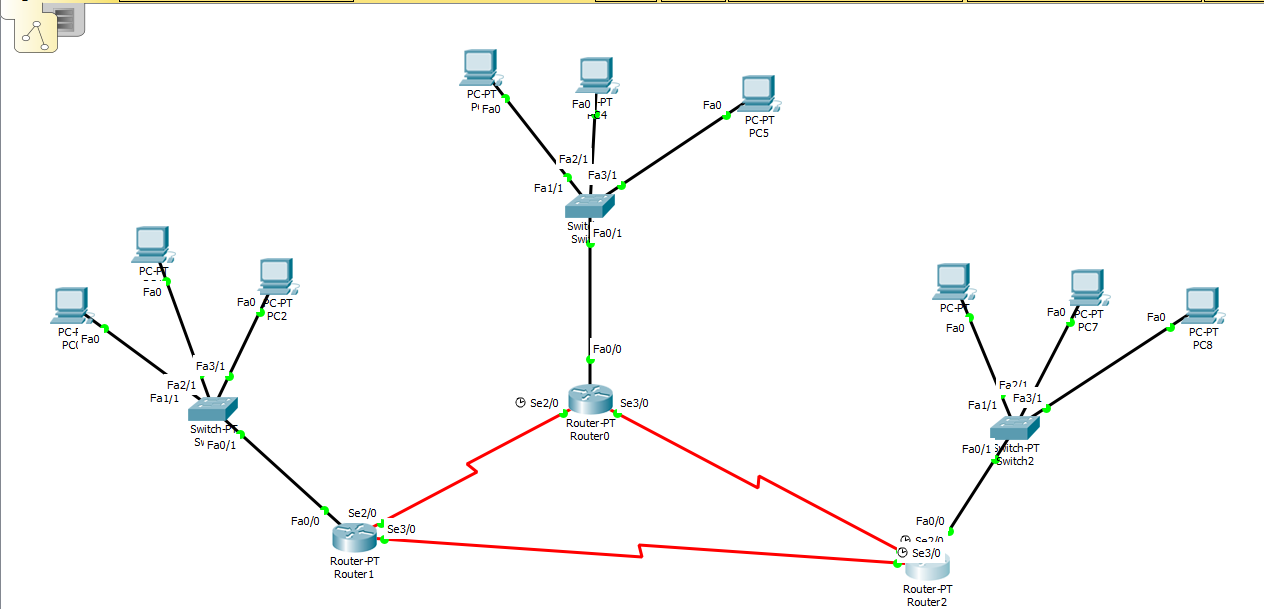
*Routing Information Protocol:*

**Routing Information Protocol** (RIP) is a dynamic routing protocol which uses hop count as a routing metric to find the best path between the source and the destination network. It is a distance vector routing protocol which has AD value 120 and works on the application layer of OSI model. RIP uses port number 520.

*Features of RIP :*

1. Updates of the network are exchanged periodically.
2. Updates (routing information) are always broadcast.
3. Full routing tables are sent in updates.
4. Routers always trust on routing information received from neighbor routers. This is also known as *Routing on rumours*.

**Topology –**



**Steps –**

*Step 1: Create Topology as shown above and assign IP to PC’s*

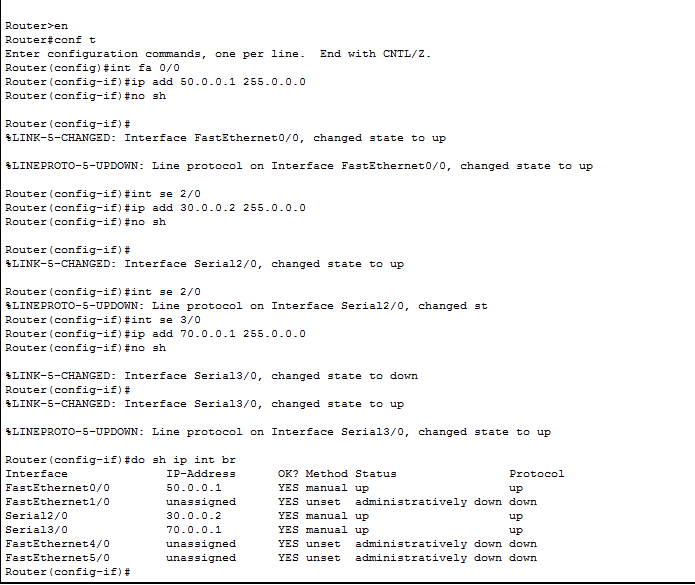
*Step 2: Start CLI for Router 1 and assign IP address for all devices connected to the router.*

**Syntax for adding IP address:**

**int link\_port**

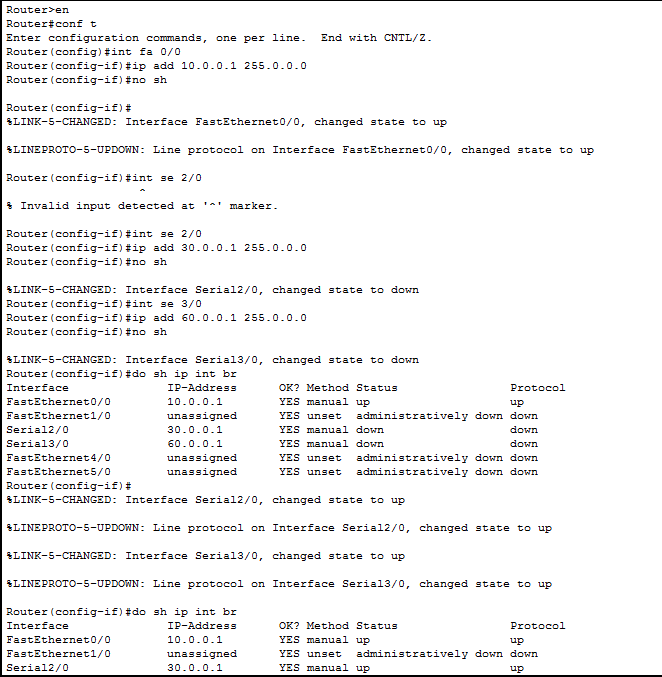
**ip add [ip\_address] [subnet\_mask]**

**no sh**

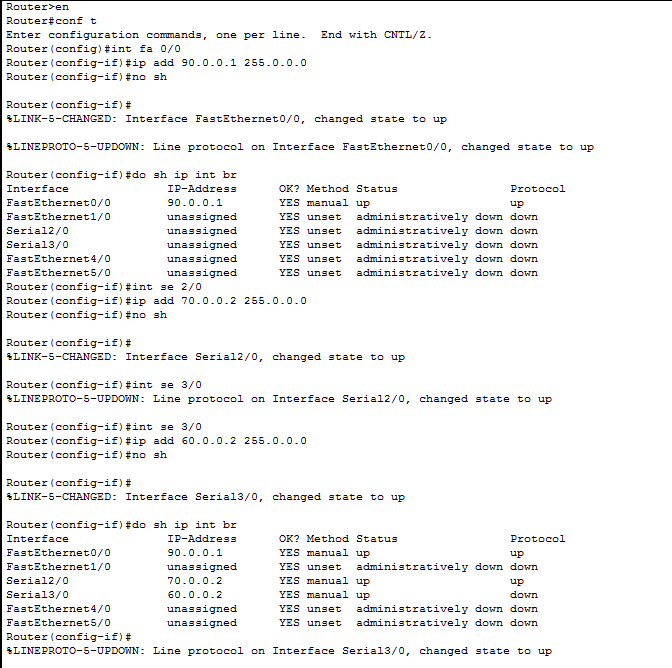


*Step 3: Repeat step 2 for router 2 and router 3*

**Router 2:**

****

**Router 3:**

****

*Step 4: After all IP address are set,configure the rip protocol on every router as follows*

**Syntax:**

Router rip

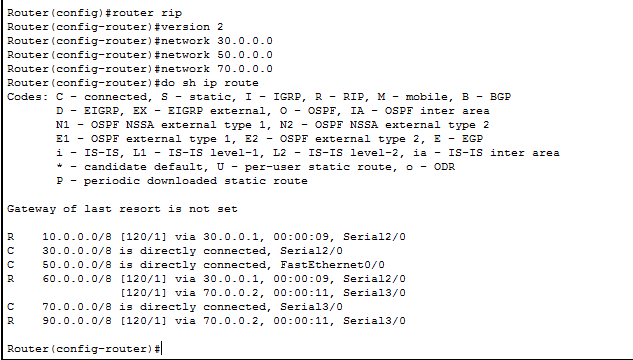
Version 2

network1 [network\_address]

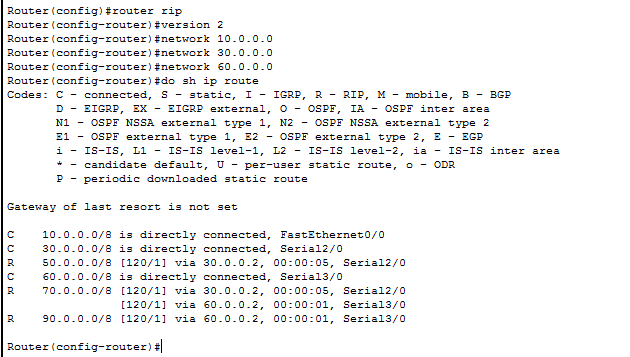
network2 [network\_address]

network3 [network\_address]

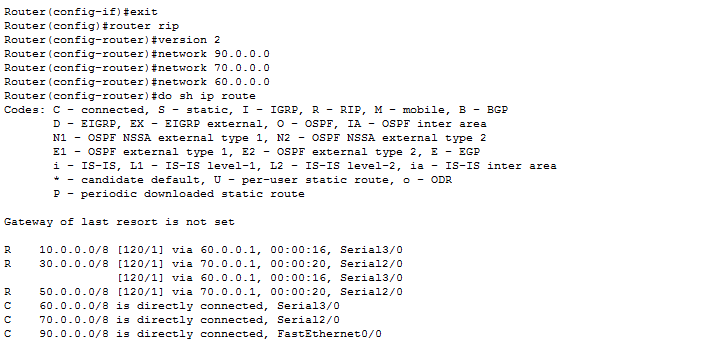
RIP configration for router 1:



RIP configration for router 2:



RIP configration for router 3:



**PRACTICAL 2**

**Open Shortest Path First (OSPF)**

**Aim –** Create a network with three routers with OSPF and each router associated network will have minimum three PC. Show connectivity.

**Theory –**

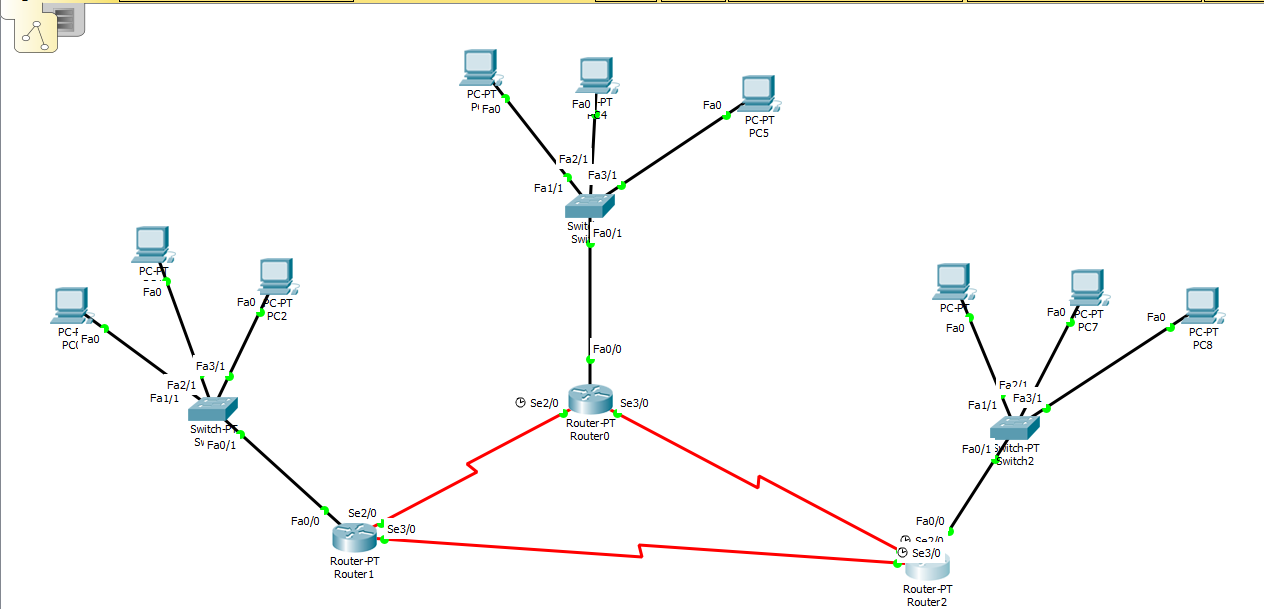
*Open Shortest Path First Protocol:*

Open Shortest Path First (OSPF) is a link-state routing protocol which is used to find the best path between the source and the destination router using its own Shortest Path First). OSPF is developed by Internet Engineering Task Force (IETF) as one of the Interior Gateway Protocol (IGP), i.e, the protocol which aims at moving the packet within a large autonomous system or routing domain. It is a network layer protocol which works on the protocol number 89 and uses AD value 110. OSPF uses multicast address 224.0.0.5 for normal communication and 224.0.0.6 for update to designated router(DR)/Backup Designated Router (BDR).

*OSPF supports/provides/advantages:*

* Both IPv4 and IPv6 routed protocols
* Load balancing with equal cost routes for same destination
* VLSM and route summarization
* Unlimited hop counts
* Trigger updates for fast convergence
* A loop free topology using SPF algorithm
* Run on most routers
* Classless protocol

**Topology –**



**Steps –**

*Step 1: Create Topology as shown above and assign the IP address to the PC.*

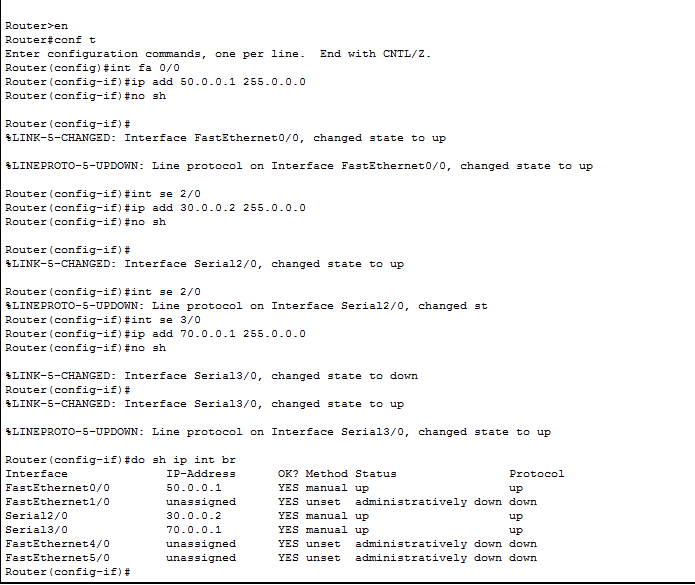
*Step 2: Start CLI for Router 1 and assign IP address for all devices connected to the router.*

**Syntax for adding IP address:**

**int link\_port**

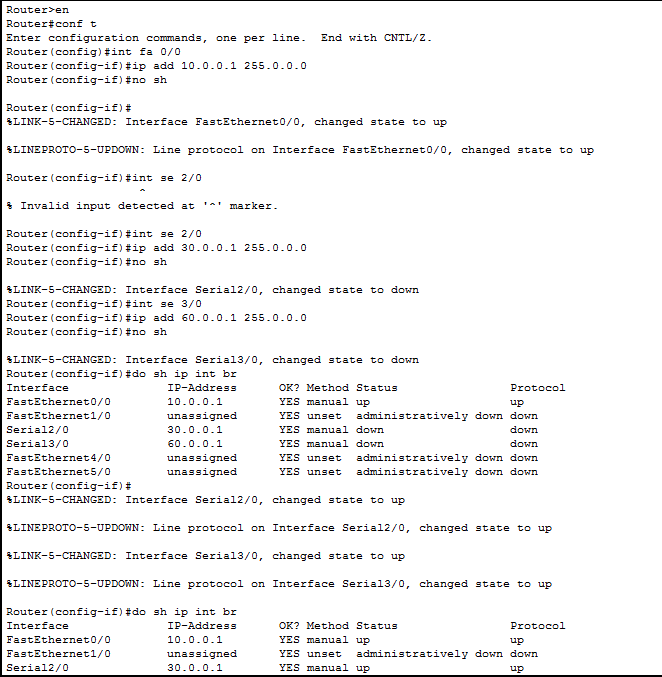
**ip add [ip\_address] [subnet\_mask]**

**no sh**

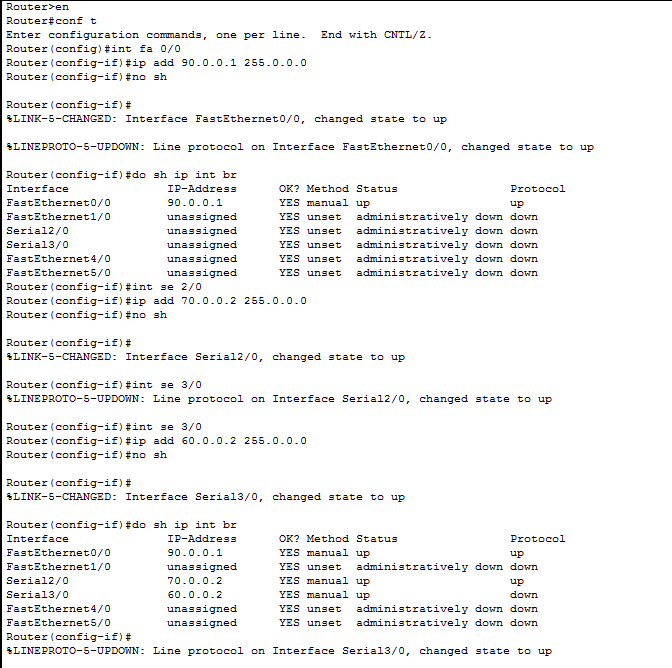


*Step 3: Repeat step 2 for router 2 and router 3*

**Router 2:**

****

**Router 3:**

****

*Step 4: After all IP address are set,configure the ospf protocol on every router as follows*

**Syntax:**

Router ospf 1

network1 [network\_address] [subnet\_mask] area 0

network2 [network\_address] [subnet\_mask] area 0

network3 [network\_address] [subnet\_mask] area 0

**OSPF configration for router 1:**

Router(config)#router ospf 1

Router(config-router)#network 10.0.0.0 0.255.255.255 area 0

Router(config-router)#network 30.0.0.0 0.255.255.255 area 0

Router(config-router)#network 60.0.0.0 0.255.255.255 area 0

Router(config-router)#exit

**OSPF configration for router 2:**

Router(config)#router ospf 1

Router(config-router)#network 30.0.0.0 0.255.255.255 area 0

Router(config-router)#network 50.0.0.0 0.255.255.255 area 0

Router(config-router)#network 70.0.0.0 0.255.255.255 area 0

Router(config-router)#exit

**OSPF configration for router 3:**

Router(config)#router ospf 1

Router(config-router)#network 70.0.0.0 0.255.255.255 area 0

Router(config-router)#network 90.0.0.0 0.255.255.255 area 0

Router(config-router)#network 60.0.0.0 0.255.255.255 area 0

Router(config-router)#exit

**PRACTICAL 3**

**Border Gateway Protocol (BGP)**

**Aim –** Create a network with three routers with BGP. Show connectivity.

**Theory –**

*Border Gateway Protocol:*

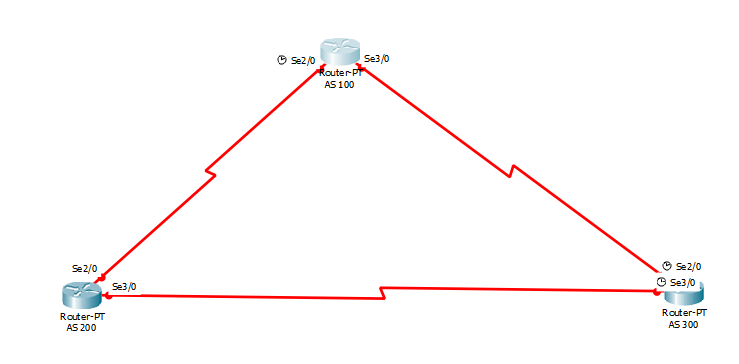
Border Gateway Protocol (BGP) is used to Exchange routing information for the internet and is the protocol used between ISP which are different ASes.

The protocol can connect together any internetwork of autonomous system using an arbitrary topology. The only requirement is that each AS have at least one router that is able to run BGP and that is router connect to at least one other AS’s BGP router. BGP’s main function is to exchange network reach-ability information with other BGP systems. Border Gateway Protocol constructs an autonomous systems’ graph based on the information exchanged between BGP routers.

*Characteristics of Border Gateway Protocol (BGP):*

* **Inter-Autonomous System Configuration:** The main role of BGP is to provide communication between two autonomous systems.
* BGP supports Next-Hop Paradigm.
* Coordination among multiple BGP speakers within the AS (Autonomous System).
* **Path Information:** BGP advertisement also include path information, along with the reachable destination and next destination pair.
* **Policy Support:** BGP can implement policies that can be configured by the administrator. For ex:- a router running BGP can be configured to distinguish between the routes that are known within the AS and that which are known from outside the AS.
* Runs Over TCP.
* BGP conserve network Bandwidth.
* BGP supports CIDR.
* BGP also supports Security.

**Topology –**



**Steps –**

*Step 1: Create Topology as shown above and assign the IP address to the PC.*

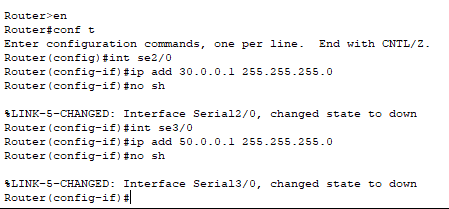
*Step 2: Start CLI for Router 1 and assign IP address for all devices connected to the router.*

**Syntax for adding IP address:**

**int link\_port**

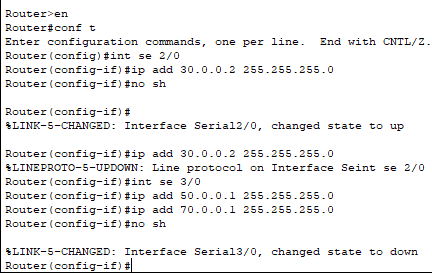
**ip add [ip\_address] [subnet\_mask]**

**no sh**

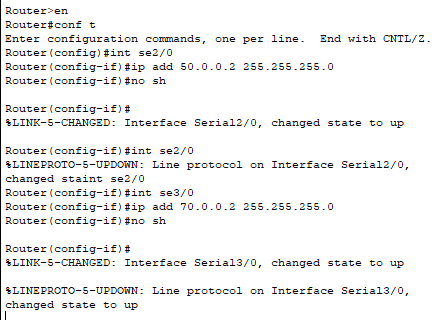
**

*Step 3: Repeat step 2 for router 2 and router 3*

**Router 2:**

****

**Router 3:**

****

*Step 4: After all IP address are set,configure the BGP protocol on every router as follows*

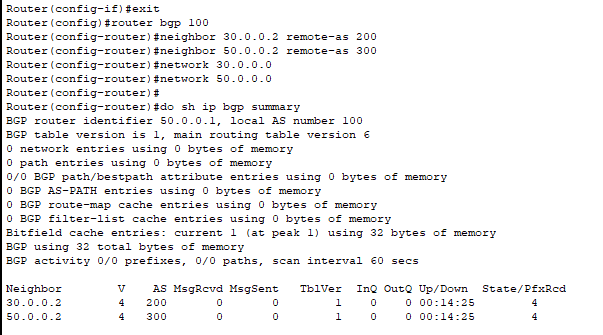
**Syntax:**

Router bgp bgp\_id

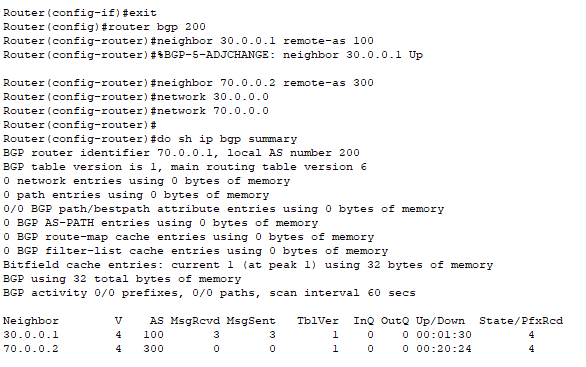
Neighbor1 [ip\_address] remote-as [neighbor\_id]

network1 [network\_address]

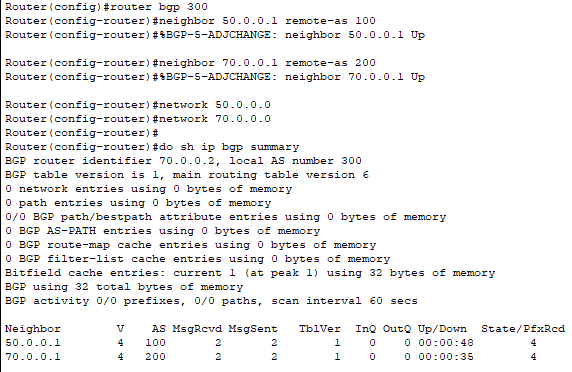
**BGP configration for router 1:**



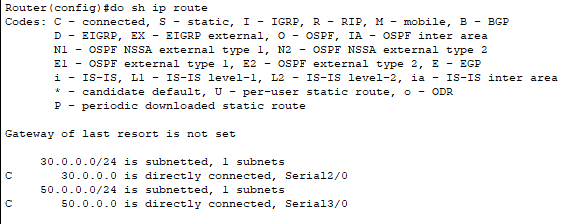
**BGP configration for router 2:**

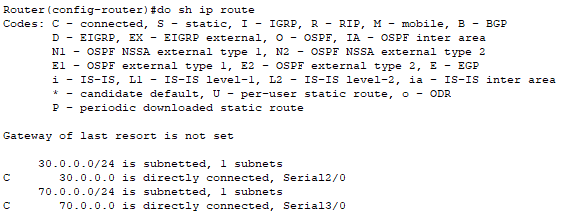


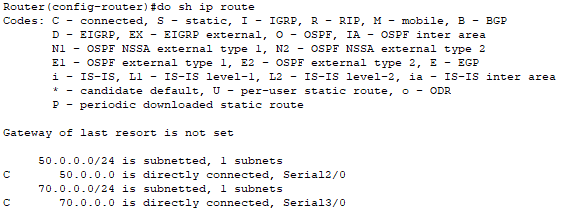
**BGP configration for router 3:**



**Outputs for routers 1,2,3 are as follows:**

****

****

****

**PRACTICAL 4**

**DHCP(Dynamic Host Configuration Protocol) and DNS()**

**Aim –** Create a network with a DHCP and DNS server.

**Theory –**

**Dynamic Host Configuration Protocol(DHCP)** is an application layer protocol which is used to provide:

1. Subnet Mask (Option 1 – e.g., 255.255.255.0)
2. Router Address (Option 3 – e.g., 192.168.1.1)
3. DNS Address (Option 6 – e.g., 8.8.8.8)
4. Vendor Class Identifier (Option 43 – e.g., ‘unifi’ = 192.168.1.9 ##where unifi = controller)

DHCP is based on a client-server model and based on discovery, offer, request, and ACK.

*DHCP Server:*

A**DHCP Server** is a network server that automatically provides and assigns IP addresses, default gateways and other network parameters to client devices. It relies on the standard protocol known as Dynamic Host Configuration Protocol or DHCP to respond to broadcast queries by clients.

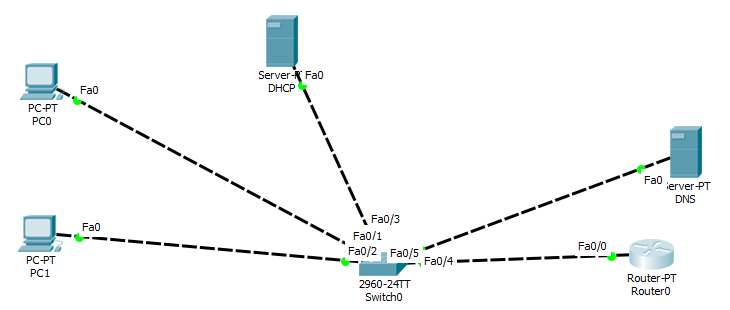
A DHCP server automatically sends the required network parameters for clients to properly communicate on the network. Without it, the network administrator has to manually set up every client that joins the network, which can be cumbersome, especially in large networks. DHCP servers usually assign each client with a unique dynamic IP address, which changes when the client’s lease for that IP address has expired.

*DNS :*

The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.

Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1.

**Topology –**



**Steps –**

*Step 1: Create Topology as shown above but do not assign the IP address to the PC.*

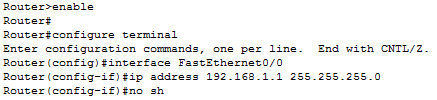
*Step 2: Start CLI for Router 1 and assign IP address for link connected to router.*

**Syntax for adding IP address:**

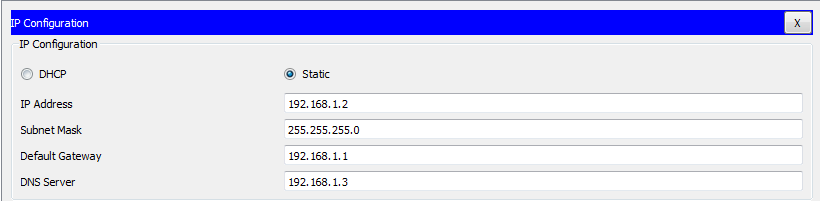
**int link\_port**

**ip add [ip\_address] [subnet\_mask]**

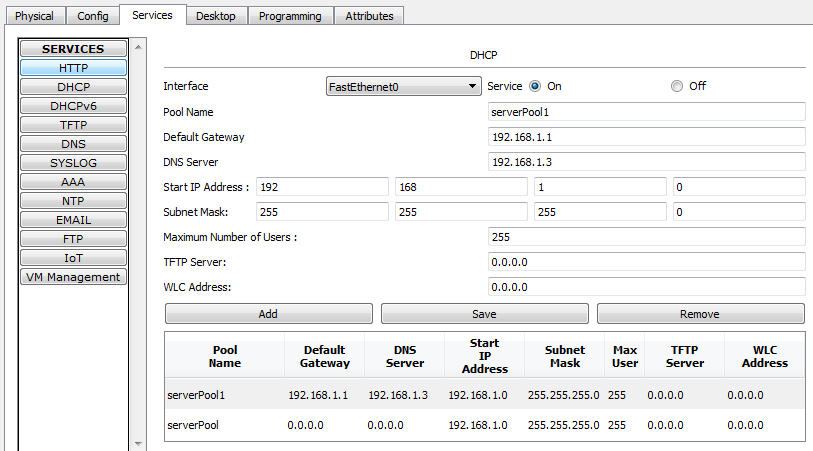
**no sh**

**

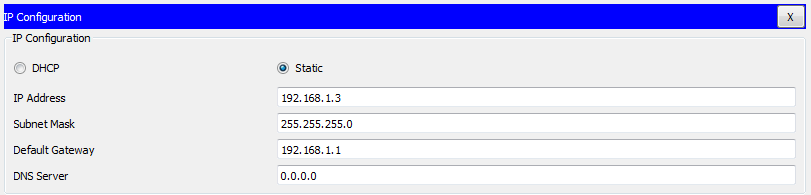
*Step 3: Now, go to DHCP server -> Desktop -> IP configuration*

**

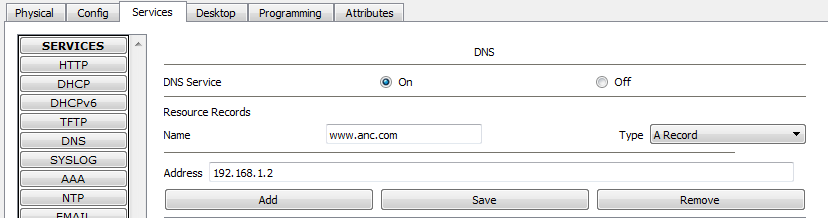
*Step 4: Now, In DHCP server go to services -> DHCP. Fill the fields like shown below and press save.*

****

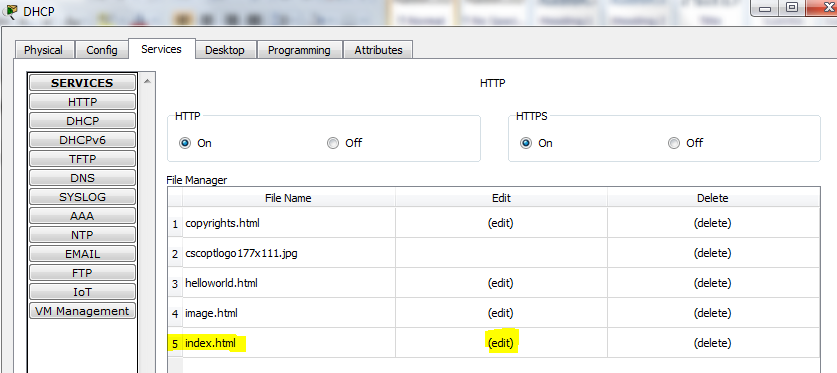
*Step 5: Now, go to DNS server -> Desktop -> IP configuration*

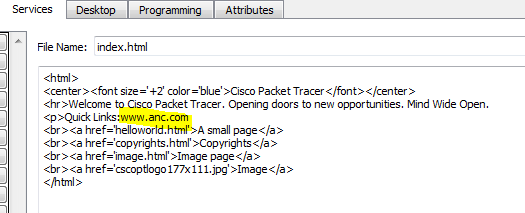
**

*Step 6: Now, In DNS server go to services -> DNS. Fill the fields like shown below and press add then save.*

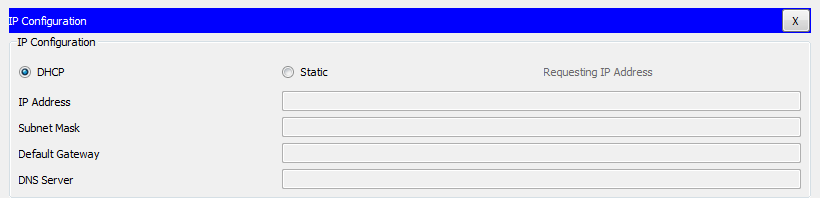
****

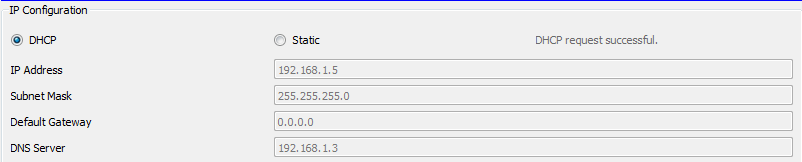
*Step 7: Now, Head back to DHCP server ->Services ->HTTP. Click edit in front of Index.html file and add a quick link www.anc.com over there. Save this file.*

**

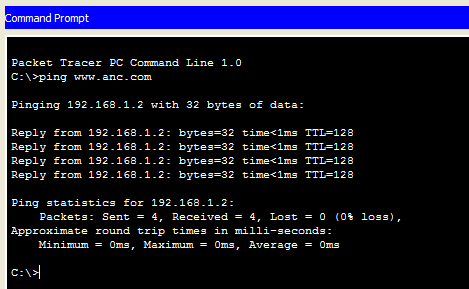
**

*Step 8: Now, go to the PC’s and request the IP address from server.*

****

****

*Step 9: Now, go to a PC-> Desktop -> command prompt and ping the DNS website www.anc.com .*

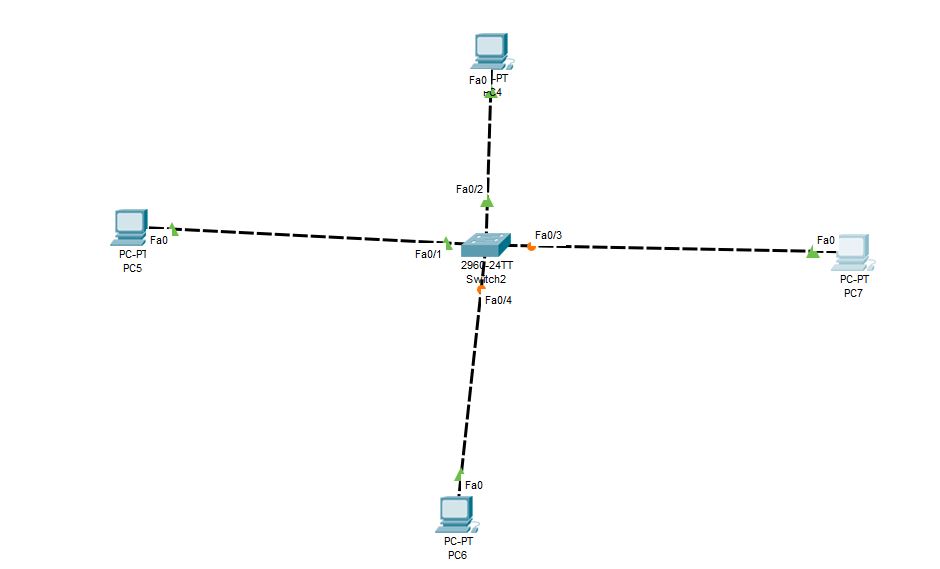
**

**PRACTICAL 5**

**Network Virtualization (Dividing one LAN into 2)**

**Aim –** Create a LAN network with 4 PC’s and 1 switch. Using network virtualization convert one physical LAN in tow virtual LAN’s.

**Topology –**



**Steps –**

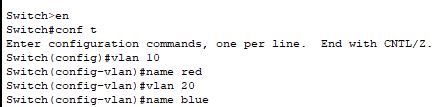
*Step 1: Create Topology as shown above but do not assign the IP address to the PC.*

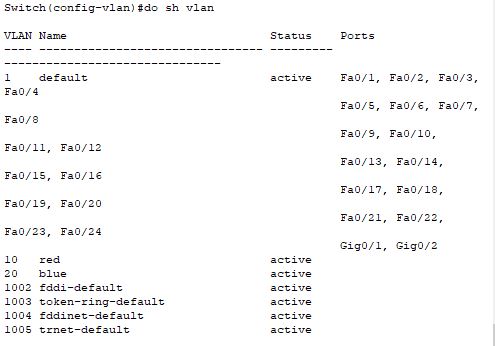
*Step 2: Start CLI for Switch and create tow vlans vlan red and vlan blue.*

**Syntax for creating VLAN:**

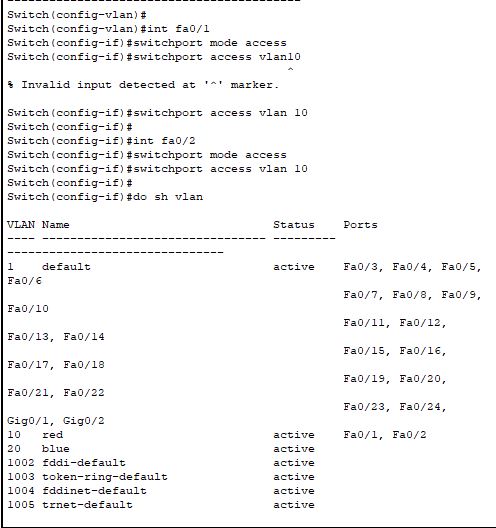
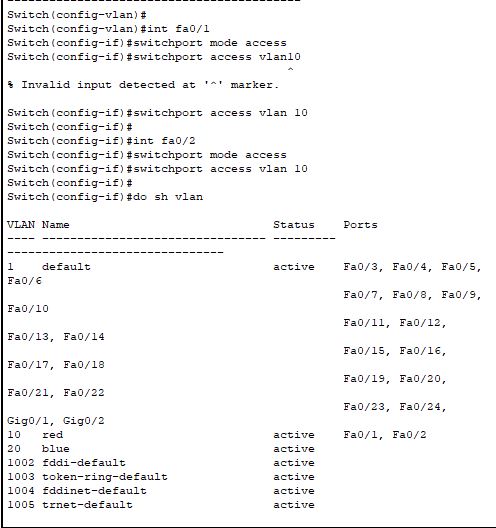
**vlan v\_number**

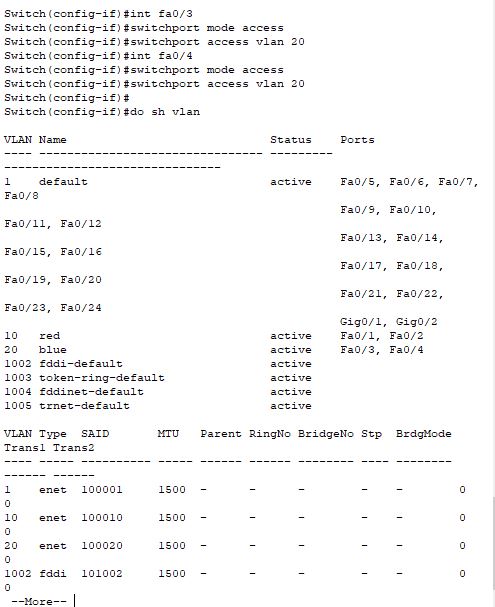
**name v\_name**

**

**

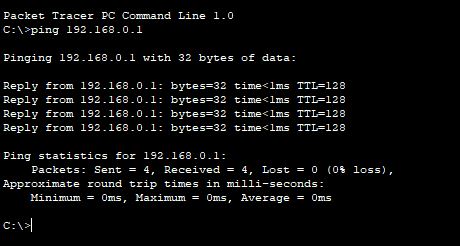
*Step 3: Now, start adding the PC’s to specific vlan*

**

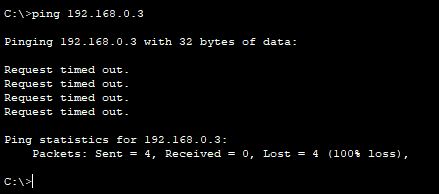
**

*Step 4: Now, When we will ping the PC’s from same VLAN they should give reply and for PC’s from different VLAN they won’t reply.*

**Same VLAN’s**

****

**Different VLAN’s**

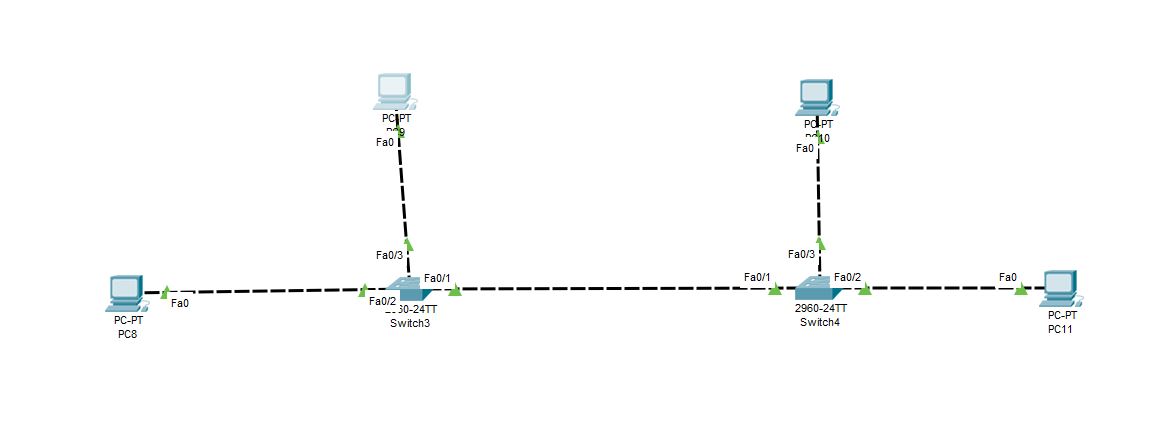
****

**PRACTICAL 6**

**Network Virtualization (Sending Data of 2 LAN’s from single data-link)**

**Aim –** Create a LAN network with 4 PC’s and 2 switches. Using network virtualization create two VLAN’s of two PC’s from one physical LAN’s.

**Topology –**



**Steps –**

*Step 1: Create Topology as shown above but do not assign the IP address to the PC.*

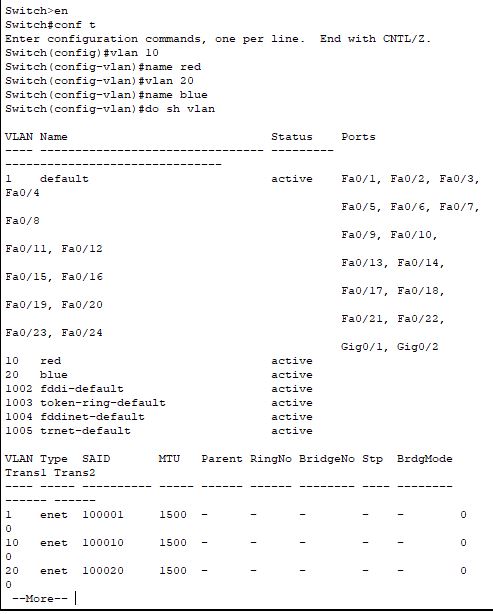
*Step 2: Start CLI for Switch and create tow vlans vlan red and vlan blue.*

**Syntax for creating VLAN:**

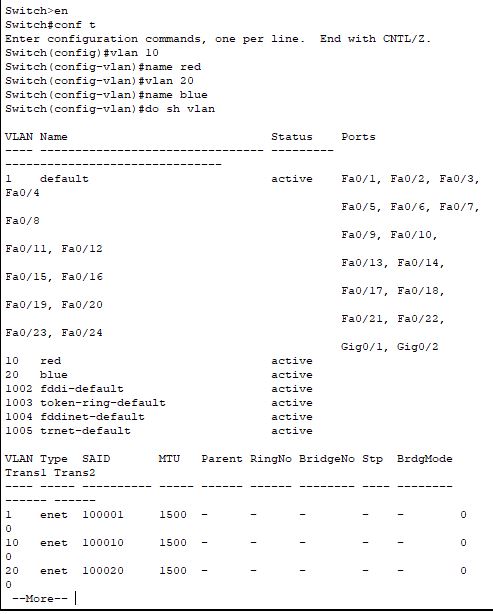
**vlan v\_number**

**name v\_name**

**Switch 1:**

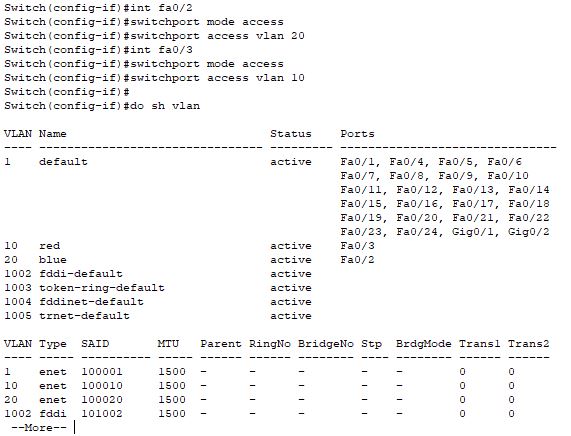
**

**Switch 2:**

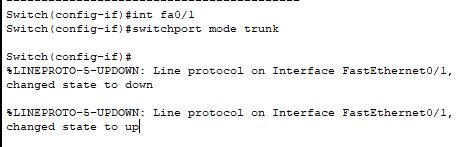
**

*Step 3: Now, start adding the PC’s to specific vlan*

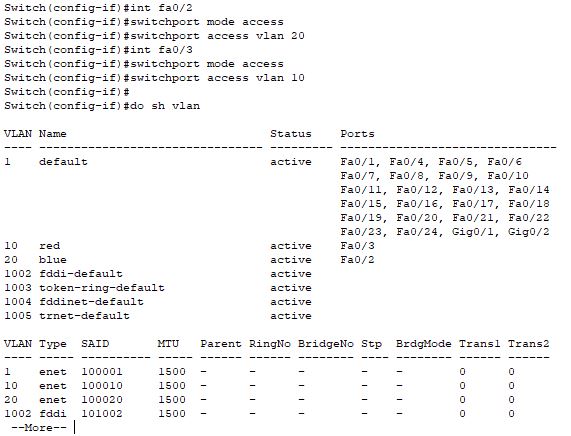
**Switch 1:**

****

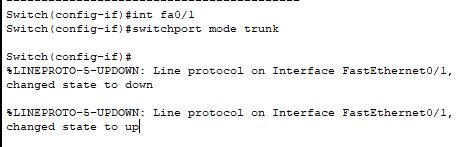
**To send the data of two VLAN’s from one link we use mode as *TRUNK***

**

**Switch 2:**

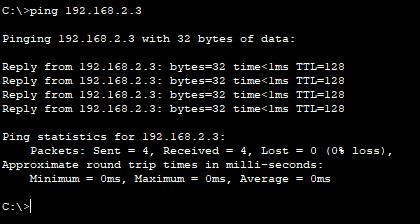
******

**To send the data of two VLAN’s from one link we use mode as *TRUNK***

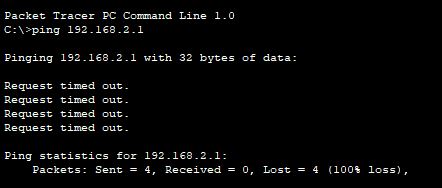
******

*Step 4: Now, When we will ping the PC’s from same VLAN they should give reply and for PC’s from different VLAN they won’t reply.*

**Same VLAN’s**

****

**Different VLAN’s**

****

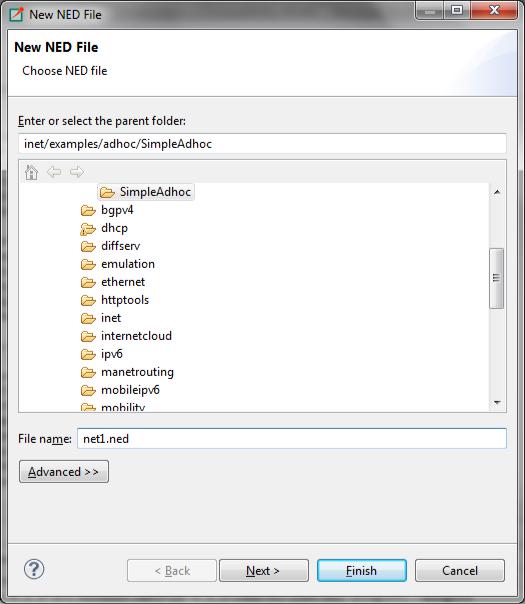
**PRACTICAL 7**

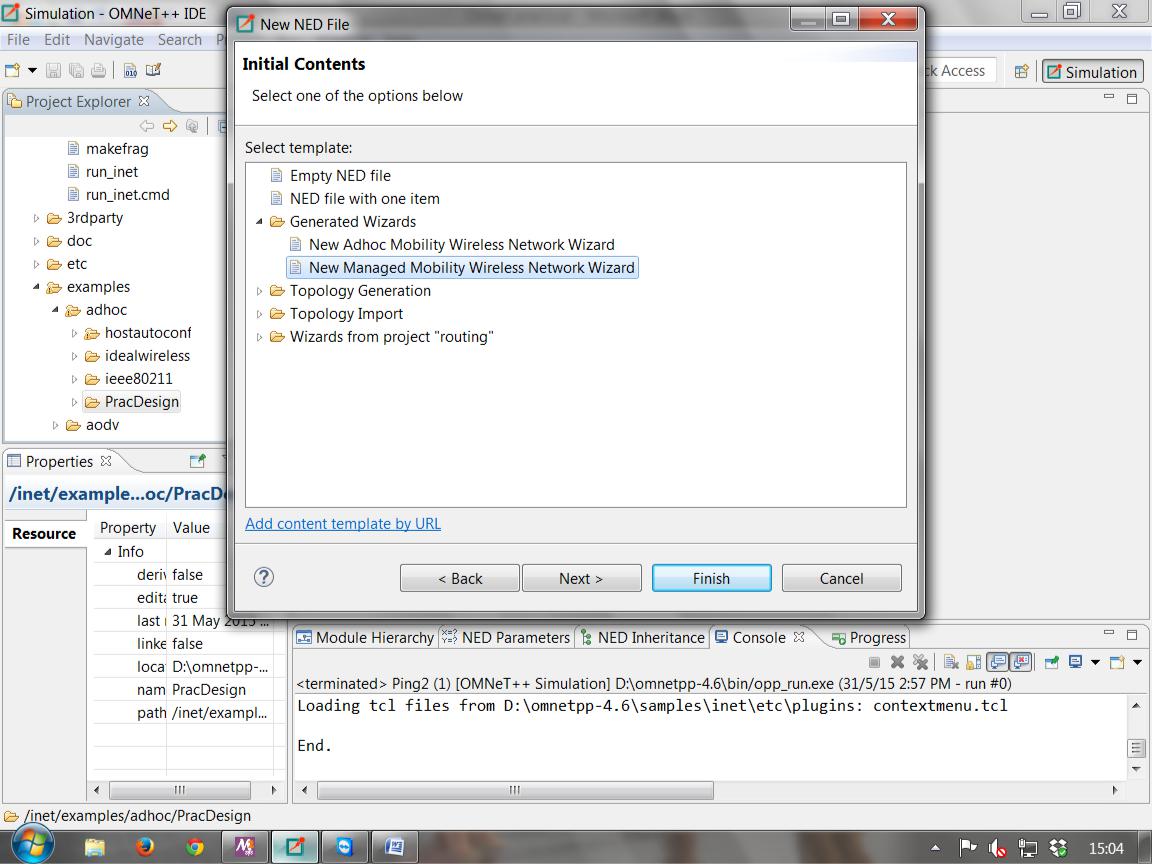
**ADHOC Network**

**Aim :** Create simple Adhoc network in OMNET++.

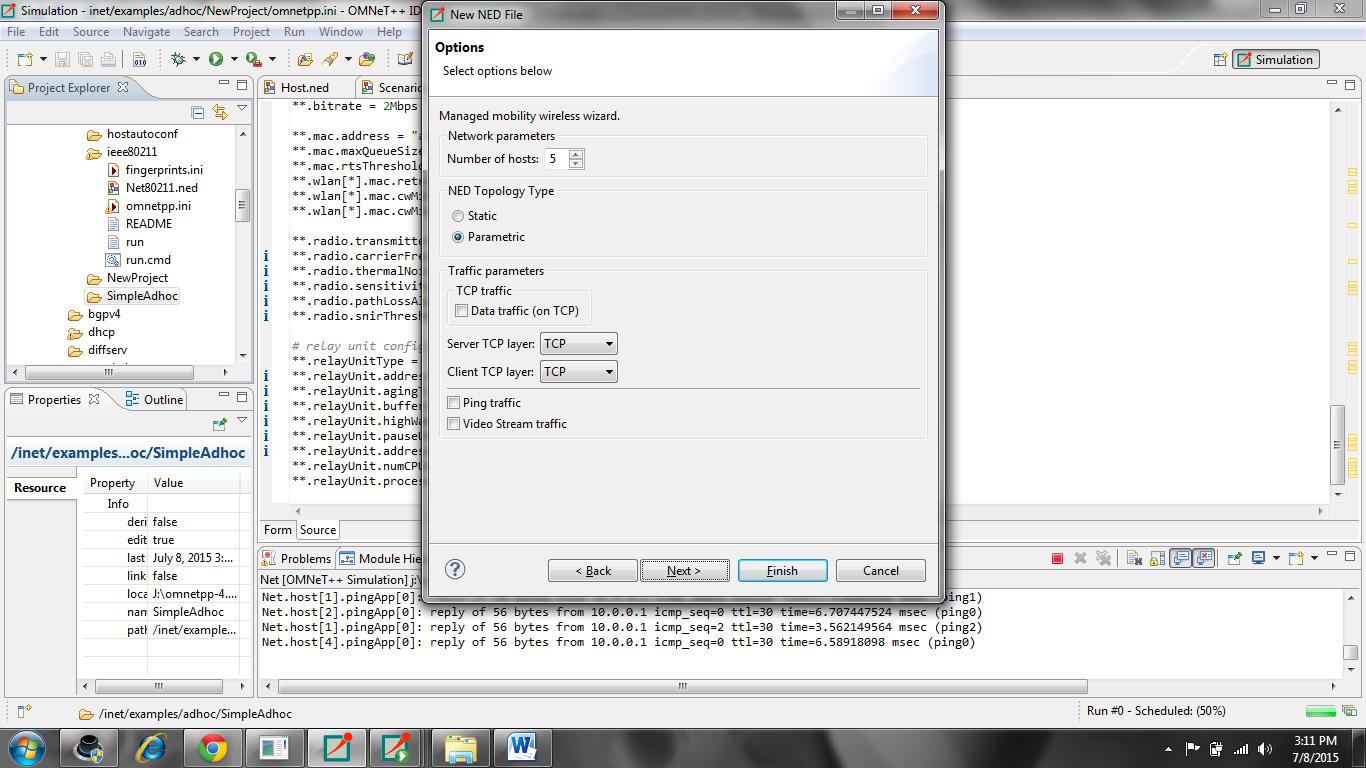
**Step for practical :**

1. open inet/examples/
2. Right click on adhoc -create new folder as SimpleAdhoc.
3. Right click on your newly created folder and select NED file. Give name as Net1.
4. Click on new manages mobility wireless network wizard.

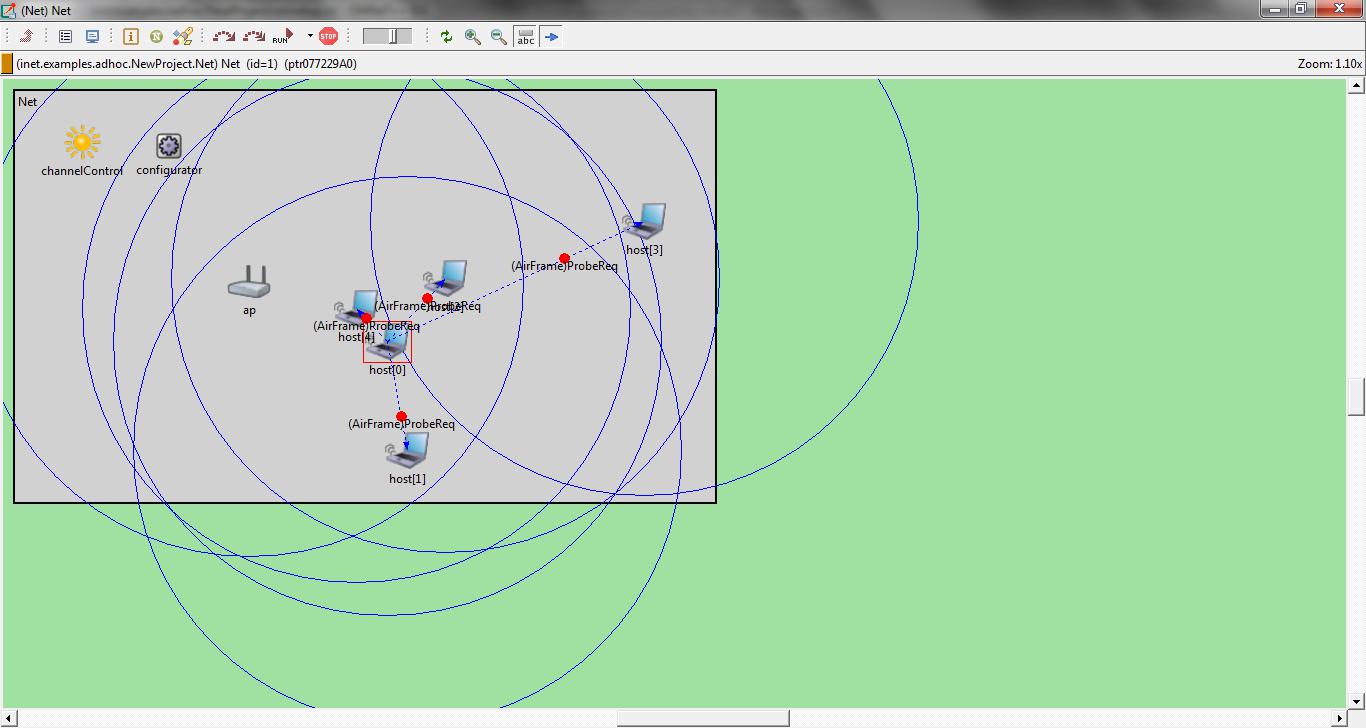




1. Then configure as follows:



1. Run the program you will get output as follows:



**Below the code that will be available in source part of .ned file.**

network = Net1

\*.numOfHosts = 5

#debug-on-errors = true

tkenv-plugin-path = ../../../etc/plugins

\*\*.constraintAreaMinX = 0m

\*\*.constraintAreaMinY = 0m

\*\*.constraintAreaMinZ = 0m

\*\*.constraintAreaMaxX = 600m

\*\*.constraintAreaMaxY = 400m

\*\*.constraintAreaMaxZ = 0m

\*\*.debug = true

\*\*.coreDebug = false

\*\*.host\*.\*\*.channelNumber = 0

# channel physical parameters

\*.channelControl.carrierFrequency = 2.4GHz \*.channelControl.pMax = 2.0mW \*.channelControl.sat = -110dBm \*.channelControl.alpha = 2

# mobility

\*\*.host\*.mobilityType = "MassMobility"

\*\*.host\*.mobility.initFromDisplayString = false

\*\*.host\*.mobility.changeInterval = truncnormal(2s, 0.5s) \*\*.host\*.mobility.changeAngleBy = normal(0deg, 30deg) \*\*.host\*.mobility.speed = truncnormal(20mps, 8mps) \*\*.host\*.mobility.updateInterval = 100ms

# ping app (host[0] pinged by others) \*.host[0].numPingApps = 0 \*.host[\*].numPingApps = 2 \*.host[\*].pingApp[\*].destAddr = "host[0]" \*\*.pingApp[0].startTime = uniform(1s,5s) \*\*.pingApp[1].startTime = 5s+uniform(1s,5s) \*\*.pingApp[\*].printPing = true

# nic settings

\*\*.wlan[\*].bitrate = 2Mbps

\*\*.wlan[\*].mgmt.frameCapacity = 10

\*\*.wlan[\*].mac.address = "auto"

\*\*.wlan[\*].mac.maxQueueSize = 14

\*\*.wlan[\*].mac.rtsThresholdBytes = 3000B

\*\*.wlan[\*].mac.retryLimit = 7

\*\*.wlan[\*].mac.cwMinData = 7

\*\*.wlan[\*].radio.transmitterPower = 2mW

\*\*.wlan[\*].radio.thermalNoise = -110dBm

\*\*.wlan[\*].radio.sensitivity = -85dBm

\*\*.wlan[\*].radio.pathLossAlpha = 2

\*\*.wlan[\*].radio.snirThreshold = 4d

[Config Ping1]

description = "host1 pinging host0"

[Config Ping2] # \_\_interactive\_\_

description = "n hosts"

# leave numHosts undefined here

\*\*.mobility.constraintAreaMinZ = 0m

\*\*.mobility.constraintAreaMaxZ = 0m

\*\*.mobility.constraintAreaMinX = 0m

\*\*.mobility.constraintAreaMinY = 0m

\*\*.mobility.constraintAreaMaxX = 600m

\*\*.mobility.constraintAreaMaxY = 400m

\*\*.debug = false

\*\*.channelNumber = 0

# channel physical parameters

\*.channelControl.carrierFrequency = 2.4GHz \*.channelControl.pMax = 20.0mW \*.channelControl.sat = -110dBm \*.channelControl.alpha = 2

# mobility

\*\*.host[\*].mobilityType = "MassMobility"

\*\*.host[\*].mobility.changeInterval = truncnormal(2s, 0.5s) \*\*.host[\*].mobility.changeAngleBy = normal(0deg, 30deg) \*\*.host[\*].mobility.speed = truncnormal(20mps, 8mps) \*\*.host[\*].mobility.updateInterval = 100ms

# nic settings

\*\*.bitrate = 2Mbps

\*\*.mac.address = "auto"

\*\*.mac.maxQueueSize = 14

\*\*.mac.rtsThresholdBytes = 3000B

\*\*.wlan[\*].mac.retryLimit = 7

\*\*.wlan[\*].mac.cwMinData = 7

\*\*.wlan[\*].mac.cwMinMulticast = 31

\*\*.radio.transmitterPower = 20.0mW

\*\*.radio.carrierFrequency = 2.4GHz

\*\*.radio.thermalNoise = -110dBm

\*\*.radio.sensitivity = -85dBm

\*\*.radio.pathLossAlpha = 2

\*\*.radio.snirThreshold = 4dB

# relay unit configuration

\*\*.relayUnitType = "MACRelayUnitNP"

\*\*.relayUnit.addressTableSize = 100

\*\*.relayUnit.agingTime = 120s

\*\*.relayUnit.bufferSize = 1MiB

\*\*.relayUnit.highWatermark = 512KiB

\*\*.relayUnit.pauseUnits = 300 # pause for 300\*512 bit (19200 byte) time

\*\*.relayUnit.addressTableFile = ""

\*\*.relayUnit.numCPUs = 2

\*\*.relayUnit.processingTime = 2us

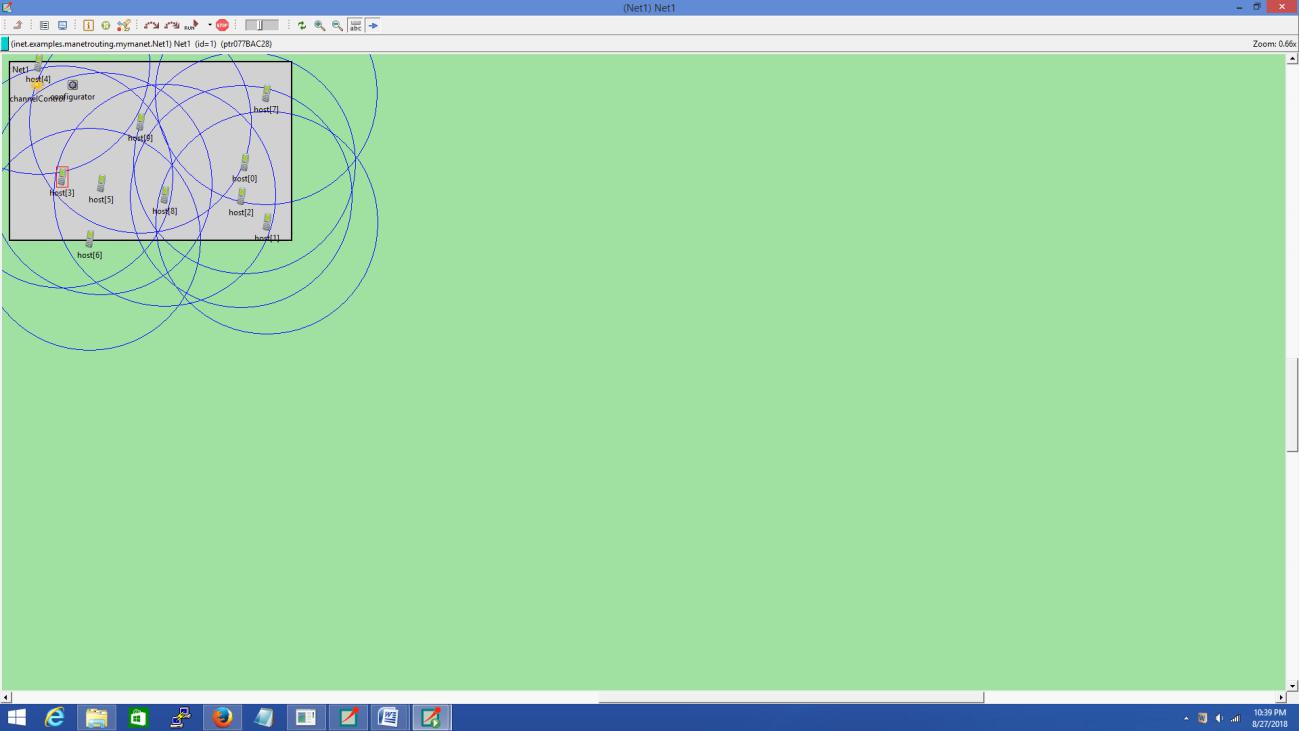
**PRACTICAL 8**

**MANET Network**

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

**Steps for practical:**

1. Then open inet/examples/
2. Right click on manetrouting -create new folder as MobileNet.
3. Right click on your newly created folder and select NED file. Give name as Net1. select new adhoc mobility wireless network wizard
4. right click on .ned file and run it as omnetpp simulation



Below the code that will be available in source part of .ned file

**package** inet.examples.manetrouting.mymanet;

**import** inet.networklayer.autorouting.ipv4.IPv4NetworkConfigurator;

**import** inet.nodes.inet.AdhocHost;

**import** inet.world.radio.ChannelControl;

**network** Net1

{

**parameters**:

**int** numHosts;

**submodules**:

host[numHosts]: AdhocHost

{

**parameters**:

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

}

channelControl: ChannelControl

{

**parameters**:

**@display**("p=60,50");

}

configurator: IPv4NetworkConfigurator

{

**@display**("p=140,50");

}

}

**file omnetpp.ini will be created with the following code**

[General]

network = Net1

*#record-eventlog = true*

*#eventlog-message-detail-pattern = \*:(not declaredOn(cMessage) and not*

*declaredOn(cNamedObject) and not declaredOn(cObject))*

\*.numHosts = 10

num-rngs = 3

\*\*.mobility.rng-0 = 1

\*\*.wlan[\*].mac.rng-0 = 2

*#debug-on-errors = true*

tkenv-plugin-path = ../../../etc/plugins

\*\*.channelNumber = 0

*# channel physical parameters*

\*.channelControl.carrierFrequency = 2.4GHz

\*.channelControl.pMax = 2.0mW

\*.channelControl.sat = -110dBm

\*.channelControl.alpha = 2

\*.channelControl.numChannels = 1

*# mobility*

\*\*.host[\*].mobilityType = "MassMobility"

\*\*.mobility.constraintAreaMinZ = 0m

\*\*.mobility.constraintAreaMaxZ = 0m

\*\*.mobility.constraintAreaMinX = 0m

\*\*.mobility.constraintAreaMinY = 0m

\*\*.mobility.constraintAreaMaxX = 600m

\*\*.mobility.constraintAreaMaxY = 400m

\*\*.mobility.changeInterval = truncnormal(2s, 0.5s)

\*\*.mobility.changeAngleBy = normal(0deg, 30deg)

\*\*.mobility.speed = truncnormal(20mps, 8mps)

\*\*.mobility.updateInterval = 100ms

* *ping app (host[0] pinged by others)* \*.host[0].pingApp[0].destAddr = "" \*.host[\*].numPingApps = 1 \*.host[\*].pingApp[0].destAddr = "host[0]" \*.host[\*].pingApp[0].startTime = uniform(1s,5s) \*.host[\*].pingApp[0].printPing = **true**
* *nic settings*

\*\*.wlan[\*].bitrate = 2Mbps

\*\*.wlan[\*].mgmt.frameCapacity = 10

\*\*.wlan[\*].mac.address = "auto"

\*\*.wlan[\*].mac.maxQueueSize = 14

\*\*.wlan[\*].mac.rtsThresholdBytes = 3000B

\*\*.wlan[\*].mac.retryLimit = 7

\*\*.wlan[\*].mac.cwMinData = 7

\*\*.wlan[\*].mac.cwMinMulticast = 31

\*\*.wlan[\*].radio.transmitterPower = 2mW \*\*.wlan[\*].radio.thermalNoise = -110dBm \*\*.wlan[\*].radio.sensitivity = -85dBm \*\*.wlan[\*].radio.pathLossAlpha = 2 \*\*.wlan[\*].radio.snirThreshold = 4

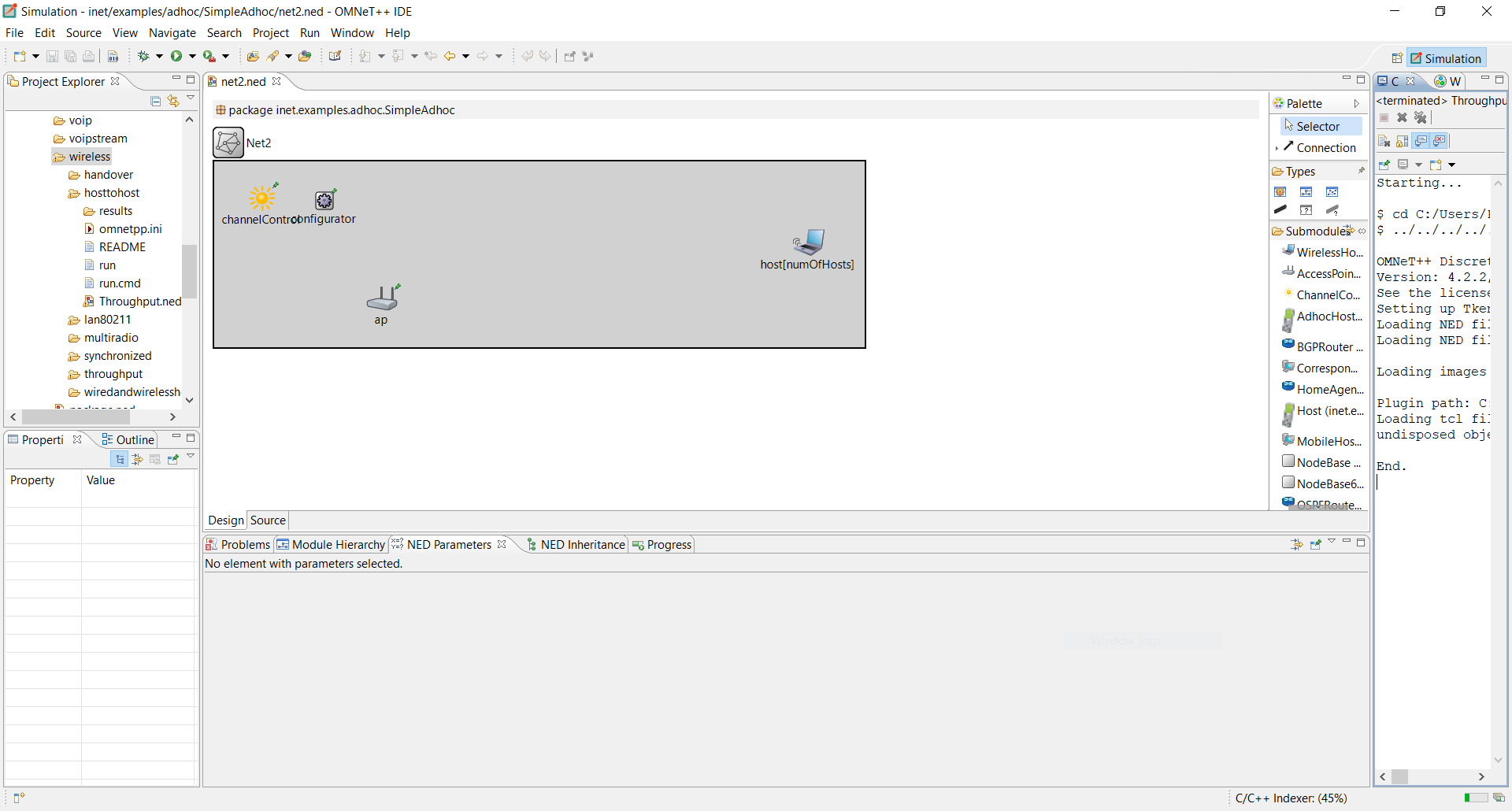
**PRACTICAL 9**

**Wireless Network**

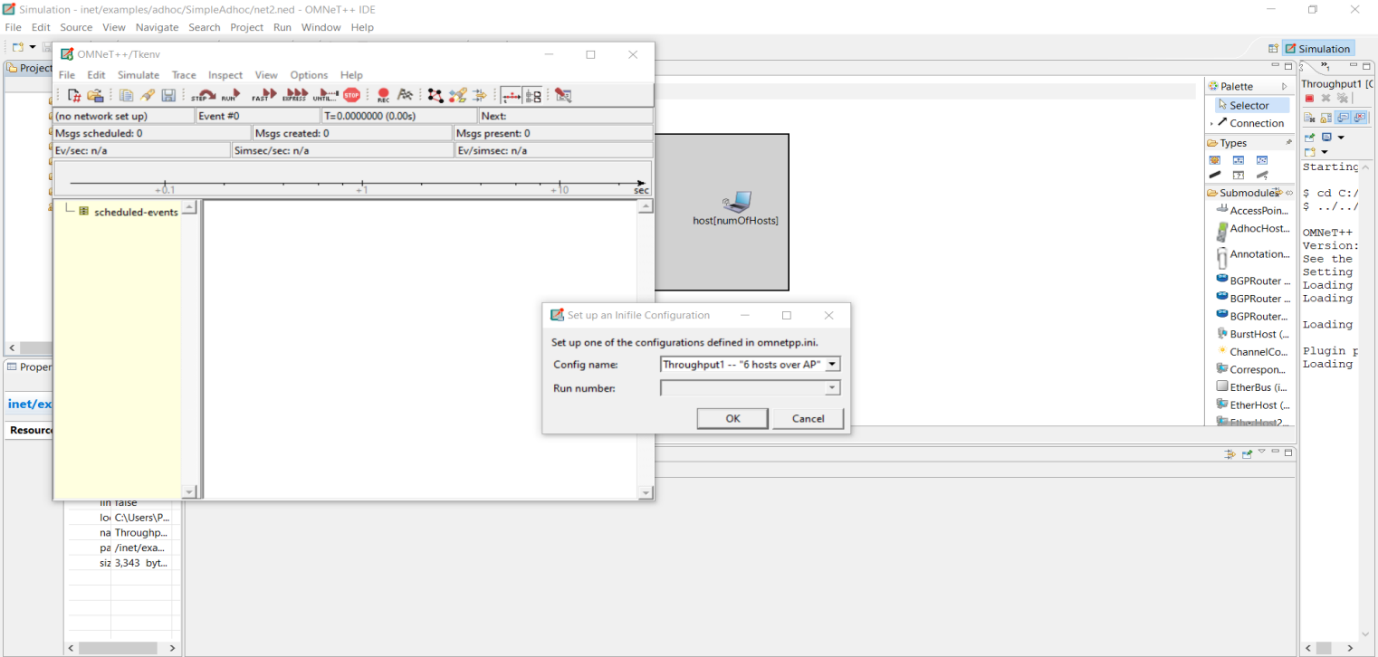
**Aim :** Create wireless Network in OMNET++.

**Step for practical :**

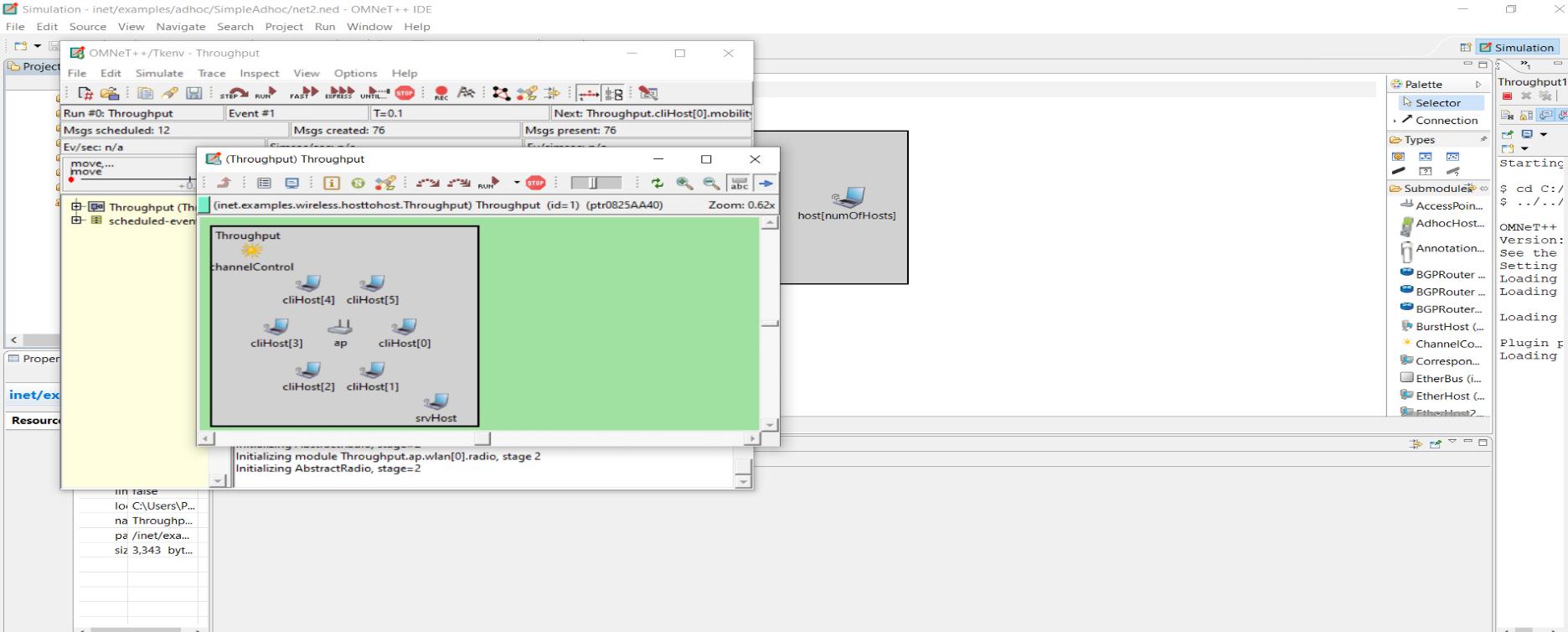
1.open inet/examples/wireless/hosttohost folder.



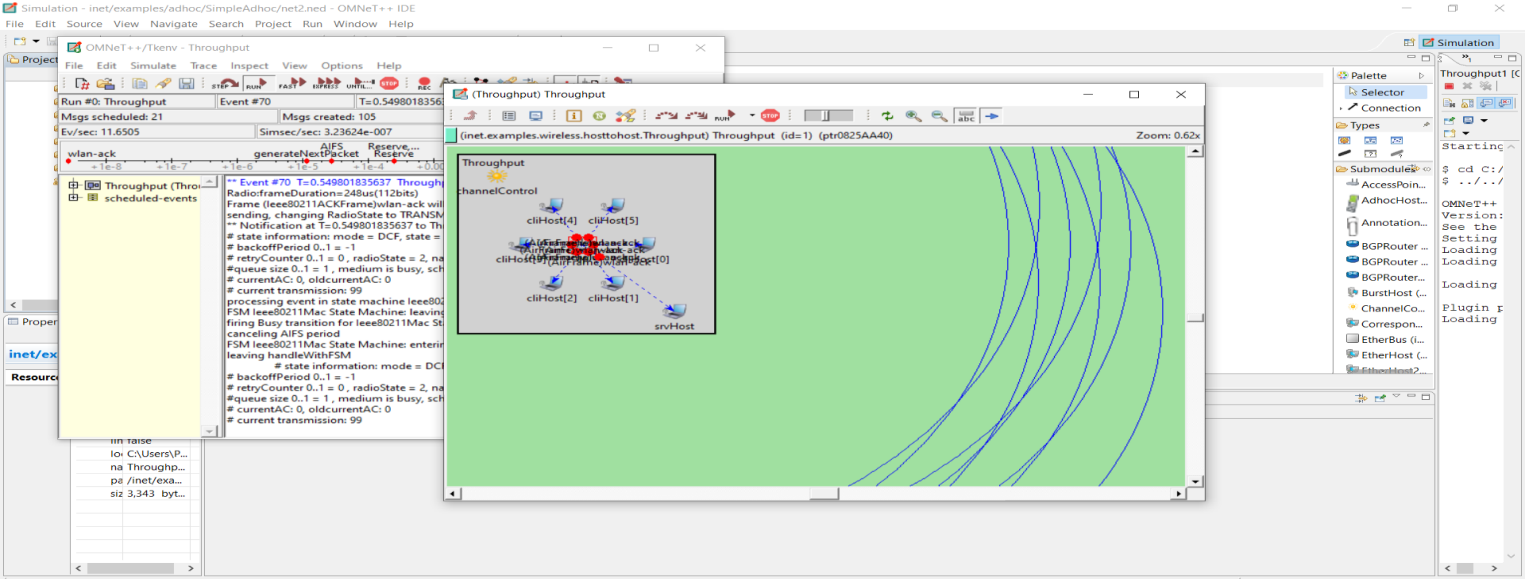
2.and run file Throughput.ned file with config name Throughput1 -- "6 hosts over AP"

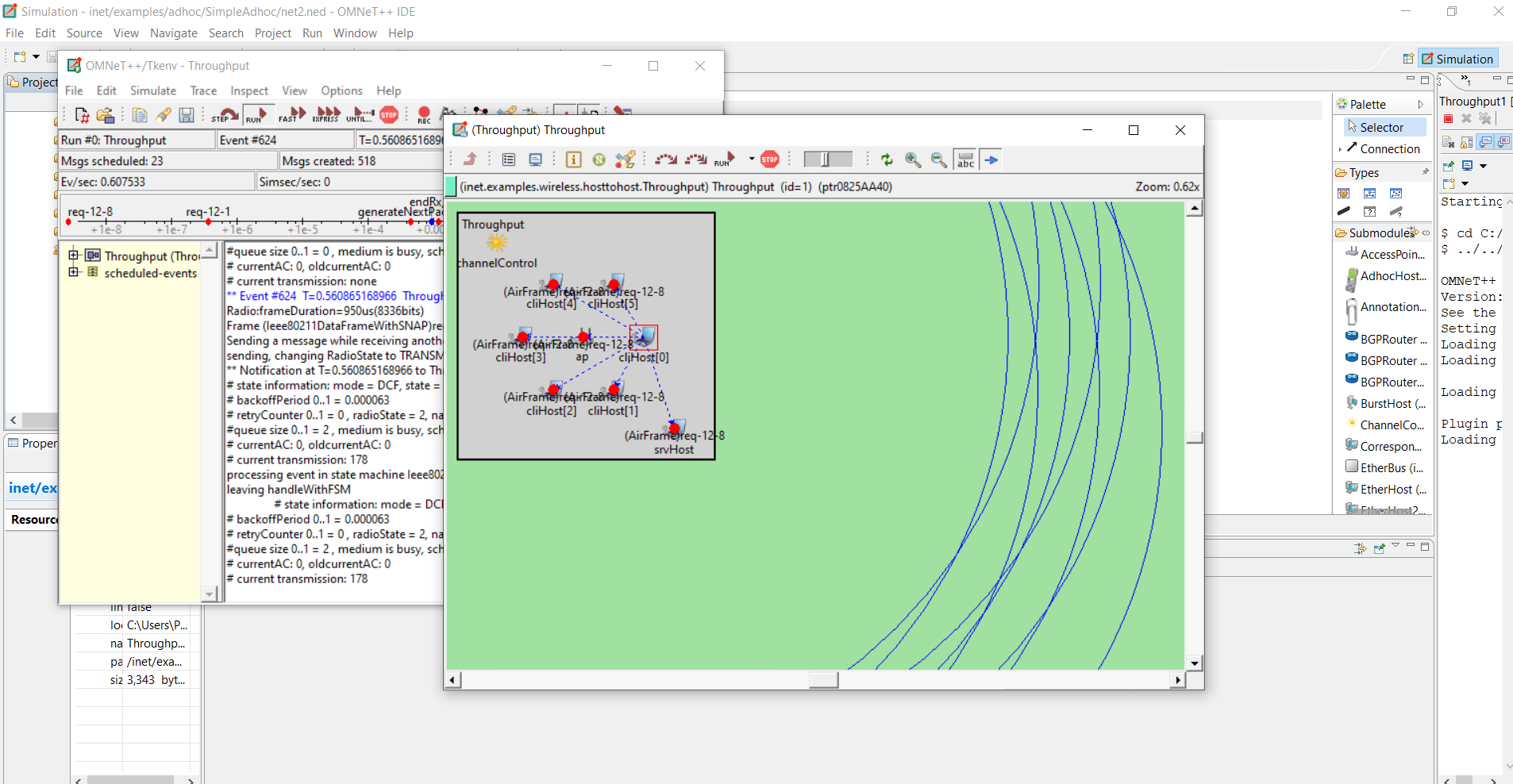


3.then click on ok.



4.then run the simulation.





Below the code that will be available in source part of Throughput.ned file

**package** inet.examples.wireless.hosttohost;

**import** inet.world.radio.ChannelControl;

**import** inet.nodes.wireless.AccessPoint;

**import** inet.mobility.models.StationaryMobility;

**import** inet.mobility.models.CircleMobility;

**import** inet.linklayer.ieee80211.Ieee80211Nic;

**import** inet.base.Sink;

**import** inet.base.NotificationBoard;

**import** inet.applications.ethernet.EtherAppCli;

**module** ThroughputClient

{

**parameters**:

**@display**("i=device/wifilaptop");

**@node**();

**gates**:

**input** radioIn **@directIn**;

**submodules**:

notificationBoard: NotificationBoard {

**parameters**:

**@display**("p=52,70");

}

cli: EtherAppCli {

**parameters**:

registerSAP = false;

**@display**("b=40,24;p=180,60,col");

}

wlan: Ieee80211Nic {

**parameters**:

**@display**("p=112,134;q=queue");

mgmtType = "Ieee80211MgmtSTASimplified";

}

mobility: CircleMobility {

**parameters**:

**@display**("p=50,141");

}

**connections** **allowunconnected**:

wlan.radioIn **<--** radioIn;

cli.out **-->** wlan.upperLayerIn;

}

**module** ThroughputServer

{

**parameters**:

**@display**("i=device/wifilaptop");

**@node**();

**gates**:

**input** radioIn **@directIn**;

**submodules**:

notificationBoard: NotificationBoard {

**parameters**:

**@display**("p=60,70");

}

sink: Sink {

**parameters**:

**@display**("p=210,68,col");

}

wlan: Ieee80211Nic {

**parameters**:

**@display**("p=120,158;q=queue");

mgmtType = "Ieee80211MgmtSTASimplified";

}

mobility: StationaryMobility {

**parameters**:

**@display**("p=50,141");

}

**connections** **allowunconnected**:

wlan.radioIn **<--** radioIn;

sink.in++ **<--** wlan.upperLayerOut;

}

**network** Throughput

{

**parameters**:

**int** numCli;

**@display**("b=297,203");

**submodules**:

cliHost[numCli]: ThroughputClient {

**parameters**:

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

}

srvHost: ThroughputServer {

**parameters**:

**@display**("p=350,350;r=,,#707070");

}

ap: AccessPoint {

**parameters**:

**@display**("p=200,200;r=,,#707070");

wlan[\*].mgmtType = "Ieee80211MgmtAPSimplified";

}

channelControl: ChannelControl {

**parameters**:

**@display**("p=61,46");

}

}

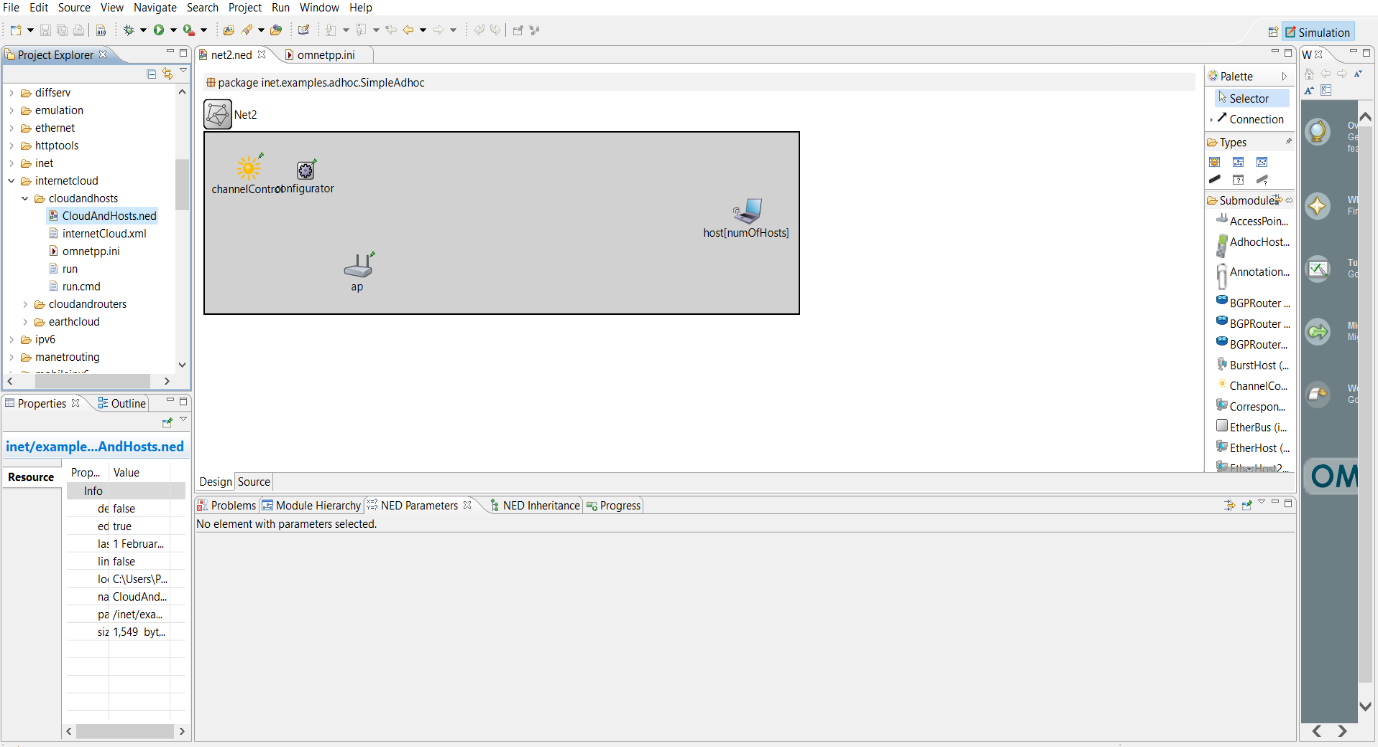
**PRACTICAL 10**

**Internet Cloud**

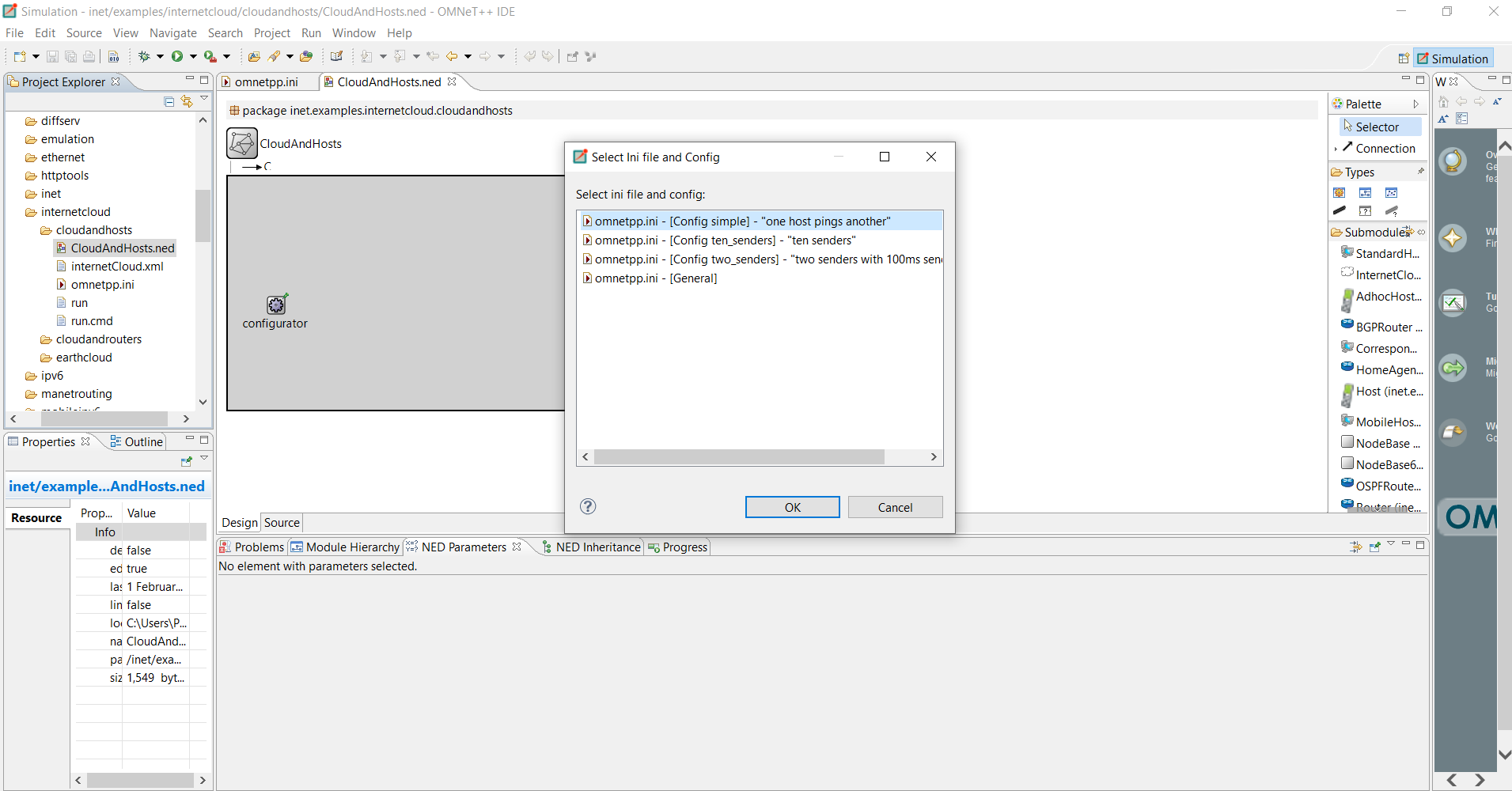
**Aim :** Create Network cloud and hosts using OMNET++.

**Step for practical :**

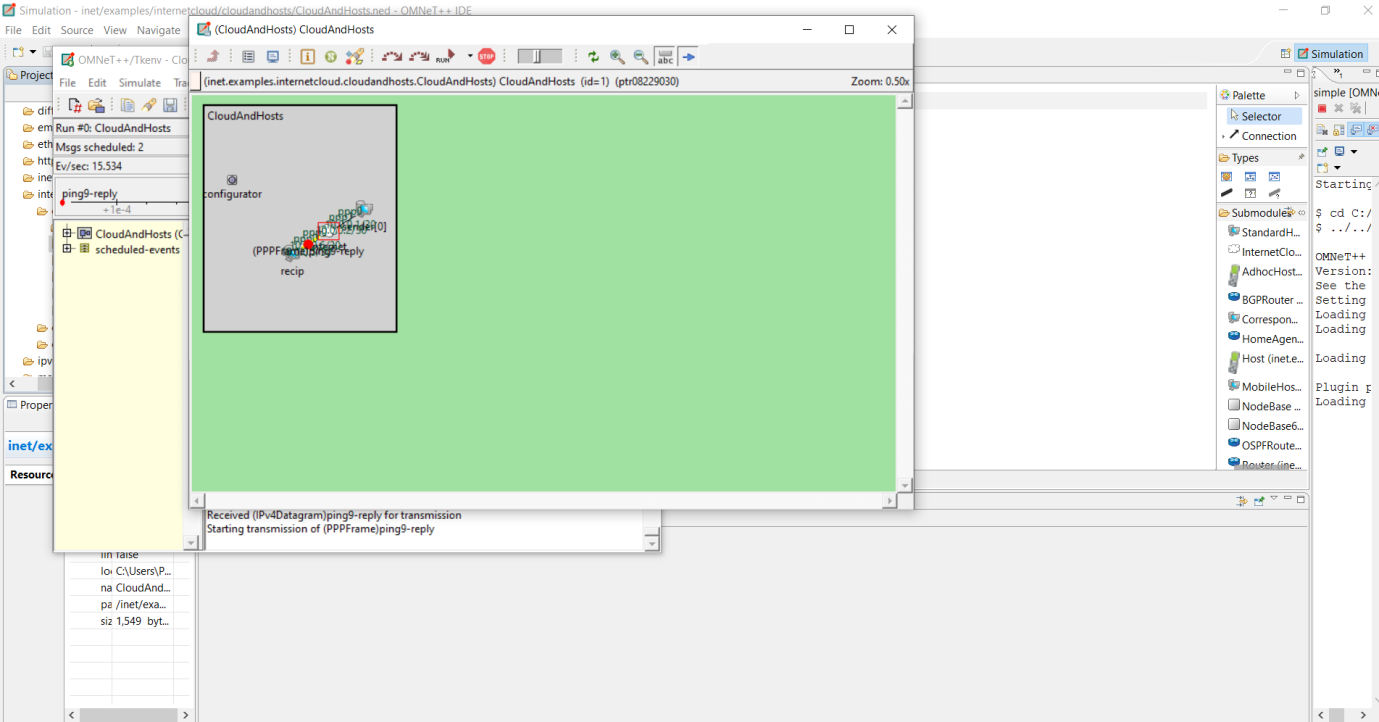
1.open inet/examples/internetcloud/cloudandhosts folder.

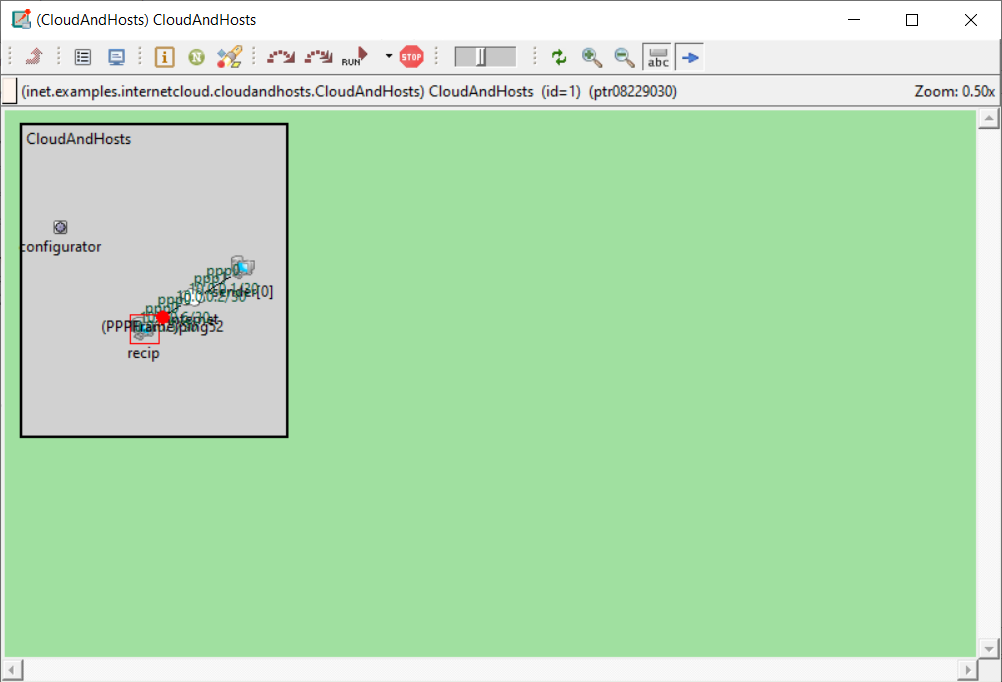


2. run file CloudAndHosts.ned with configuration “one host ping another”.



3.Click on “ok”.





**Below the code that will be available in source part of CloudAndHosts.ned file.**

**package** inet.examples.internetcloud.cloudandhosts;

**import** inet.networklayer.autorouting.ipv4.IPv4NetworkConfigurator;

**import** inet.nodes.inet.StandardHost;

**import** inet.nodes.internetcloud.InternetCloud;

**import** ned.DatarateChannel;

**network** CloudAndHosts

{

**parameters**:

**int** numSenders;

**types**:

**channel** C **extends** DatarateChannel

{

delay = 10ms;

datarate = 5Mbps;

}

**submodules**:

configurator: IPv4NetworkConfigurator {

**parameters**:

**@display**("p=61,163");

}

sender[numSenders]: StandardHost;

recip: StandardHost;

internet: InternetCloud;

**connections**:

recip.pppg++ **<-->** C **<-->** internet.pppg++;

**for** i=0..numSenders-1 {

sender[i].pppg++ **<-->** C **<-->** internet.pppg++;

}

}