Internet based Mobile Ad hoc Network (iMANETs)

Case Study Document

15CSE312

COMPUTER NETWORKS

Done by:

Pruthvi (CB.EN.U4CSE17045)
Shreebuvan V (CB.EN.U4CSE17055)
Mahesh Sivashankaran (CB.EN.U4CSE17058)
Vignesh Aravind K (CB.EN.U4CSE17069)

Contribution Table

REG.NO.	NAME	MAIL ID	CONTRIBUTION
CB.EN.U4CSE17045	Pruthvi	pothana.pruthvi@gmail.com	QOS Parameters, Characteristics and types of MANET
CB.EN.U4CSE17055	Shreebuvan V	shreebuvan99@gmail.com	Introduction, Architecture Diagram, Cisco Packet Tracer Implementation with steps, Output and Validation, Data Table
CB.EN.U4CSE17058	Mahesh Sivashankaran	sivamahesh452@gmail.com	Types of Protocols, Python code implementation, Analysis of various protocols in the python code
CB.EN.U4CSE17069	Vignesh Aravind K	vignesharavindkk@gmail.com	Analytical Questions, Description of Python code implementation, Conclusion

Table of Contents

What is a MANET:

- MANET stands for Mobile ad-hoc Network also called as wireless ad-hoc network or ad-hoc wireless network that usually has a routable networking environment on top of a Link Layer ad hoc network.
- They consist of set of mobile nodes connected wirelessly in a self-configured, self-healing network without having a fixed infrastructure. MANET nodes are free to move randomly as the network topology changes frequently.
- Each node behaves as a router as they forward traffic to other specified node in the network.

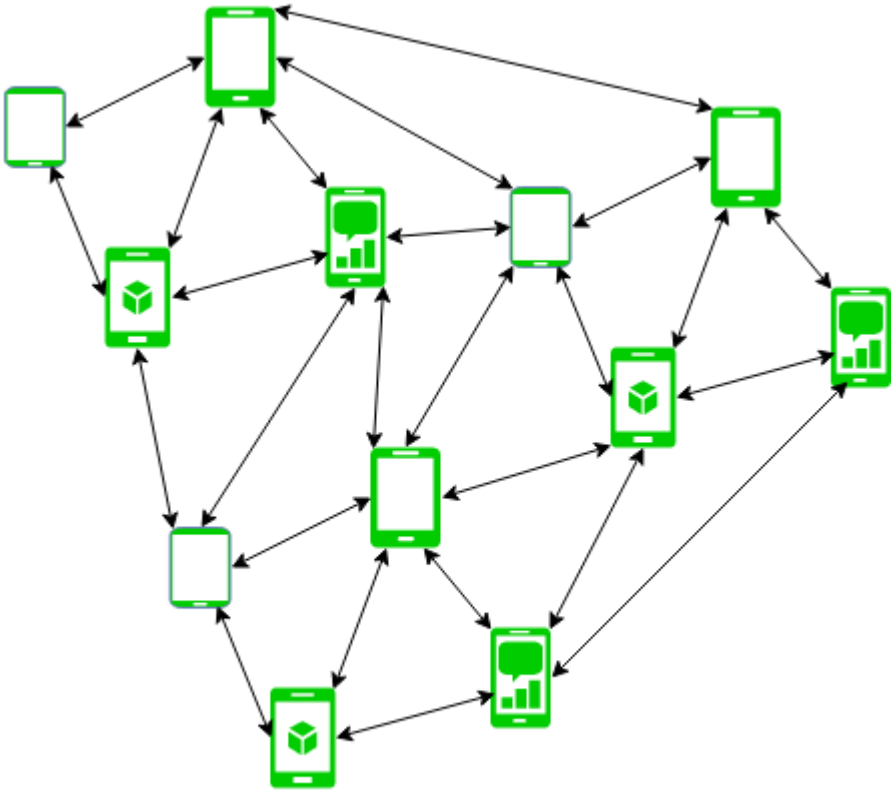


Figure - Mobile Ad Hoc Network

- MANET may operate as standalone fashion or they can be the part of larger internet. They form highly dynamic autonomous topology with the presence of one or multiple different transceivers between nodes.
- The main challenge for the MANET is to equipped each device to continuously maintain the information required to properly route traffic.
- This can be used in road safety, ranging from sensors for environment, home, health, disaster rescue operations, air/land/navy defence, weapons, robots, etc.

Characteristics of MANET:

Dynamic Topologies: Network topology which is typically multihop, may change randomly and rapidly with time, it can form unidirectional or bi-directional links.

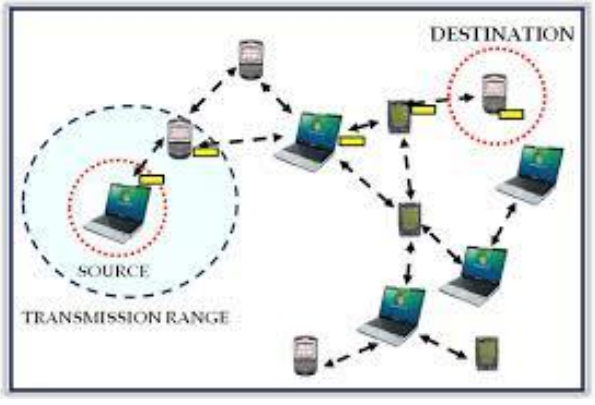
Bandwidth constrained variable capacity links: Wireless links usually have lower reliability, efficiency, stability, and capacity as compared to a wired network. The throughput of wireless communication is even less than a radio's maximum transmission rate after dealing with the constraints like multiple access, noise, interference conditions, etc.

Autonomous Behaviour: Each node can act as a host and router, which shows its autonomous behaviour

Energy Constrained Operation: As some or all the nodes rely on batteries or other exhaustible means for their energy. Mobile nodes are characterized by less memory, power, and lightweight features.

Limited Security: Wireless network is more prone to security threats. A centralized firewall is absent due to its distributed nature of the operation for security, routing, and host configuration.

Less Human Intervention: They require minimum human intervention to configure the network, therefore they are dynamically autonomous in nature



Types of MANET:

1. **Vehicular Ad hoc Network (VANETs):**
   Enable effective communication with another vehicle or with the roadside equipment's. Intelligent vehicular ad hoc networks (InVANETs) deals with another vehicle or with the roadside equipment's.
2. **Smart Phone Ad hoc Network (SPANC):**
   To create peer-to-peer network without relying on cellular carrier networks, wireless access points or traditional network infrastructure. Here peer can join or leave the network without destroying it.
3. **Internet based Mobile Ad hoc Network (iMANETs):**
   It supports internet protocols such as TCP/UDP and IP. To link mobile nodes and establish routes distributed and automatically.
4. **Hub-Spoke MANET:**
   Multiple sub MANET's may be connected in hub-spoke VPN to create a geographically distributed MANET. Normal Ad-hoc routing algorithm does not apply directly.
5. **Military or Tactical MANETs:**
   This is used by the military units. Emphasis on data rate, real time demand, fast re-routing during mobility, security, radio range, etc.
6. **Flying Ad hoc Network (FANETs):**
   This is composed of unmanned aerial vehicle (commonly known as drone). Provides links to remote areas and mobility.

Topic of Interest:

Internet based Mobile Ad hoc Network (iMANETs)

Architecture Diagram:

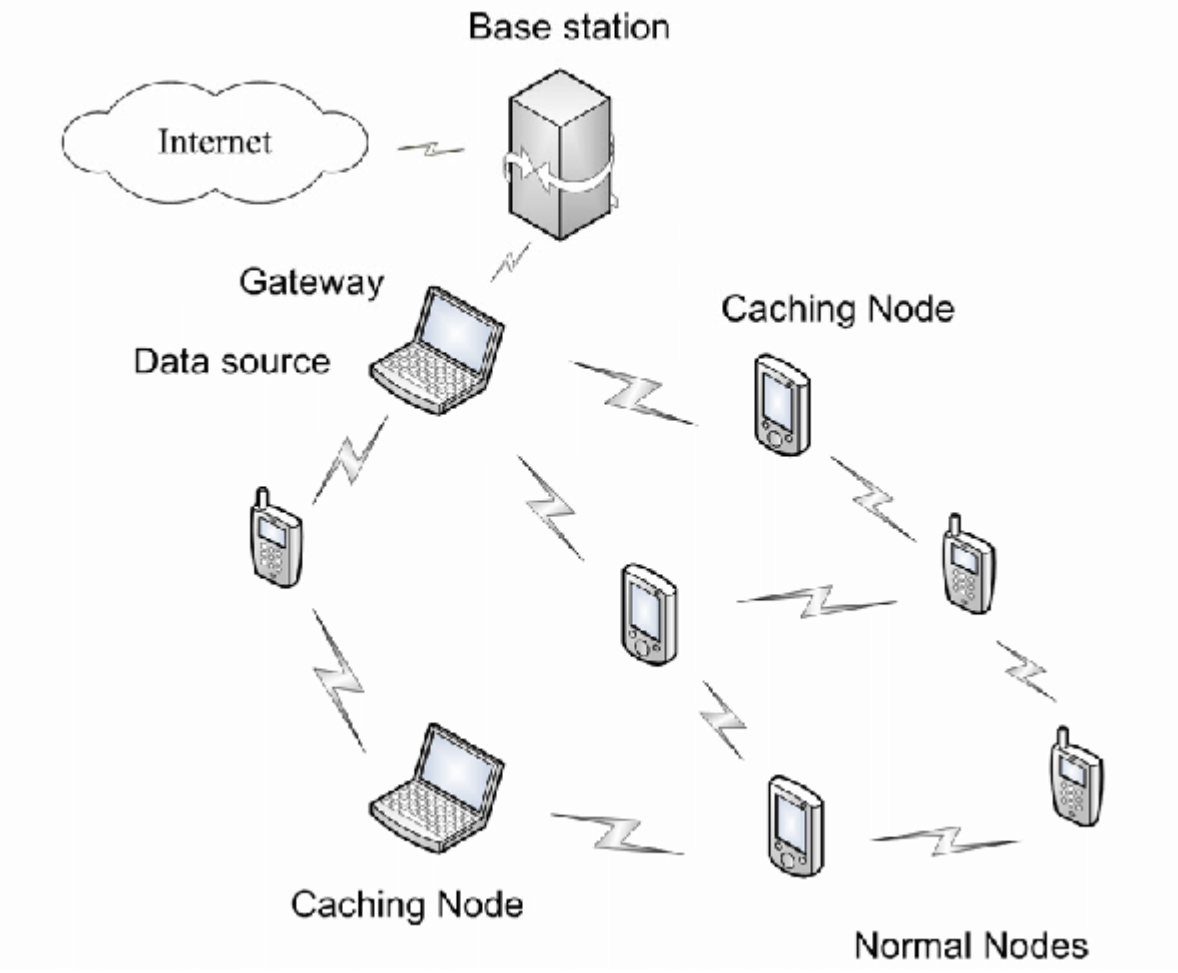


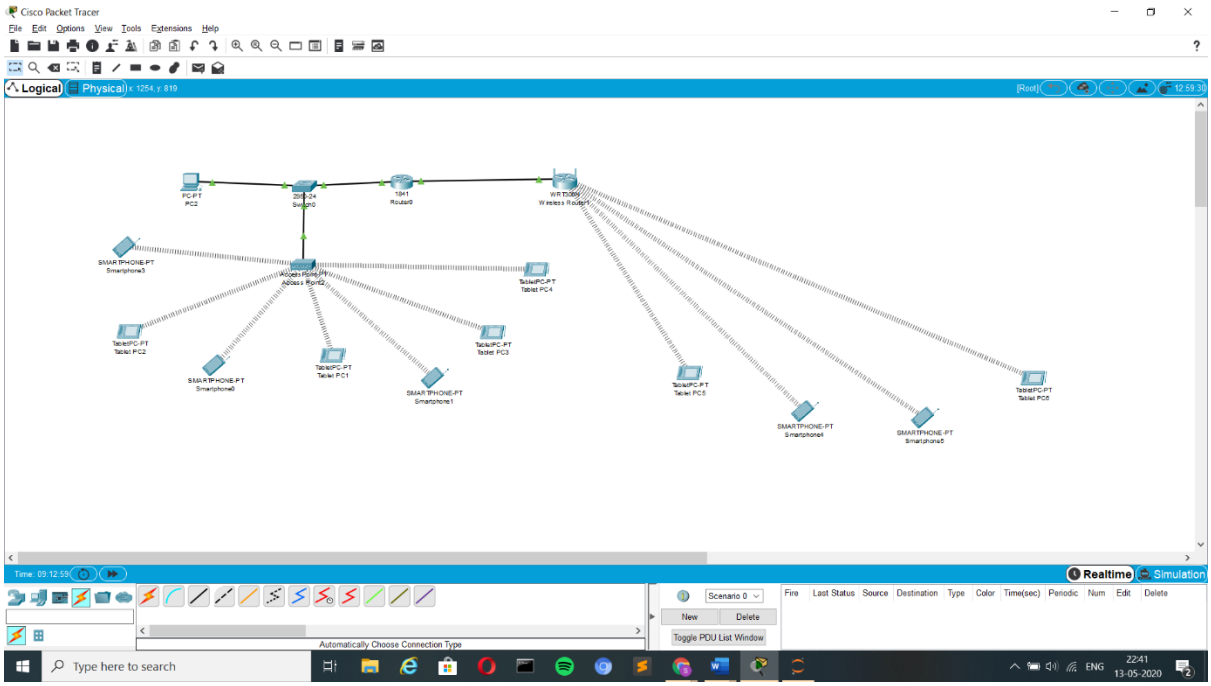
Image of how typically an iMANET looks like

Analytical Questions:

1. Flying ad hoc networks have their own characteristic Compared to iMANET Is there a difference in their layered architecture?
2. How iMANET works?
3. What factors influence the selection of data paths on routing?
4. Why iMANET are less secure?
5. How to explain the dual role of a mobile node in a MANET?
6. What changes in a network over time as mobile devices add or leave a network?
7. Can ad-Hoc networks be used by multiple devices? Why?
8. What is the difference between WANET and iMANET?
9. What are the algorithms that used for traffic signals and intersection control?
10. What are the present issues in iMANETS?

Implementation of iMANET in Cisco Packet Tracer:

The image below shows the final iMANET implemented in Cisco Packet Tracer



Components present in the above image:

1. 5 Smartphones
2. 6 Tablet PC
3. 1 Access point
4. 1 PC
5. 1 1841 Router
6. 1 Wireless Router

From the above image we can see two networks with IP's 192.168.1.x (Right portion) and 192.168.2.x (left portion).

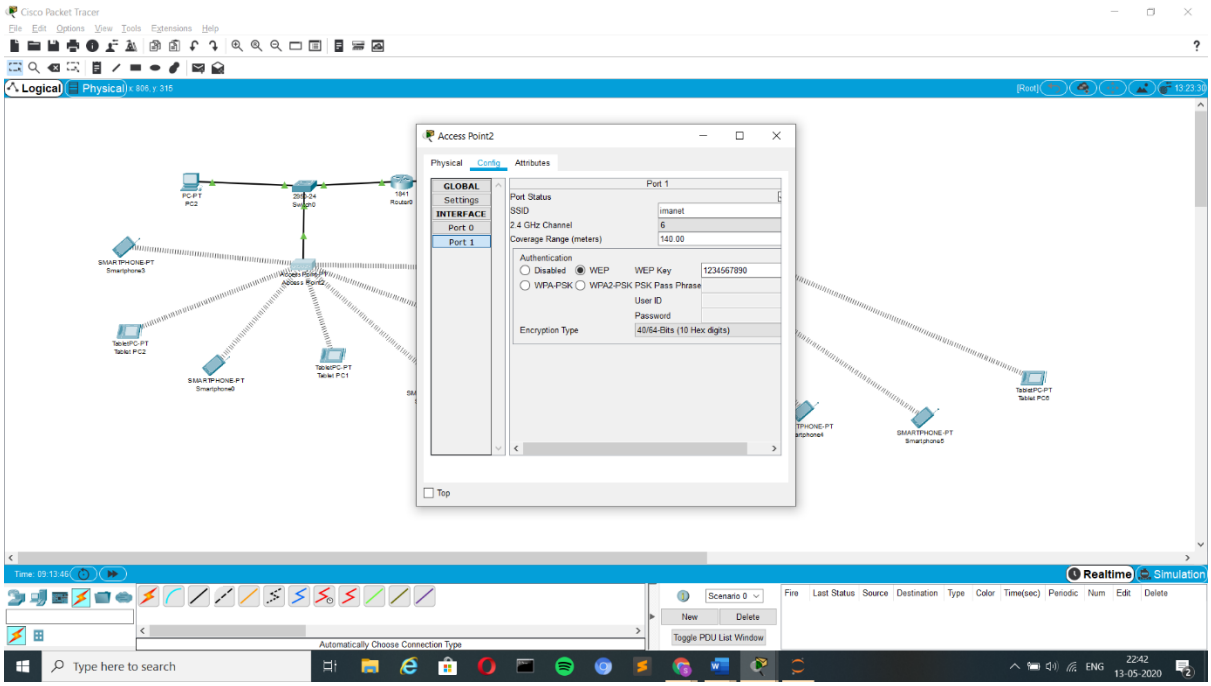
Note: Data Table is also provided after Output and Validation

GitHub Link:

https://github.com/shreebuvan/Networks-Case-Study/blob/master/CaseStudy.pkt

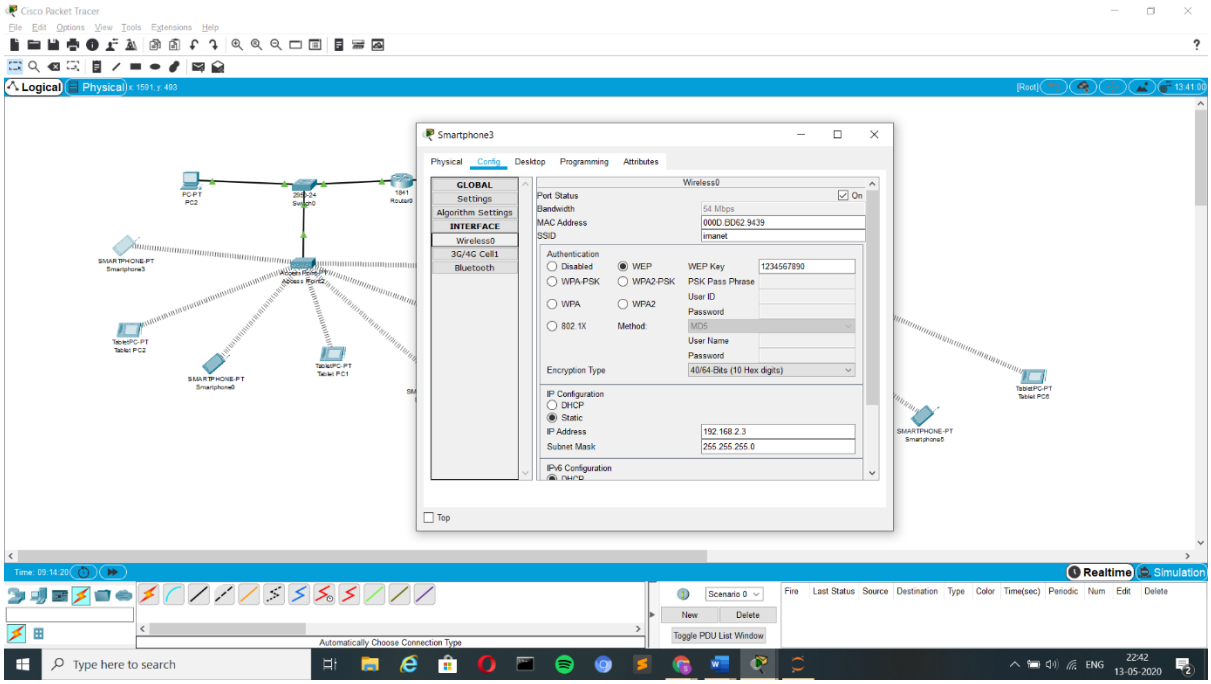
The steps to be followed for the construction of the above network:

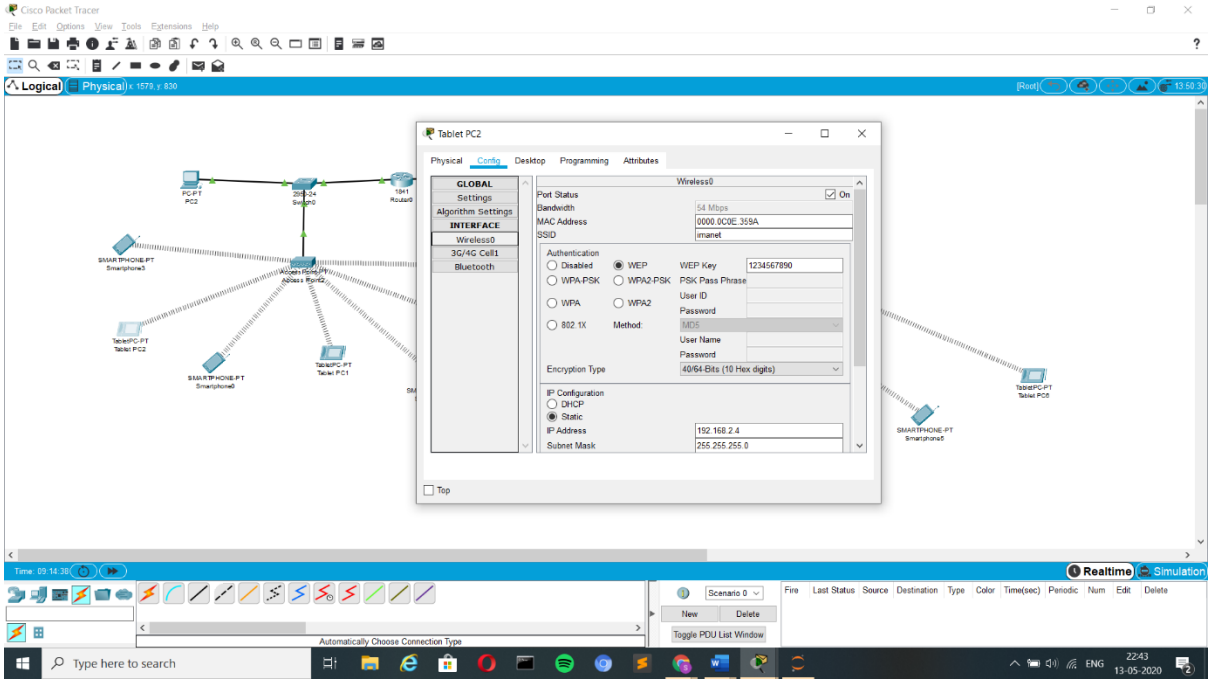
1. First, we have to configure the devices in the left with one network address (192.168.2.x).
2. Next, we have to configure the devices in the right with one network address (192.168.1.x).
3. Now we can configure the Access point with a WEP Key and a SSID in port 1. (In this case it is 1234567890 and iMANET)



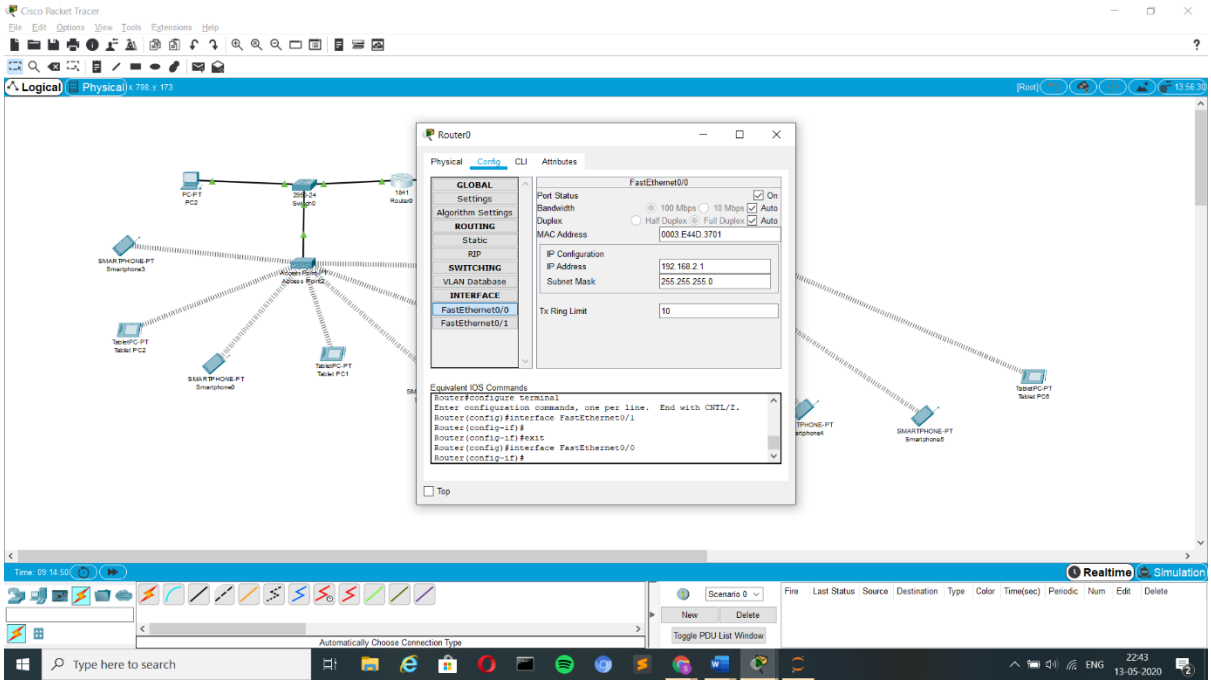
4. Now in turn this has to be configured for all the devices in the left portion of the network. Where each device has an option called wireless in their config section.

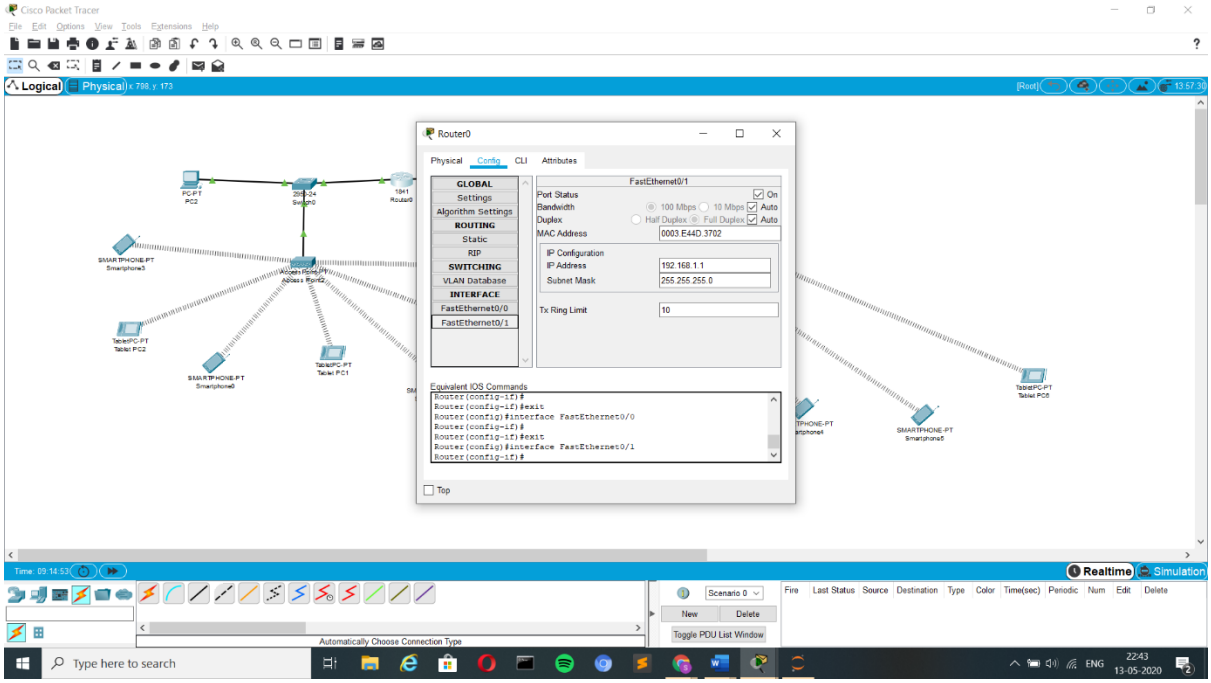
5. This wireless section in every device in the left has to be configured with the WEP Key and SSID. (In this case it is 1234567890 and iMANET)



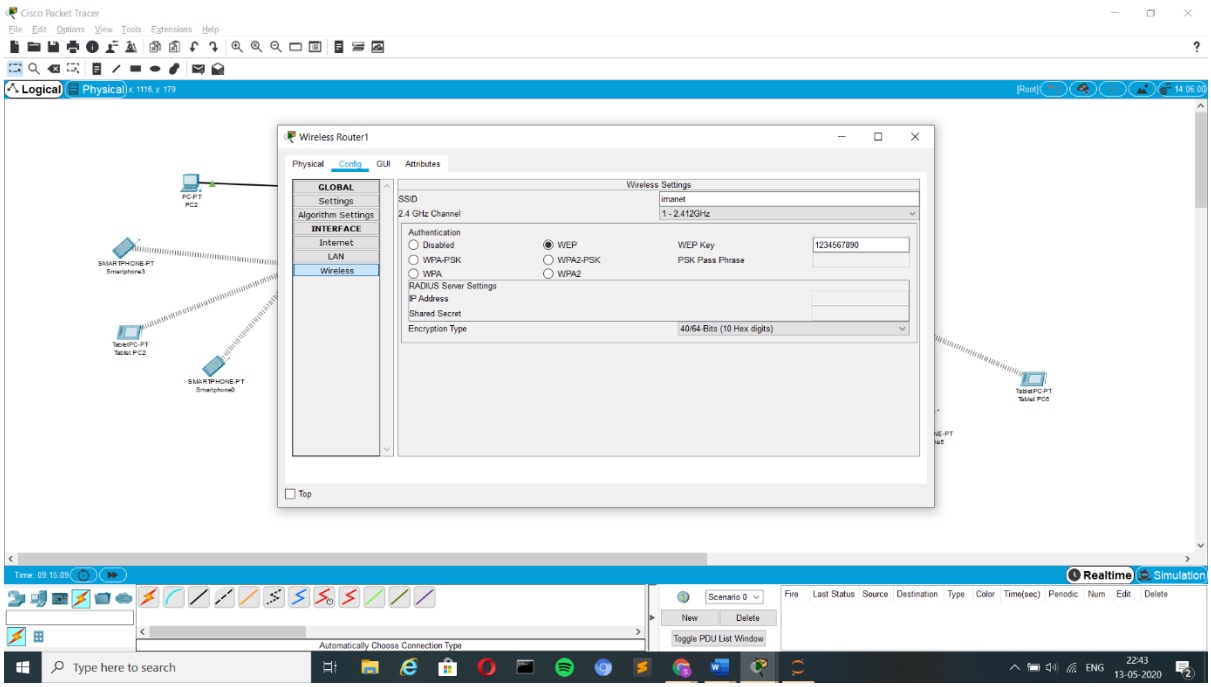


6. Now next the 1841 Router has to be configured with the networks address for the left (192.168.2.1) and for the right (192.168.1.1).

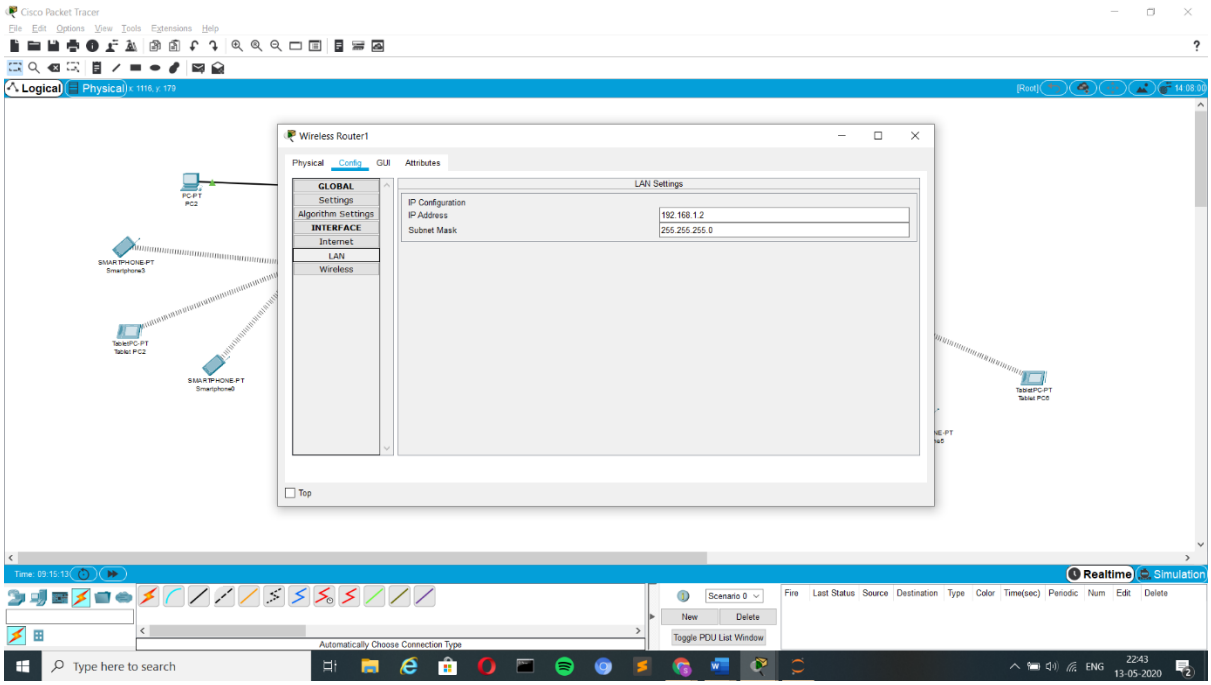


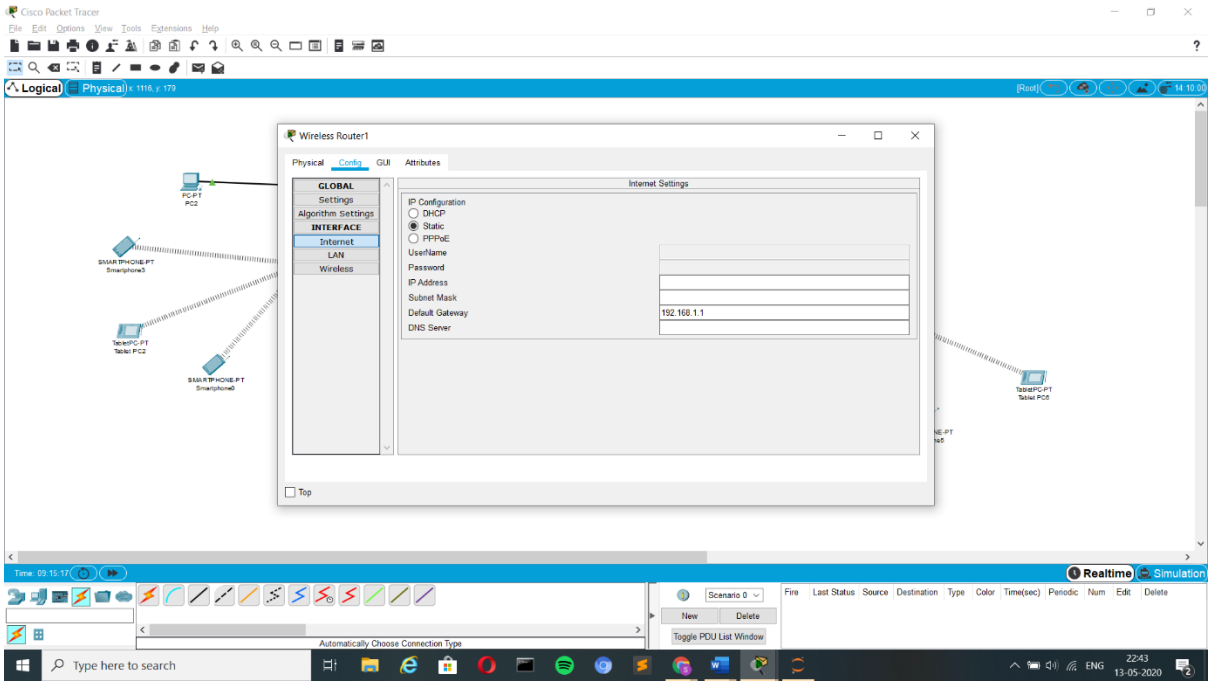


7. Next, we have to configure the wireless router. The wireless section in the config in given with the WEP Key and SSID. (In this case it is 1234567890 and iMANET)

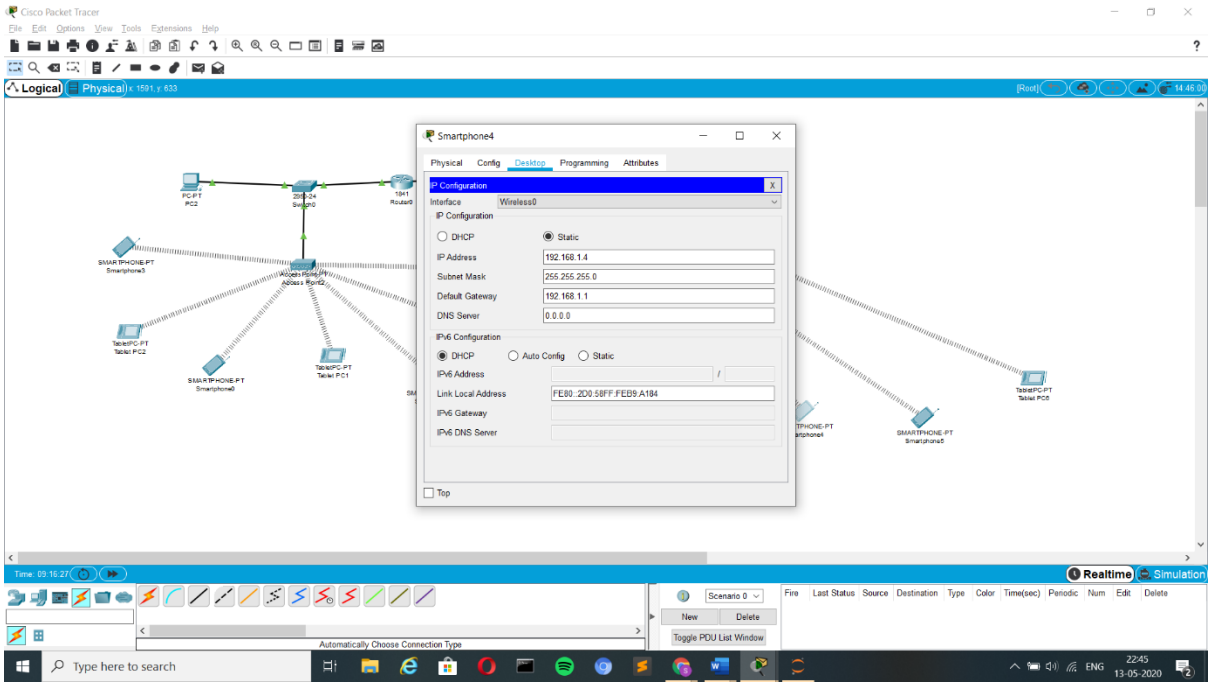


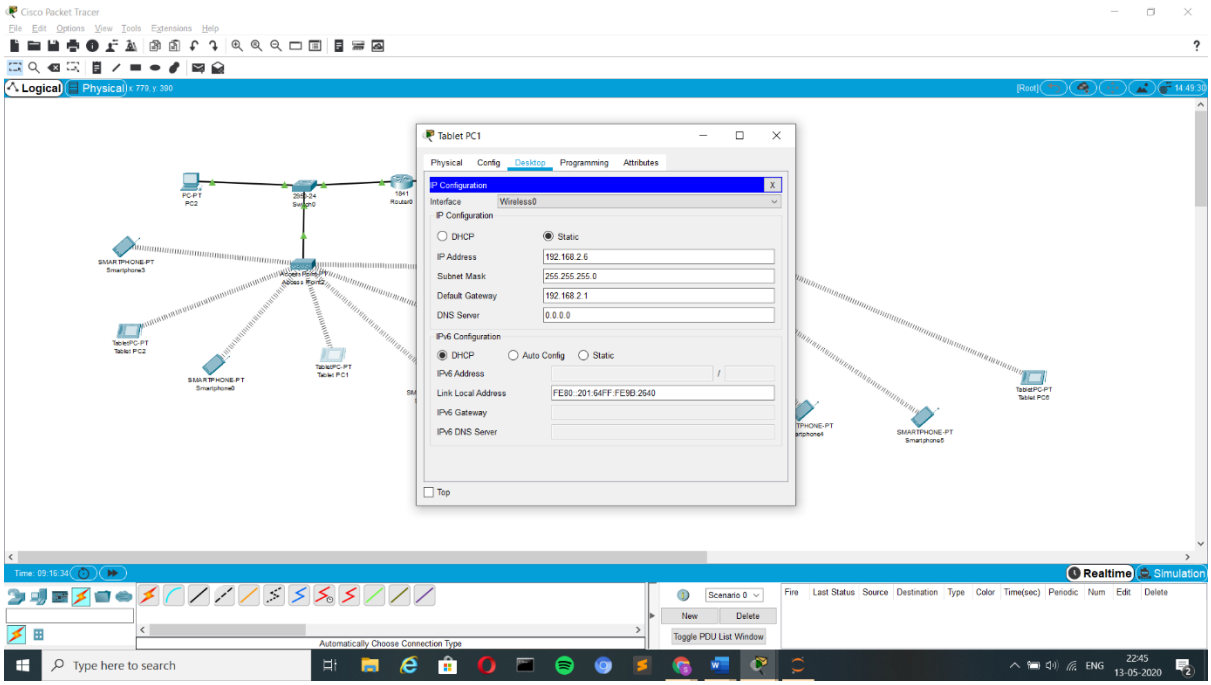
8. Then, it is given with the IP Address 192.168.1.2 in the LAN section in the config. Finally, the default gateway is set to 192.168.1.1 under the internet section in the config.



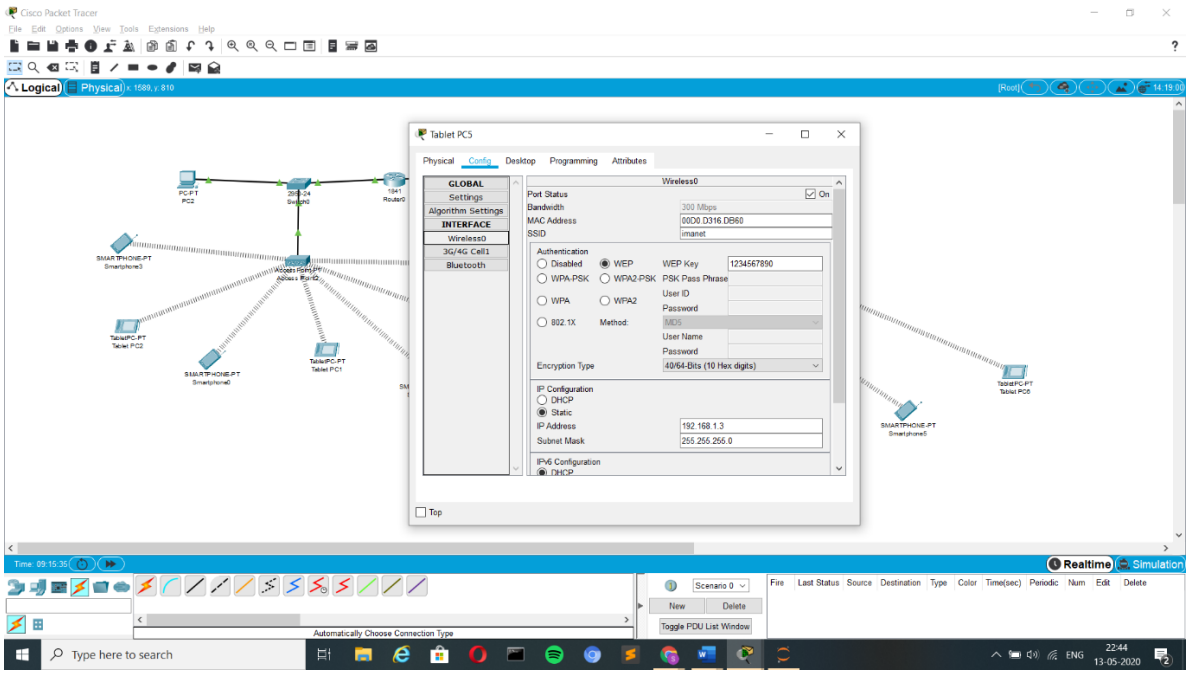


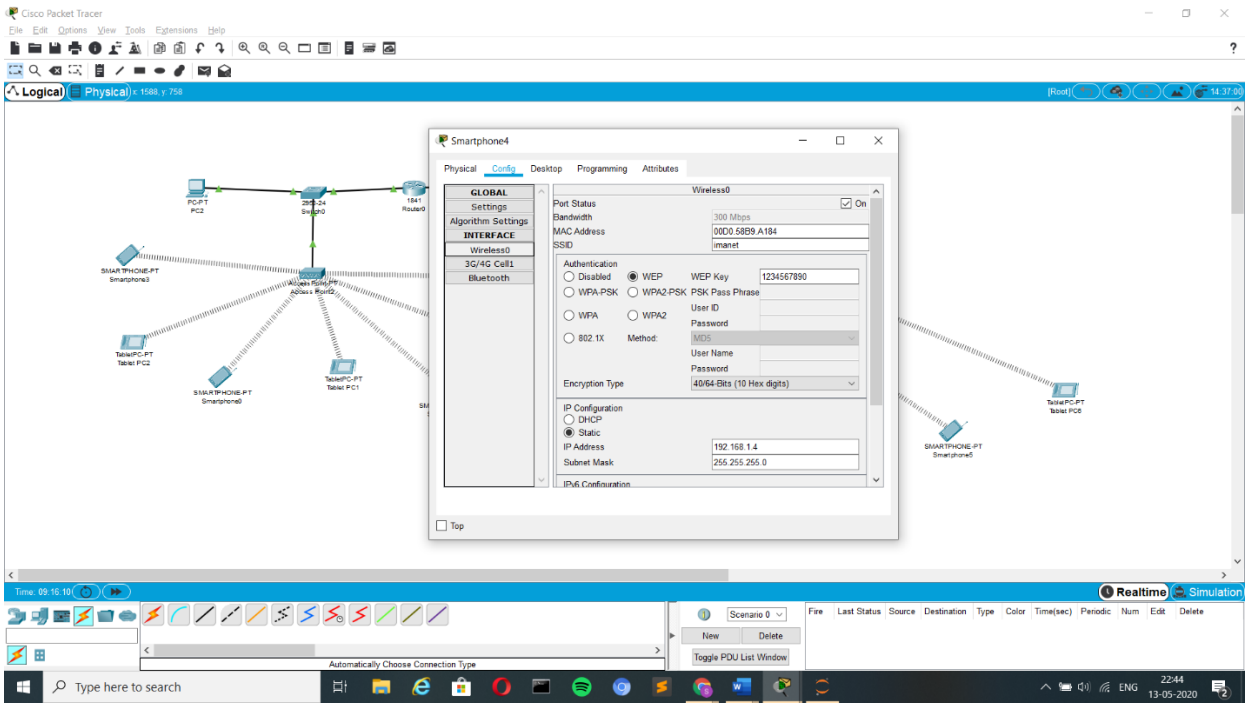
9. Now next we have to configure with devices in the right with the IP's in the family 192.168.1.x.





10. Finally, all the devices in the right portion of the network has to be configured. Where each device has an option called wireless in their config section.
11. This wireless section in every device in the left has to be configured with the WEP Key and SSID. (In this case it is 1234567890 and iMANET)





Final Outcome Video link:

GitHub Link:

https://github.com/shreebuvan/Networks-Case-Study/blob/master/Cisco_Packet_Tracer_implemetation_video.mp4

Drive Link:

https://drive.google.com/open?id=1Sq79X2Xwb8OmWOvi7gA1XaexnnKyuX5T

Note:

The video consists of both the implementation and the validation of the output part also in it.

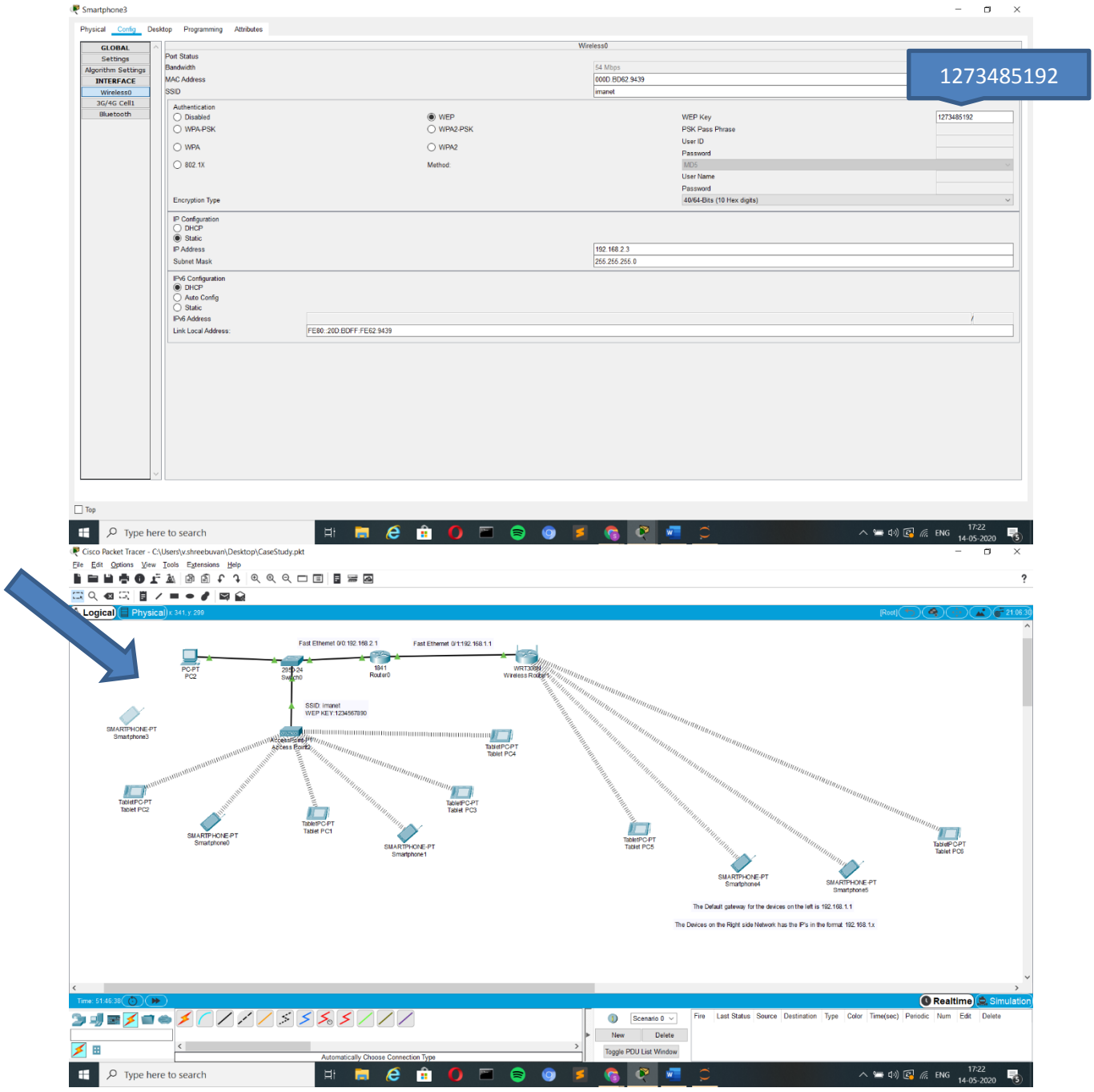
Output and Validation:

The wireless can be established only if the credentials are given correctly in the devices

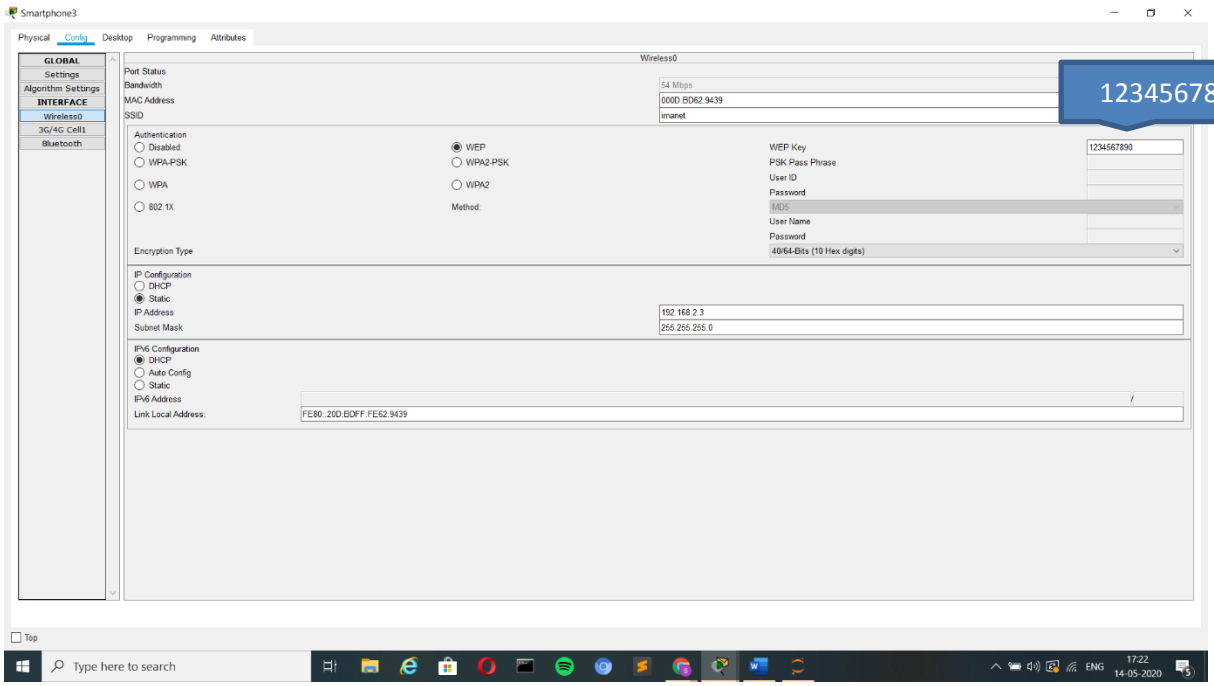
Valid Credentials:

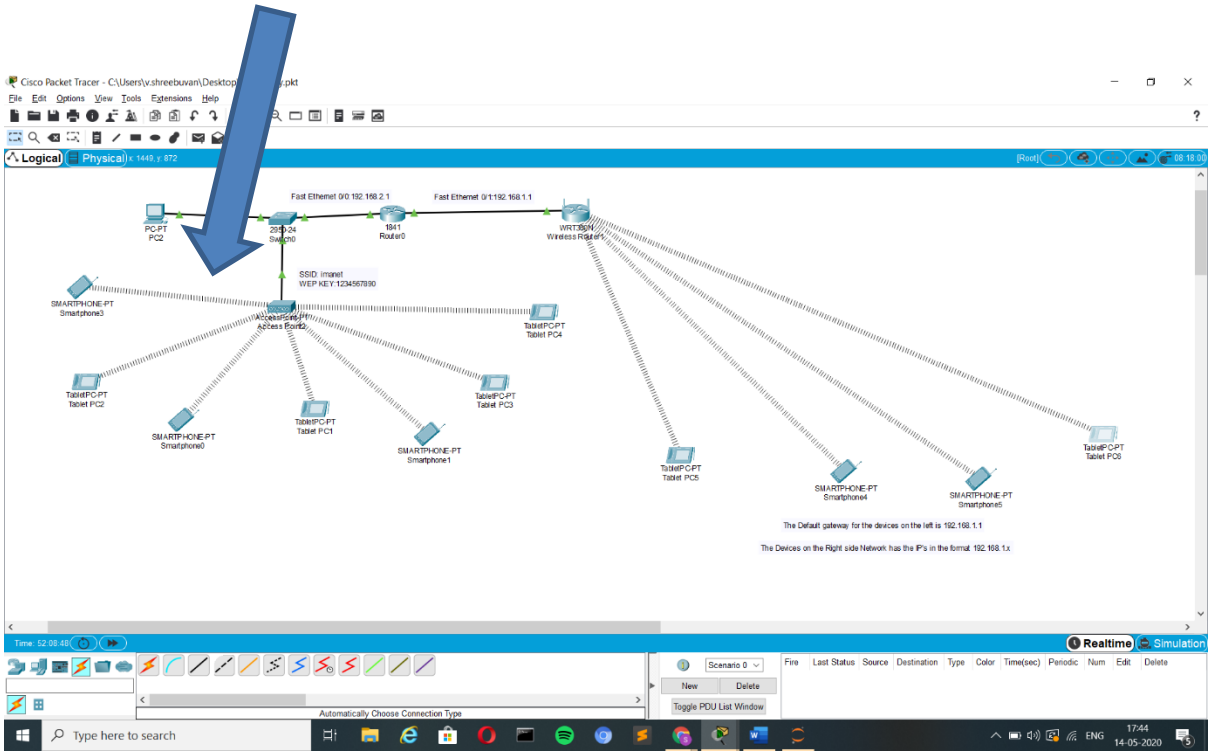
SSID: imanet WEP KEY: 1234567890

Incorrect credentials:



Correct Credentials:

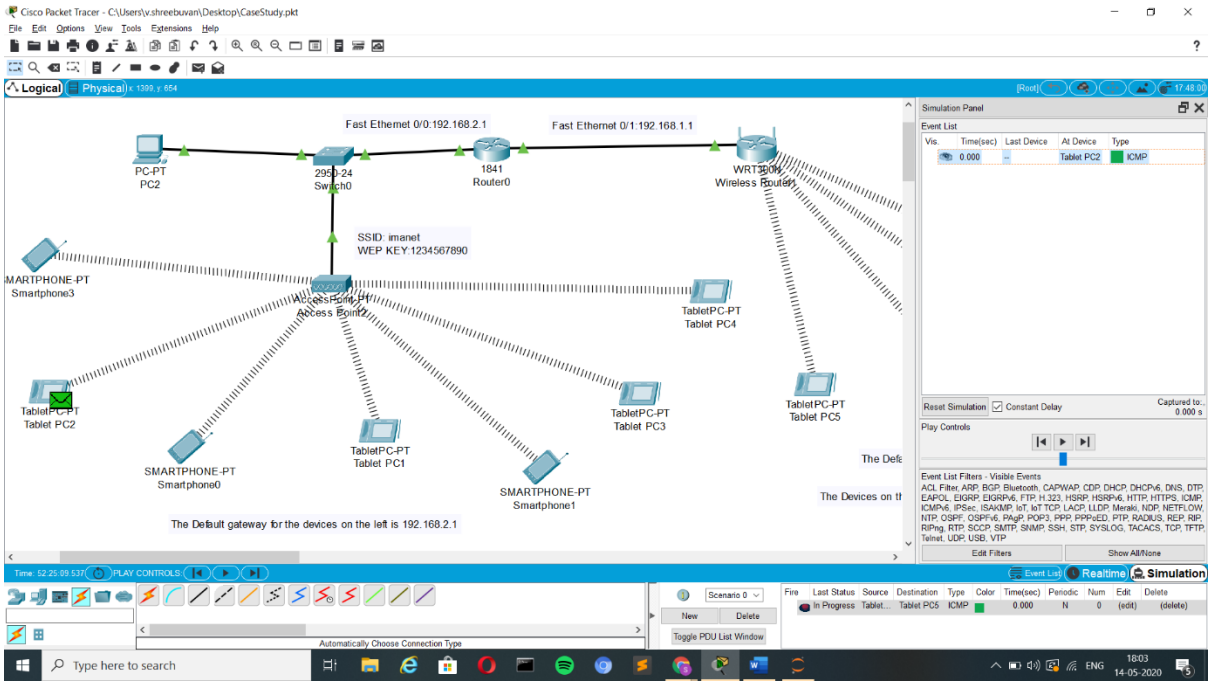




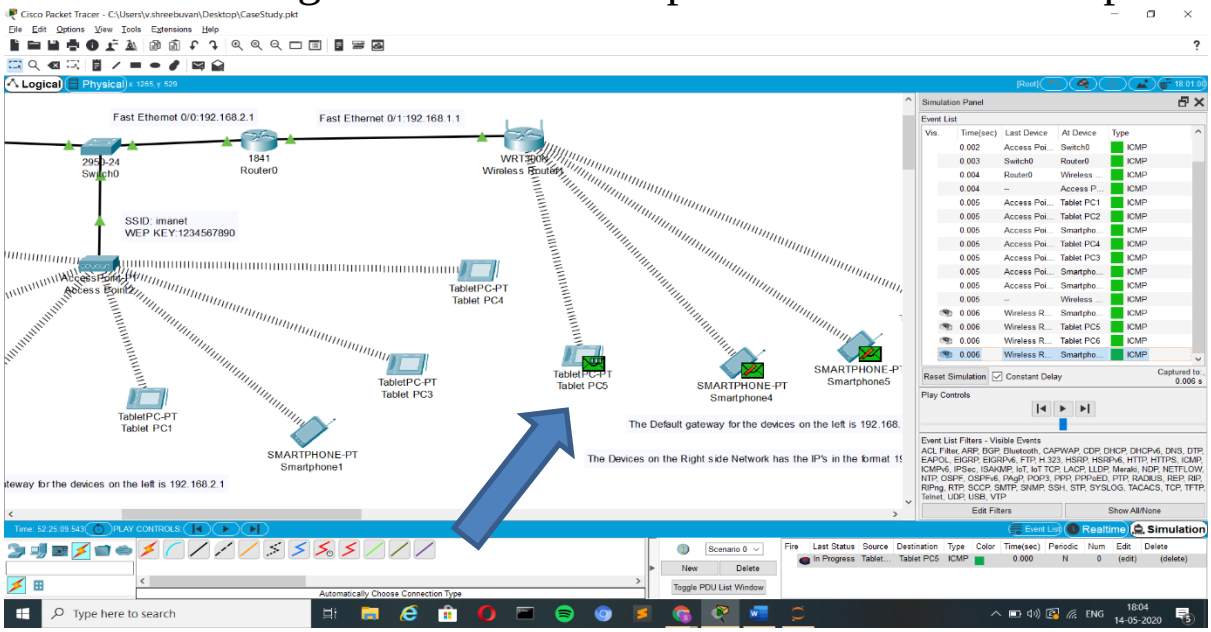
Validation on message passing from one device to other:

Simulation:

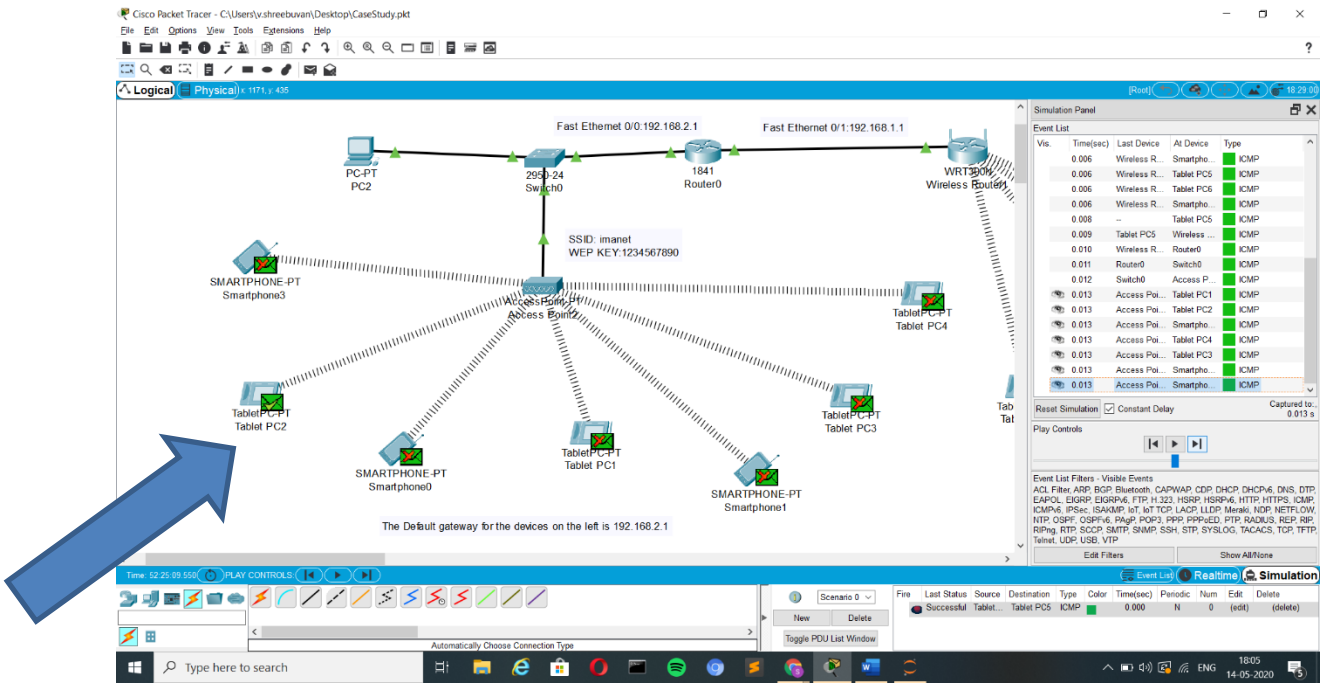
Message sent from TabletPC2 to TabletPC5.



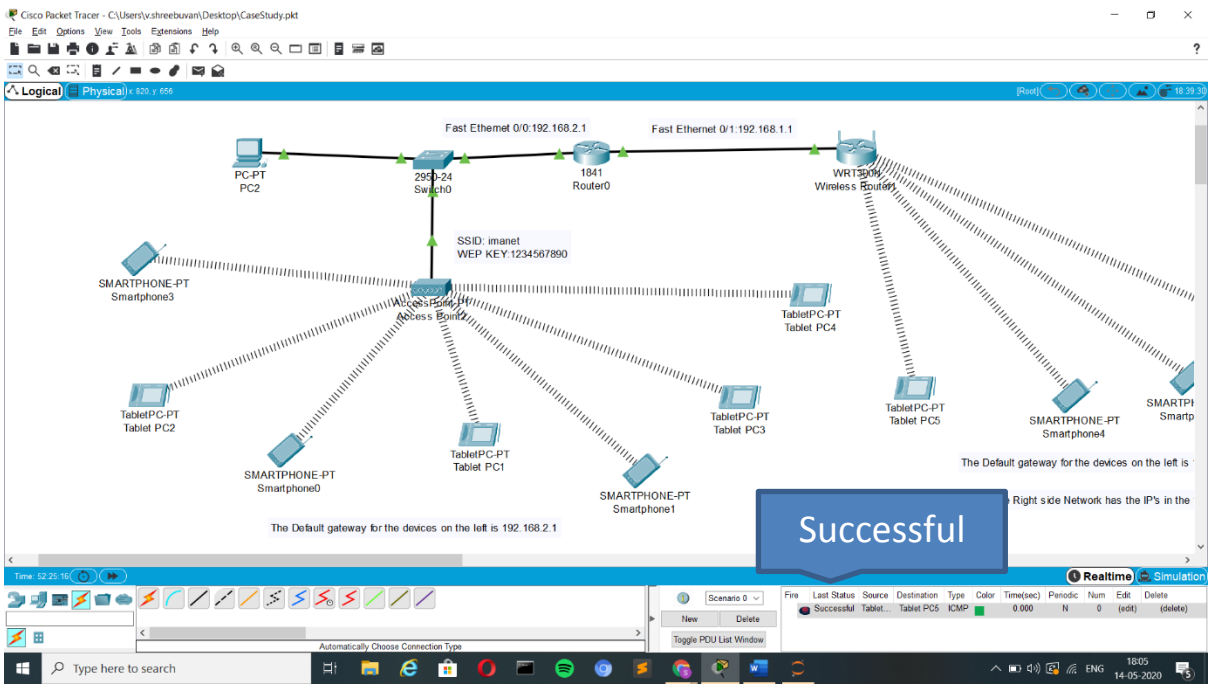
Here the message send from tabletpc2 has reached tablet pc5.



The message received conformation from TabletPC5 is send successfully to TabletPC2



Realtime:



Data Tables:

IP Addressing:

Device Name	IP Address
1841 Router0 (Fast Ethernet 0/0)	192.168.2.1/24
1841 Router0 (Fast Ethernet 0/1)	192.168.1.1/24
WRT300N Wireless Router1	192.168.1.2/24
PC-PT(PC2)	192.168.2.2/24
SMARTPHONE-PT (Smartphone 3)	192.168.2.3/24
TabletPC-PT (Tablet PC2)	192.168.2.5/24
SMARTPHONE-PT (Smartphone 0)	192.168.2.5/24
TabletPC-PT (Tablet PC1)	192.168.2.6/24
SMARTPHONE-PT (Smartphone 1)	192.168.2.7/24
TabletPC-PT (Tablet PC3)	192.168.2.8/24
TabletPC-PT (Tablet PC4)	192.168.2.9/24
TabletPC-PT (Tablet PC5)	192.168.1.3/24
SMARTPHONE-PT (Smartphone 4)	192.168.1.4/24
SMARTPHONE-PT (Smartphone 5)	192.168.1.5/24
TabletPC-PT (Tablet PC6)	192.168.1.6/24

Note:

- SSID: imanet
- WEP KEY: 1234567890

DESTINATION-SEQUENCED DISTANCE VECTOR ROUTING (DSDV):

Destination Sequenced Distance Vector (DSDV) is a hop-by-hop vector routing protocol requiring each node to periodically broadcast routing updates. This is a table-driven algorithm based on modifications made to the Bellman-Ford routing mechanism. Each node in the network maintains a routing table that has entries for each of the destinations in the network and the number of hops required to reach each of them. Each entry has a sequence number associated with it that helps in identifying stale entries. This mechanism allows the protocol to avoid the formation of routing loops. Each node periodically sends updates tagged throughout the network with a monotonically increasing even sequence number to advertise its location. New route broadcasts contain the address of the destination, the number of hops to reach the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labelled with the most recent sequence number is always used. When the neighbours of the transmitting node receive this update, they recognize that they are one hop away from the source node and include this information in their distance vectors. Every node stores the "next routing hop" for every reachable destination in their routing table.

Dynamic Source Routing (DSR):

Dynamic source routing protocol (DSR) is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The major difference between this and the other on-demand routing protocols is that it is beacon-less and hence does not require periodic hello packet (beacon) transmissions, which are used by a node to inform its neighbours of its presence. The basic approach of this protocol (and all other on-demand routing protocols) during the route construction phase is to establish a route by flooding Route-Request packets in the network. The destination node, on receiving a Route-Request packet, responds by sending a Route-Reply packet back to the source, which carries the route traversed by the Route-Request packet received.

Ad Hoc On-Demand Distance Vector (AODV):

The AODV protocol builds routes between nodes only if they are requested by source nodes. AODV is therefore considered an on-demand algorithm and does not create any extra traffic for communication along links. The routes are maintained as long as they are required by the sources. They also form trees to connect multicast group members. AODV makes use of sequence numbers to ensure route freshness. They are self-starting and loop-free besides scaling to numerous mobile nodes.

Python script to count the number of packets sent, received, dropped and time taken to run the simulation in DSDV, DSR, AODV Protocols in MANET (Mobile Ad-hoc Network):

GitHub Link: https://github.com/shreebuvan/Networks-Case-Study/blob/master/Algorithm_implementation.py

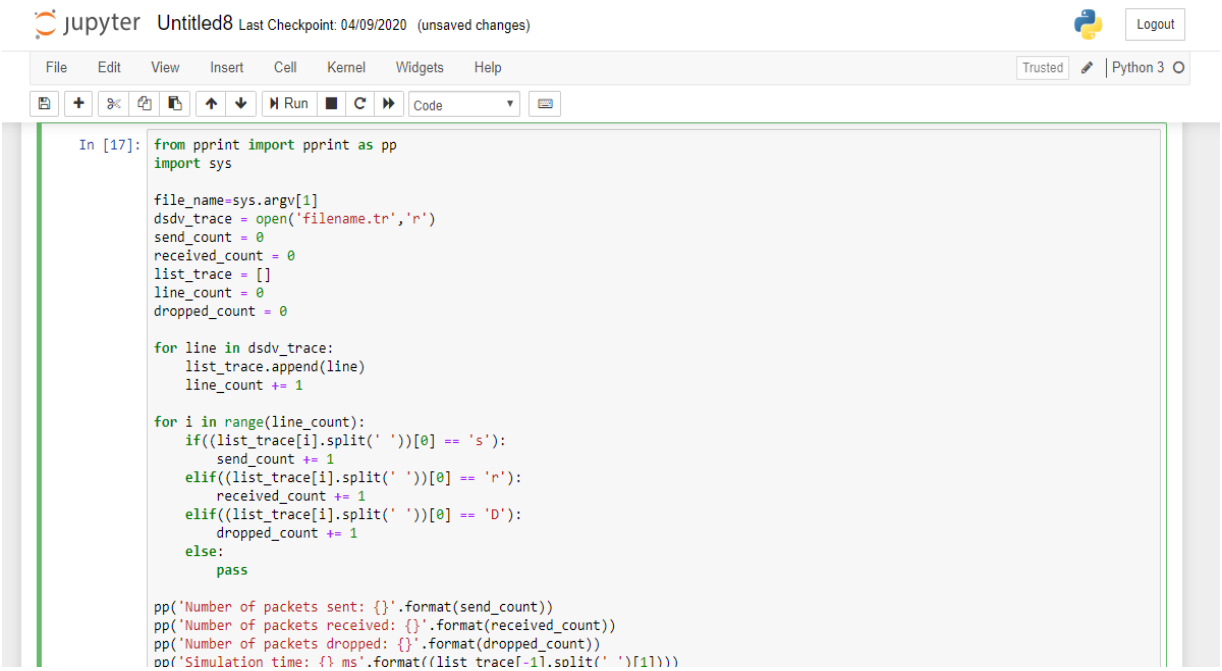
```
from pprint import pprint as pp

import sys

file_name=sys.argv[1]

dsdv_trace = open('filename.tr','r')

send_count = 0

received_count = 0

list_trace = []

line_count = 0

dropped_count = 0


for line in dsdv_trace:

    list_trace.append(line)

    line_count += 1

for i in range(line_count):

    if((list_trace[i].split(' '))[0] == 's'):

        send_count += 1

    elif((list_trace[i].split(' '))[0] == 'r'):

        received_count += 1

    elif((list_trace[i].split(' '))[0] == 'D'):

        dropped_count += 1

    else:

        pass
```

pp('Number of packets sent: {}'.format(send_count))

pp('Number of packets received: {}'.format(received_count))

pp('Number of packets dropped: {}'.format(dropped_count))

pp('Simulation time: {} ms'.format((list_trace[-1].split(' ')[1])))



Description:

The above block of code is used to count the number of packets sent, received, dropped and time taken to run the simulation in DSDV, DSR, AODV Protocols in MANET (Mobile Ad-hoc Network).

```
from pprint import pprint as pp

import sys
```

- Import statements to access the file system in-order to read the trace file.

```
file_name=sys.argv[1]

dsdv_trace = open('filename.tr','r')
```

- It is used to read the contents of the file stored in the system.

```
send_count = 0

received_count = 0

list_trace = []

line_count = 0

dropped_count = 0
```

- Initializing all the variables to the initial value as shown above.

```
for line in dsdv_trace:

    list_trace.append(line)

    line_count += 1
```

- To read the data line by line from the trace file and append the lines in list named list_trace and incrementing the count of lines.

```
for i in range(line_count):
    if((list_trace[i].split(' '))[0] == 's'):
        send_count += 1
    elif((list_trace[i].split(' '))[0] == 'r'):
        received_count += 1
    elif((list_trace[i].split(' '))[0] == 'D'):
        dropped_count += 1
    else:
        pass
```

- Read the appended data from the list and to check whether the packet is sent, received or dropped from the first character of the line.
    1. If the character is -s then the packet is sent.
    2. If the character is -r then the packet is received
    3. If the character is -D then the packet is dropped.
- Based on the condition above the corresponding variable is incremented.

```
pp('Number of packets sent: {}'.format(send_count))
pp('Number of packets received: {}'.format(received_count))
pp('Number of packets dropped: {}'.format(dropped_count))
pp('Simulation time: {} ms'.format((list_trace[-1].split(' ')[1])))
```

- Here the packets send, received, dropped and the time taken for simulation are shown below as output in the console.

Analysis for DSDV Protocol:

DSDV trace file link:
https://raw.githubusercontent.com/softvar/ns2-roadv/master/dsdv.tr

GitHub link: https://github.com/shreebuvan/Networks-Case-Study/blob/master/dsdv.tr

Output:

In [25]:
```
from pprint import pprint as pp
import sys

file_name=sys.argv[1]
dsdv_trace = open('dsdv.tr','r')
send_count = 0
received_count = 0
list_trace = []
line_count = 0
dropped_count = 0

for line in dsdv_trace:
    list_trace.append(line)
    line_count += 1

for i in range(line_count):
    if((list_trace[i].split(' '))[0] == 's'):
        send_count += 1
    elif((list_trace[i].split(' '))[0] == 'r'):
        received_count += 1
    elif((list_trace[i].split(' '))[0] == 'D'):
        dropped_count += 1
    else:
        pass

pp('Number of packets sent: {}'.format(send_count))
pp('Number of packets received: {}'.format(received_count))
pp('Number of packets dropped: {}'.format(dropped_count))
```

```
'Number of packets sent: 5267'
'Number of packets received: 5775'
'Number of packets dropped: 0'
```

Analysis for DSR Protocol:

DSR trace file link:
https://raw.githubusercontent.com/softvar/ns2-roadv/master/Dsr.tr

GitHub link: https://github.com/shreebuvan/Networks-Case-Study/blob/master/dsr.tr

Output:

In [22]:
```
from pprint import pprint as pp
import sys

file_name=sys.argv[1]
dsdv_trace = open('dsr.tr','r')
send_count = 0
received_count = 0
list_trace = []
line_count = 0
dropped_count = 0

for line in dsdv_trace:
    list_trace.append(line)
    line_count += 1

for i in range(line_count):
    if((list_trace[i].split(' '))[0] == 's'):
        send_count += 1
    elif((list_trace[i].split(' '))[0] == 'r'):
        received_count += 1
    elif((list_trace[i].split(' '))[0] == 'D'):
        dropped_count += 1
    else:
        pass

pp('Number of packets sent: {}'.format(send_count))
pp('Number of packets received: {}'.format(received_count))
pp('Number of packets dropped: {}'.format(dropped_count))
pp('Simulation time: {} ms'.format((list_trace[-1].split(' ')[1])))
```

```
'Number of packets sent: 9338'
'Number of packets received: 15842'
'Number of packets dropped: 3'
'Simulation time: 81.196998408 ms'
```

Analysis for AODV Protocol:

AODV trace file content:

s 0.000000000 0 RTR — 0 AODV 44 [0 0 0 0] ——– [0:255 -1:255 1 0] [0x11 [0 2] 4.000000] (HELLO)

s 10.000000000 0 RTR — 0 AODV 48 [0 0 0 0] ——– [0:255 -1:255 30 0][0x2 1 1 [1 0] [0 4]] (REQUEST)

s 21.500000000 0 RTR — 0 AODV 48 [0 0 0 0] ——– [0:255 -1:255 30 0][0x2 1 4 [1 0] [0 12]] (REQUEST)

r 21.501260809 2 RTR — 0 AODV 48 [0 ff ff ff ff 0 800] ——– [0:255 -1:25530 0] [0x2 1 4 [1 0] [0 12]] (REQUEST)

GitHub link: https://github.com/shreebuvan/Networks-Case-Study/blob/master/aodv.tr

Output:

In [23]:
```
from pprint import pprint as pp
import sys

file_name=sys.argv[1]
dsdv_trace = open('aodv.tr','r')
send_count = 0
received_count = 0
list_trace = []
line_count = 0
dropped_count = 0

for line in dsdv_trace:
    list_trace.append(line)
    line_count += 1

for i in range(line_count):
    if((list_trace[i].split(' '))[0] == 's'):
        send_count += 1
    elif((list_trace[i].split(' '))[0] == 'r'):
        received_count += 1
    elif((list_trace[i].split(' '))[0] == 'D'):
        dropped_count += 1
    else:
        pass

pp('Number of packets sent: {}'.format(send_count))
pp('Number of packets received: {}'.format(received_count))
pp('Number of packets dropped: {}'.format(dropped_count))
pp('Simulation time: {} ms'.format((list_trace[-1].split(' ')[1])))
```

```
'Number of packets sent: 3'
'Number of packets received: 1'
'Number of packets dropped: 0'
'Simulation time: 21.501260809 ms'
```

QOS Parameters:

End-to-End Delay:
End-to-end delay or one-way delay (OWD) refers to the time taken for a packet to be transmitted across a network from source to destination. It is a common term in IP network monitoring, and differs from round-trip time (RTT) in that only path in the one direction from source to destination is measured.
end-to-end Delay is N*L/R (N links in series, using store-and-forward between the links)
total delay =(N+P-1) *L/R
Where
N = link, L = packet length, R = transmission rate

Throughput:

Throughput (also known as the flow rate) is a measure of a process flow rate. Essentially, it measures the movements of inputs and outputs within the production process through any mode of the communication channel

Throughput= (Total Packets received*size of packet*8)/time

Packet Loss Ratio:

The ratio of total packet lost while transmission and the total packet sent gives the packet loss ratio. With many consequences of error-free transmission, there are some sources which cause an error in the delivery

Packet Loss= ((Total packet sent-Total-Total packet received)/Total packet sent) *100

Transmission Rate:

The transmission time is the amount of time from the beginning until the end of a message transmission. In the case of a digital message, it is the time from the first bit until the last bit of a message has left the transmitting node

Transmission time=Packet size/Bit rate

Propagation Time:

The amount of time it takes for the head of the signal to travel from the sender to the receiver. It can be computed as the ratio between the link length and the propagation speed over the specific medium.

Propagation Time=Distance of the link/Propagation speed

Processing Delay:

the time it takes routers to process the packet header. During the processing of a packet, routers may check for bit-level errors in the packet that occurred during transmission as well as determining where the packet's next destination is.

Queuing Delay:

the sum of the delays encountered by a packet between the time of insertion into the network and the time of delivery to the address. This term is most often used in reference to routers. When packets arrive at a router, they have to be processed and transmitted.

Queuing Delay=(L*(Q+1)-B*L)/(R)

Where

L=Packet Length, R=Transmission Rate, Q=Packets in queue, B=Current packet in transmission

Jitter:

Jitter is the variation in the periodicity of a signal or periodic event from its target or true frequency. In telecommunications, jitter further refers to the variation in latency of packets carrying voice or video data over a communications channel**.**

Jitter=difference between samples then divide by the number of samples (minus 1).

Network latency:

the time it takes for data or a request to go from the source to the destination. Latency in networks is measured in milliseconds. The closer your latency is to zero, the better

Network latency = $N * (Nth (D_{prop}) + Nth (D_{trans}))$

where,

N - number of intermediate nodes, D_{prop} - propagation delay, D_{trans} - transmission delay

GitHub Link for the project:

https://github.com/shreebuvan/Networks-Case-Study

Conclusion:

The objective of a multicast routing protocol for iMANET is to support the propagation of data from a sender to all the receivers of a multicast group while trying to use the available bandwidth efficiently in the midst of frequent topology changes. Though multicasting can improve the efficiency of the wireless link by exploiting the inherent broadcast property of wireless transmission, it is extremely difficult in iMANET due to its limited resource of bandwidth.

Another important issue to be considered is the limited battery power, Routing protocols in iMANET, establish the path between the source and the destination based on the number of hops. Establishment of the shortest path alone is not sufficient to prolong the network lifetime. Energy consumption reduction techniques are necessary as the nodes in iMANET are limited by battery supply. Energy is drained when the iMANET nodes transmit and receive the data.

Energy 98 consumption in iMANET also depends on the residual battery capacity, distance between the nodes, retransmission and overhearing. As such, energy management techniques are necessary in order to improve the performance of the multicast routing protocol.

Further, QoS support in iMANET is a challenging task due to the dynamic nature of Mobile Ad hoc Networks. The resources must be assigned or reserved in order to achieve a desired QoS. The constraints of MANET, namely, bandwidth, dynamic topology, limited processing and storing capabilities of devices must be concentrated to support QoS in iMANET. QoS in iMANET can be achieved by the QoS model, QoS signaling and QoS routing.