

**PRACTICAL FILE**  
**OF**  
**ADHOC AND WIRELESS SENSOR NETWORK LAB**  
**AT**  
**BABA BANDA SINGH BAHADUR ENGINEERING**  
**COLLEGE**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE  
AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY**

(Computer Science & Engineering)



**SUBMITTED BY:**

PRAVIN KUMAER (2001305)

**SUBMITTED TO:**

PRO. ANKITA PURI

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

BABA BANDA SINGH BAHADUR ENGINEERING COLLEGE,

FATEHGARH SAHIB

<b>S.NO</b>	<b>TITLE</b>	<b>PAGE NO.</b>
1	Introduction to wireless sensor networks and its applications	02-05
2	Network Simulator installation of wireless sensor network	06-07
3	Implementation of routing protocol in NS2 for DSR protocol.	08-09
4	Introduction of wireless network simulators	10-11
5	Implementation of routing protocol in NS2 for AODV protocol for TORA protocol	12-14

# Practical – 1

## AIM: Introduction to wireless sensor networks and its applications

### Wireless Sensor Networks (WSNs)

A Wireless sensor network can be defined as a network of devices that can communicate the information gathered from a monitored field through wireless links. The data is forwarded through multiple nodes, and with a gateway, the data is connected to other networks like wireless Ethernet.

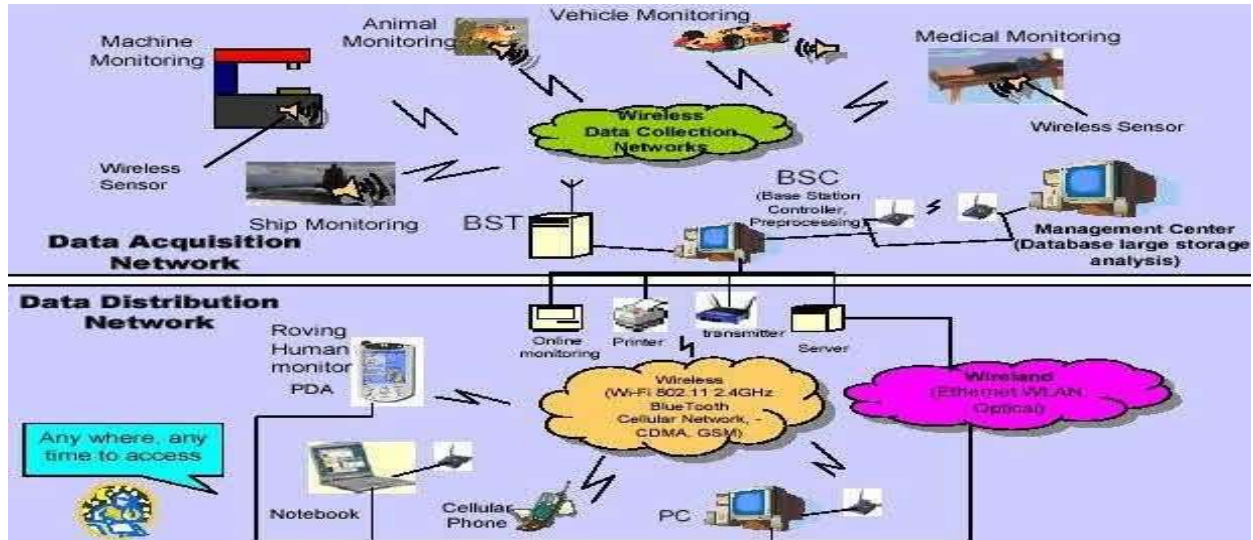


Fig: Wireless Sensor Networks

WSN is a wireless network that consists of base stations and numbers of nodes (wireless sensors). These networks are used to monitor physical or environmental conditions like sound, pressure, temperature, and cooperatively pass data through the network to the main location as shown in the figure.

### Types of Wireless Sensor Networks

Depending on the environment, the types of networks are decided so that those can be deployed underwater, underground, on land, and so on. Different types of WSNs include:

- Terrestrial WSNs
- Underground WSNs
- Underwater WSNs
- Multimedia WSNs
- Mobile WSNs

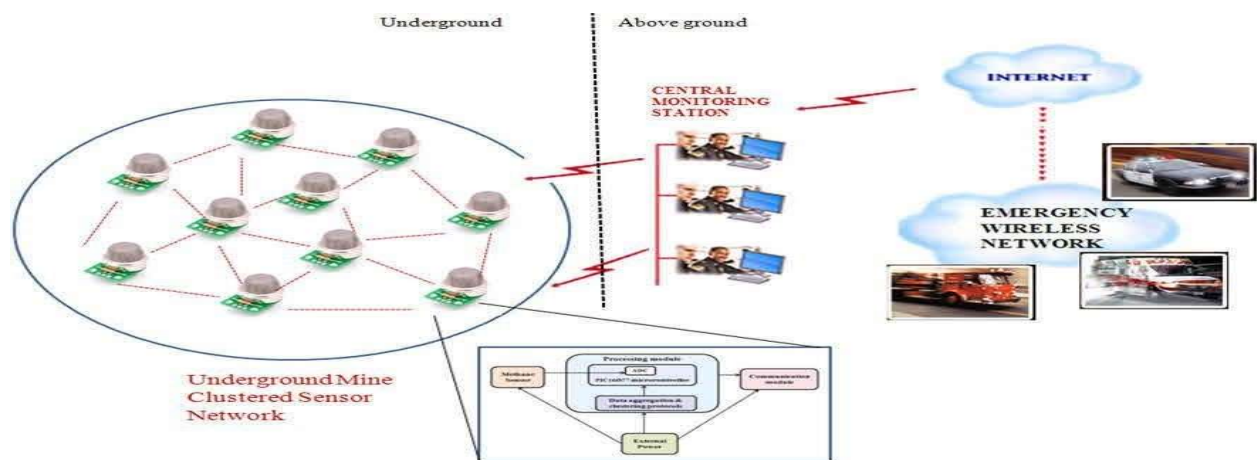
#### • Terrestrial WSNs

Terrestrial WSNs are capable of communicating base stations efficiently, and consist of hundreds to thousands of wireless sensor nodes deployed either in an unstructured (ad hoc) or structured (Pre-planned) manner. In an unstructured mode, the sensor nodes are randomly distributed within the target area that is dropped from a fixed plane. The preplanned or structured mode considers optimal placement, grid placement, and 2D, 3D placement models.

In this WSN, the battery power is limited; however, the battery is equipped with solar cells as a secondary power source. The Energy conservation of these WSNs is achieved by using low duty cycle operations, minimizing delays, and optimal routing, and so on.

#### • Underground WSNs

The underground wireless sensor networks are more expensive than the terrestrial WSNs in terms of deployment, maintenance, and equipment cost considerations and careful planning. The WSNs networks consist of several sensor nodes that are hidden in the ground to monitor underground conditions. To relay information from the sensor nodes to the base station, additional sink nodes are located above the ground.

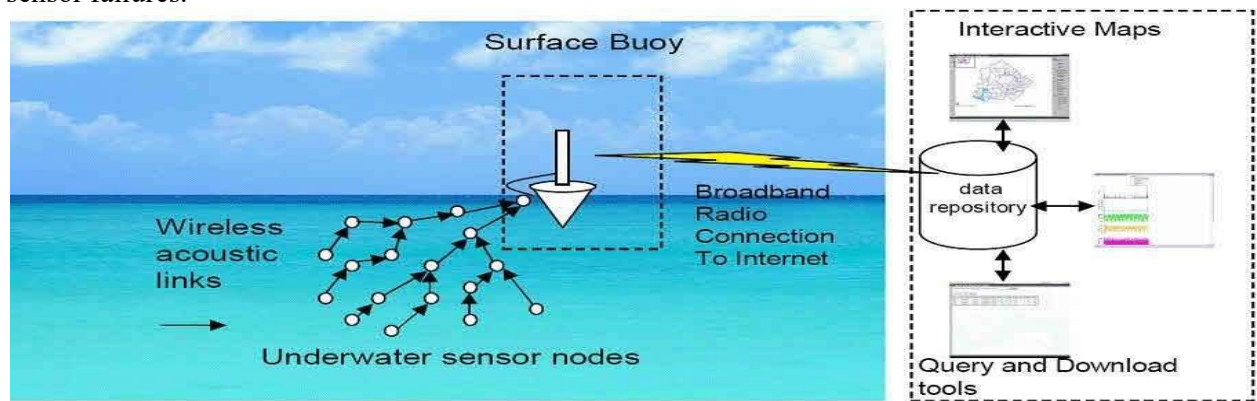


### • Underground WSNs

The underground wireless sensor networks deployed into the ground are difficult to recharge. The sensor battery nodes equipped with limited battery power are difficult to recharge. In addition to this, the underground environment makes wireless communication a challenge due to the high level of attenuation and signal loss.

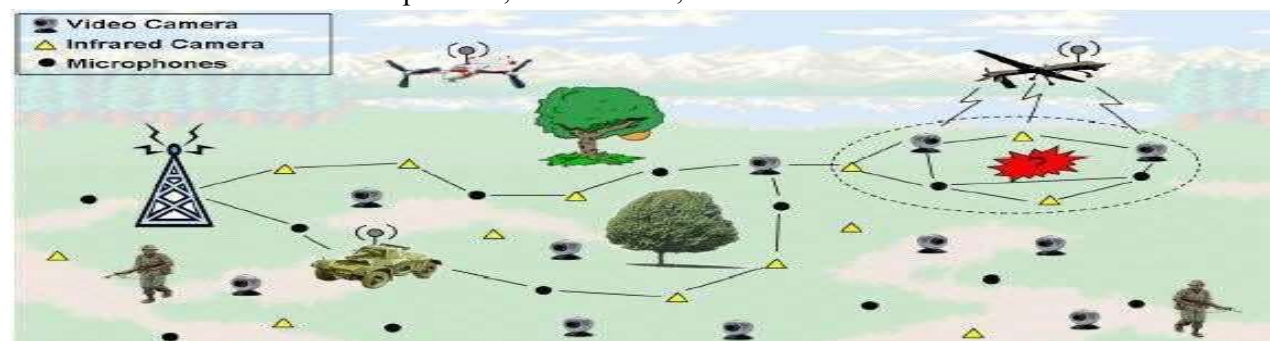
### • Under Water WSNs

More than 70% of the earth is occupied with water. These networks consist of several sensor nodes and vehicles deployed underwater. Autonomous underwater vehicles are used for gathering data from these sensor nodes. A challenge of underwater communication is a long propagation delay, and bandwidth and sensor failures.



### Multimedia WSNs

Multimedia wireless sensor networks have been proposed to enable tracking and monitoring of events in the form of multimedia, such as imaging, video, and audio. These networks consist of low-cost sensor nodes equipped with microphones and cameras. These nodes are interconnected with each other over a wireless connection for data compression, data retrieval, and correlation.



## **Multimedia WSNs**

The challenges with the multimedia WSN include high energy consumption, high bandwidth requirements, data processing, and compressing techniques. In addition to this, multimedia contents require high bandwidth for the content to be delivered properly and easily.

## **Mobile WSNs**

These networks consist of a collection of sensor nodes that can be moved on their own and can be interacted with the physical environment. The mobile nodes can compute sense and communicate.

Mobile wireless sensor networks are much more versatile than static sensor networks. The advantages of MWSN over static wireless sensor networks include better and improved coverage, better energy efficiency, superior channel capacity, and so on.

## **Classification of Wireless Sensor Networks**

The classification of WSNs can be done based on the application but its characteristics mainly change based on the type. Generally, WSNs are classified into different categories like the following.

- Static & Mobile
- Deterministic & Nondeterministic
- Single Base Station & Multi Base Station
- Static Base Station & Mobile Base Station
- Single-hop & Multi-hop WSN
- Self-Reconfigurable & Non-Self Configurable
- Homogeneous & Heterogeneous

### **• Static & Mobile WSN**

All the sensor nodes in several applications can be set without movement so these networks are static WSNs. Especially in some applications like biological systems uses mobile sensor nodes which are called mobile networks. The best example of a mobile network is the monitoring of animals.

### **• Deterministic & Nondeterministic WSN**

In a deterministic type of network, the sensor node arrangement can be fixed and calculated. This sensor node's pre-planned operation can be possible in simply some applications. In most applications, the location of sensor nodes cannot be determined because of the different factors like hostile operating conditions & harsh environment, so these networks are called non-deterministic that need a complex control system.

### **• Single Base Station & Multi Base Station**

In a single base station network, a single base station is used and it can be arranged very close to the region of the sensor node. The interaction between sensor nodes can be done through the base station. In a multi-base station type network, multiple base stations are used & a sensor node is used to move data toward the nearby base station.

### **• Static Base Station & Mobile Base Station**

Base stations are either mobile or static similar to sensor nodes. As the name suggests, the static type base station includes a stable position generally close to the sensing area whereas the mobile base station moves in the region of the sensor so that the sensor nodes load can be balanced.

### **• Single-hop & Multi-hop WSN**

In a single-hop type network, the arrangement of sensor nodes can be done directly toward the base station whereas, in a multi-hop network, both the cluster heads & peer nodes are utilized to transmit the data to reduce the energy consumption.

### ● **Self Reconfigurable & Non-Self Configurable**

In a nonself configurable network, the arrangement of sensor networks cannot be done by them within a network & depends on a control unit for gathering data. In wireless sensor networks, the sensor nodes maintain and organize the network and collaboratively work by using other sensor nodes to accomplish the task.

### ● **Homogeneous and Heterogeneous**

In a homogeneous wireless sensor network, all the sensor nodes mainly include similar energy utilization, storage capabilities & computational power. In the heterogeneous network case, some sensor nodes include high computational power as well as energy necessities as compared to others. The processing & communication tasks are separated consequently.

### **Applications of Wireless Sensor networks**

The applications can be divided in three categories:

- Monitoring of objects.
- Monitoring of an area.
- Monitoring of both area and objects.

#### **Monitoring Objects**

- Structural Monitoring
- Eco-physiology
- Condition-based Maintenance
- Medical Diagnostics
- Urban terrain mapping

#### **Monitoring Area**

- Environmental and Habitat Monitoring
- Precision Agriculture
- Indoor Climate Control
- Military Surveillance
- Treaty Verification
- Intelligent Alarms

#### **Monitoring Interactions between Objects and Space**

- Wildlife Habitats
- Disaster Management
- Emergency Response
- Ubiquitous Computing
- Asset Tracking
- Health Care
- Manufacturing Process Flows

## Practical – 2

### AIM: Network Simulator installation of wireless sensor network NS2

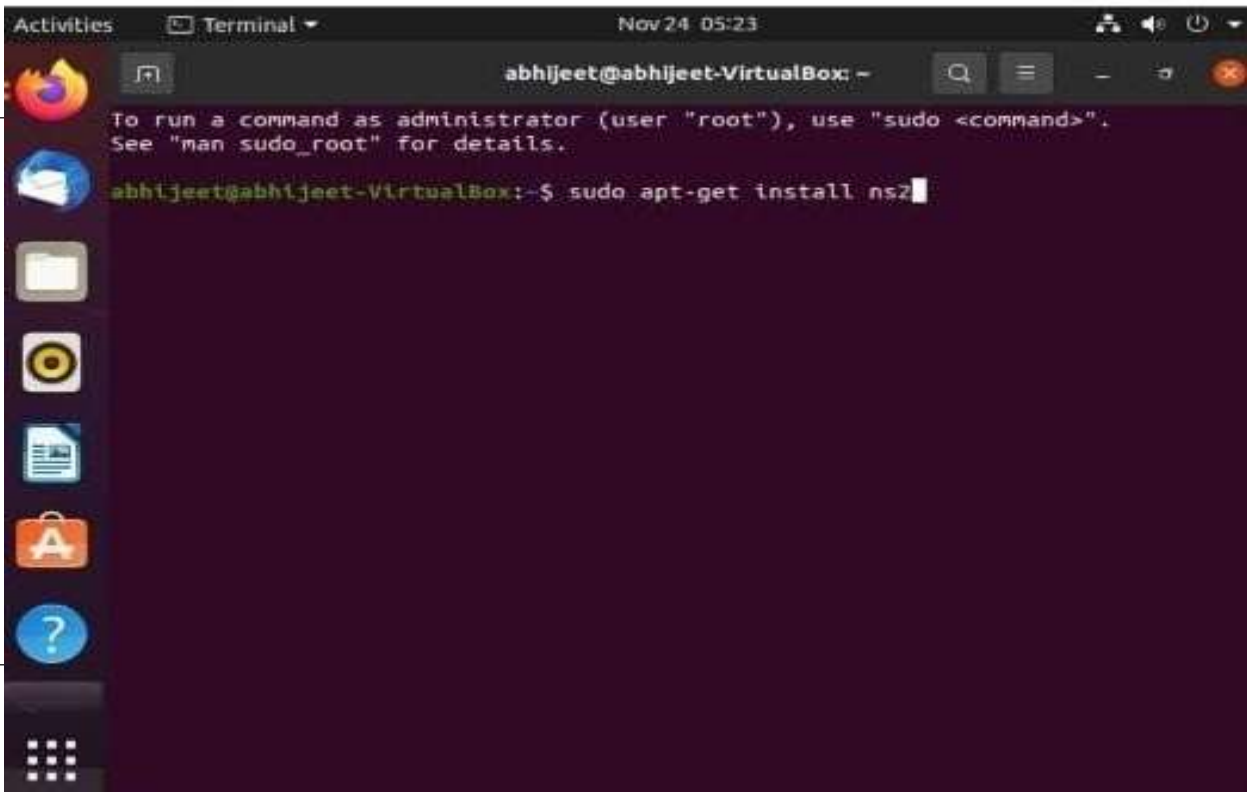
- It is also a discrete event simulator that provides substantial support for simulation of TCP, routing, and also multicast protocols over wired and wireless networks.
- Also, it uses C++ and also OTcl languages.
- Sample ns2 code. In which there are also four different nodes are available and two different routers.

#### Code:

```
set node(s1) [$ns node] set node(s2) [$ns node] set node(r1) [$ns node] set node(r2) [$ns node] set
node(s3) [$ns node] set node(s4) [$ns node]
$ns duplex-link $node(s1) $node(r1) 15Mb 2.5ms DropTail
$ns duplex-link $node(s2) $node(r1) 15Mb 3.2ms DropTail
$ns duplex-link $node(r1) $node(r2) 2.5Mb 22ms RED
$ns queue-limit $node(r1) $node(r2) 28
$ns queue-limit $node(r2) $node(r1) 28
$ns duplex-link $node(s3) $node(r2) 15Mb 4.2ms DropTail
$ns duplex-link $node(s4) $node(r2) 15Mb 5ms DropTail
$ns duplex-link-op $node(s1) $node(r1) orient right-down
$ns duplex-link-op $node(s2) $node(r1) orient right-up
$ns duplex-link-op $node(r1) $node(r2) orient right
$ns duplex-link-op $node(r1) $node(r2) queuePos 0
$ns duplex-link-op $node(r2) $node(r1) queuePos 0
$ns duplex-link-op $node(s3) $node(r2) orient left-down
$ns duplex-link-op $node(s4) $node(r2) orient left-up
set tcp1 [$ns create-connection TCP/Reno $node(s1) TCPSink
$node(s3) 0] $tcp1 set window_ 15 set tcp2 [$ns create-connection TCP/Reno $node(s2) TCPSink
$node(s3) 1] $tcp2 set window_ 15 set ftp1 [$tcp1 attach-source FTP] set ftp2 [$tcp2 attach-source FTP]
$ns at 0.0 "$ftp1 start"
$ns at 3.0 "$ftp2 start"
$ns at 15 "finish"
$ns run
```

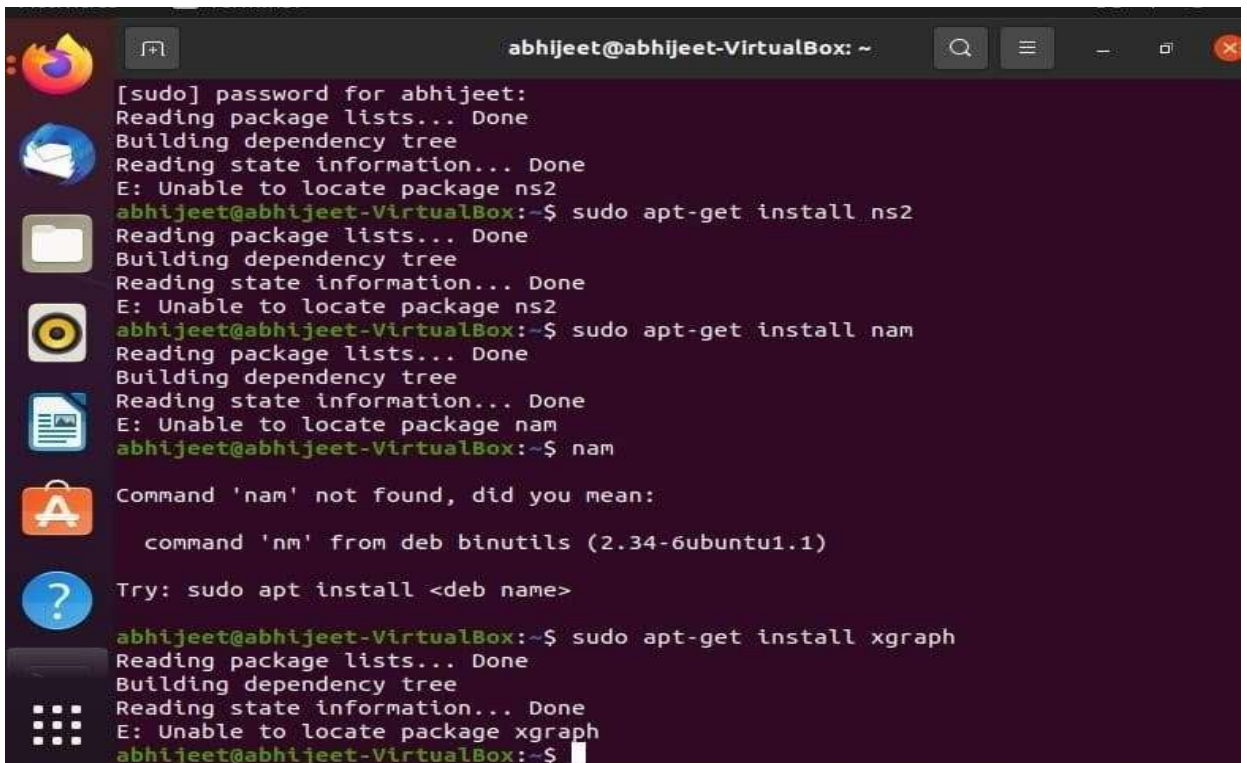


## Screenshots:



A terminal window titled 'abhijeet@abhijeet-VirtualBox: ~' with a search bar and window controls. The terminal displays a message: 'To run a command as administrator (user "root"), use "sudo <command>". See "man sudo\_root" for details.' Below this, the command 'abhijeet@abhijeet-VirtualBox:~\$ sudo apt-get install ns2' is entered and the cursor is at the end of the line.

```
abhijeet@abhijeet-VirtualBox:~$ sudo apt-get install ns2
```



A terminal window titled 'abhijeet@abhijeet-VirtualBox: ~' showing the output of several commands. The first command 'sudo apt-get install ns2' results in 'E: Unable to locate package ns2'. The second command 'sudo apt-get install nam' results in 'E: Unable to locate package nam'. The third command 'nam' results in 'Command 'nam' not found, did you mean: command 'nm' from deb binutils (2.34-6ubuntu1.1)'. The fourth command 'sudo apt-get install xgraph' results in 'E: Unable to locate package xgraph'.

```
[sudo] password for abhijeet:
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package ns2
abhijeet@abhijeet-VirtualBox:~$ sudo apt-get install ns2
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package ns2
abhijeet@abhijeet-VirtualBox:~$ sudo apt-get install nam
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package nam
abhijeet@abhijeet-VirtualBox:~$ nam
Command 'nam' not found, did you mean:
  command 'nm' from deb binutils (2.34-6ubuntu1.1)
Try: sudo apt install <deb name>
abhijeet@abhijeet-VirtualBox:~$ sudo apt-get install xgraph
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package xgraph
abhijeet@abhijeet-VirtualBox:~$
```



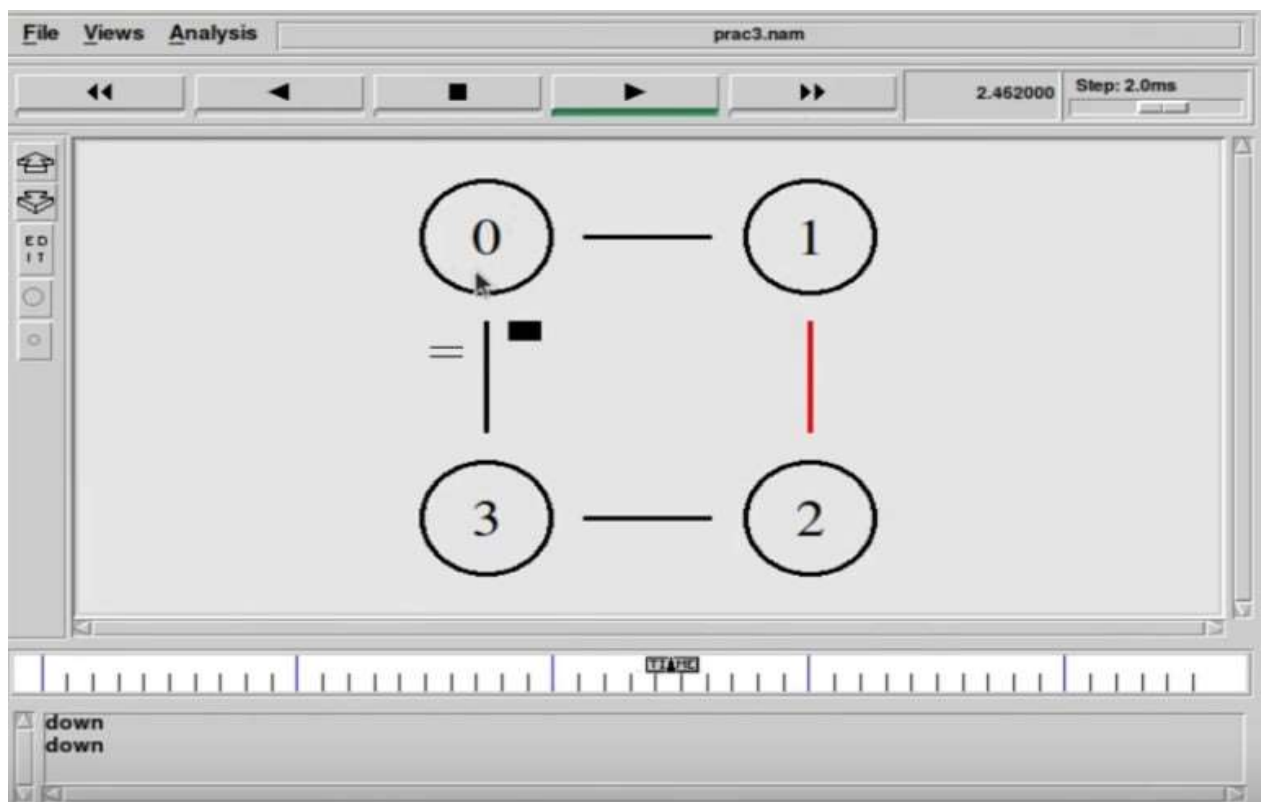
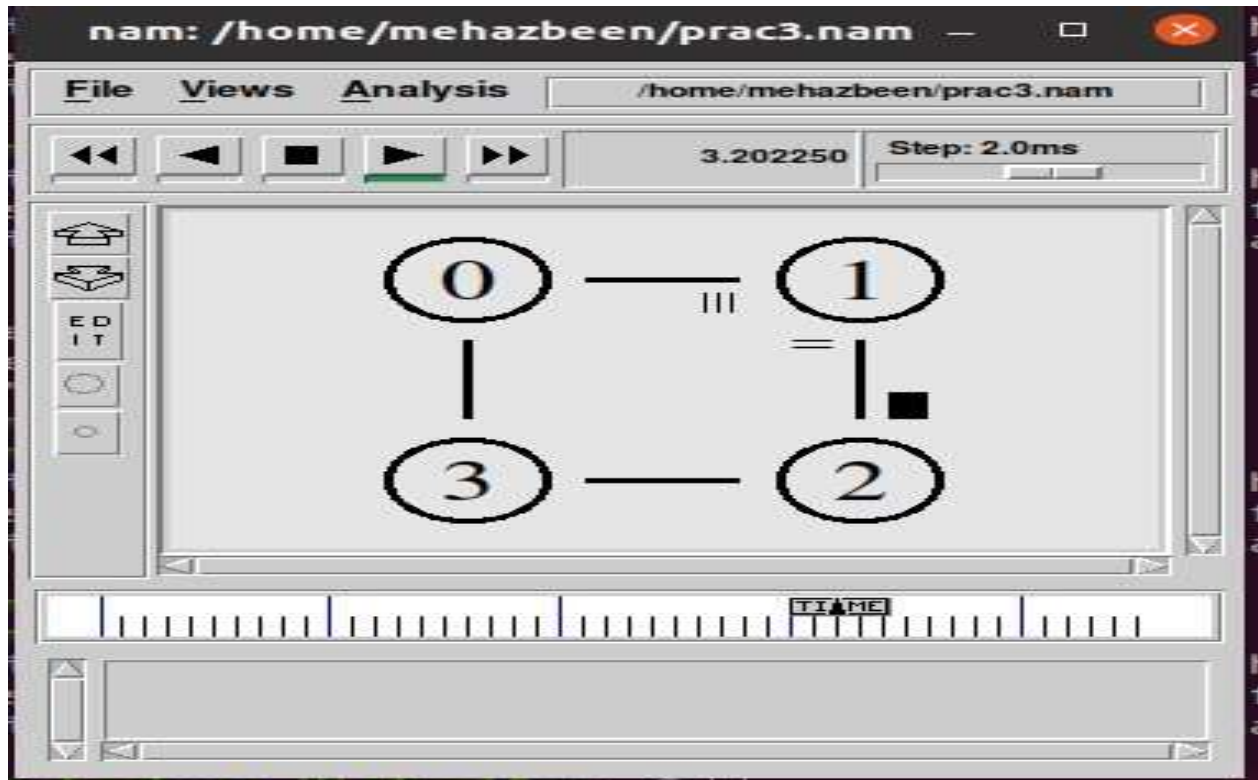
## Practical – 3

**AIM: Implementation of routing protocol in NS2 for DSR protocol.**

**Code:**

```
Set ns [new simulator]
Set nf[open prac.nam w]
$ ns nam trace.all $nf
Proc finish(){
Global ns nf
$ns flush.trace
Close $nf
Execu nam prac3.nam&
Exit
Set n0[$ns node]
Set n1[$ns node]
Set n2[$ns node]
Set n3[$ns node]
$ns duplex.link $n0 $n1 10Mb 10ms DropTail
$ns duplex.link $n0 $n2 10Mb 10ms DropTail
$ns duplex.link $n0 $n3 10Mb 10ms DropTail
$ns duplex.link $n0 $n4 10Mb 10ms DropTail
$ns duplex link.op $n0$n1 orient right
$ns duplex link.op $n0$n1 orient down
$ns duplex link.op $n0$n1 orient left
$ns duplex link.op $n0$n1 orient up
Set tcp[new Agent/TCP]
$tcp set class_2
$ns attach.agent $n0 $tcp
Set sink[new Agent/TCP sink]
$ns attach.agent$ns$sink
Set ftp[new Application/FTP]
$ftp attach.agent$tcp
$ftp set type_FTP
$ftp set packet_size_1000
$ftp set rate_imb
$ns at 1.0"$ftp start"
$ns at 4.0"$ftp start"
$ns at 5.0"$ftp start"
$ns run
```

## Screenshots:



## Practical – 4

### AIM: Introduction of wireless network simulators

- Ns2 (Network Simulator 2).
- Ns3 (Network Simulator 3).
- OPNET.
- OMNeT++.
- QualNet.
- J-Sim.

### NS2 (Network Simulator 2)

It is also a discrete event simulator that provides substantial support for simulation of TCP, routing, and also multicast protocols over wired and wireless networks.

Also, it uses C++ and also OTcl languages. Sample ns2 code. In which there are also four different nodes are available and two different routers.

### NS3 (Network Simulator 3):

Ns3 uses C++ and also Python languages for simulating the script. C++ used for implementation of simulation and also core model. Ns-3 is built as a library which may be statically or dynamically linked to a C++ main program. Python: C++ wrapped by Python. Python programs also to import an “ns3” module

### OPNET

It provides a comprehensive development environment supporting the modelling of communication networks and also distributed systems. Both behaviour and performance of modelled systems can also be analysed by performing discrete event simulations’ is a main programming language in OPNET and also use GUI for initial configurations. The simulation scenario requires c or C++

### OMNeT++:

It is a component-based, modular and also open architecture discrete event simulator framework. The most common use of OMNeT++ is also for simulation of computer networks, but it is also used for queuing network simulations and other areas as well. C++ is a class library, eclipse-based simulation IDE is also used for designing, running and evaluating simulations.

### QualNet

It is also a commercial network simulator from Scalable Network Technologies. Also, it is ultra high-fidelity network simulation software that predicts wireless, wired and mixed-platform network and also networking device performance. A simulator for large, heterogeneous networks and also the distributed applications that execute on such networks It use C++ for implementing new protocols and also follows a procedural paradigm.

### JSIM:

It’s also a java-based simulator tool. Java is easy to learn and easy to use. In case of any problems, source texts provided with J-Sim can be used to generate new code, compiled in the target environment, thus 100-percent compatible with JVM used Use java and Tcl language

### OMNeT++:

It is a component-based, modular and also open architecture discrete event simulator framework. The most common use of OMNeT++ is also for simulation of computer networks, but it is also used for queuing network simulations and other areas as well. C++ is a class library, eclipse-based simulation IDE is also used for designing, running and evaluating simulations.

**NetSim**

It has an object-oriented system modelling and simulation (M&S) environment to support simulation and analysis of voice and data communication scenarios for High Frequency Global Communication Systems (HFGCS). NetSim use java as a programming language it creates applet and linked into HTML document for viewable on the java-compatible browser.

**REAL**

REAL is a simulator for studying the dynamic behaviour of flow and congestion control schemes in packet switch data networks. It provides users with a way of specifying such networks and to observe their behaviour. REAL uses C as a programming language. Sample code for REAL simulator.

## Practical – 5

### **AIM: Implementation of routing protocol in NS2 for AODV protocol for TORA protocol**

#### **TORA:**

TORA is a protocol in wireless Adhoc networks that works with timing parameters. NS-2.35 comes with the TORA protocol by default but it has to be tweaked manually to make it run.

#### **AODV**

The knowledge categories represented by AODV are Route Requests (RREQs), Route Replies (RREPs), and Route Errors (RERRs). Methods for UDP and general IP header dealing with administrators are used to obtain these data categories. As an example, the referencing center point would need to use its IP address as the information's Originator IP address. The IP specified transmission address (255.255.255.255) is used for transmission information. This implies that identical data is not sent at random. In either case, AODV movement necessitates widespread dissemination of self-assured knowledge (e.g., RREQ), likely via offhand association. The TTL in the IP header shows how far similar RREQs have spread. Irregularity isn't always a requirement. Anyway, expanded the end point of a correspondence affiliation have authentic courses to another, AODV doesn't expect part of work. Exactly when a course to another goal is required, the center point imparts a RREQ found a course to the target. A course can be settled when the RREQ comes to either the goal itself, or a midway center point with 'sufficiently another' course to the goal.

#### **Implementation:**

##### **Step 1:**

Generate a Scenario for TORA protocol using NS2 Scenario Generator NSG Software.

We have created a Tcl file using NSG2.1.jar

```
$] java -jar NSG2.1.jar
```

Three files have to be modified

~ns-2.35/tora/tora.cc

~ns-2.35/tora/tora.h

~ns-2.35/imep/imep.cc

There are various websites that tells you how to configure TORA by making changes to the above three files.

##### **Change 1: tora.h**

In the tora.h file, go to the end of the File before the agent completes, include these two lines

```
#include <classifier/classifier-port.h> protected:
```

```
PortClassifier *dmux_; tora.h
```

```

tora.h (~ns-allinone-2.35/ns-2.35/tora) - gedit
void log_link_layer_feedback(Packet *p);
void log_link_layer_recycle(Packet *p);
void log_lnk_del(nsaddr_t dst);
void log_lnk_kept(nsaddr_t dst);

void log_nb_del(nsaddr_t dst, nsaddr_t id);
void log_rcv_gry(Packet *p);
void log_rcv_upd(Packet *p);
void log_rcv_clr(Packet *p);
void log_route_table(void);

void log_dst_state_change(TORADest *td);

void logNextHopChange(TORADest *td);
void logNbDeletedLastDN(TORADest *td);
void logToraDest(TORADest *td);
void logToraNeighbor(TORANeighbor *tn);

protected:
    PortClassifier *dmux_;
};

#endif /* __tora_h__ */

```

## Tora.h Change

### Change 2: tora.cc

Open the tora.cc and include the following lines in the "int toraAgent::command(int argc, const char\*const\* argv) " function as indicated in the figure below.

```

else if (strcmp(argv[1], "port-dmux") == 0) { dmux_ = (PortClassifier *)TclObject::lookup(argv[2]); if
(dmux_ == 0) { fprintf(stderr, "%s: %s lookup of %s failed\n", __FILE__, argv[1], argv[2]); return
TCL_ERROR;
} return TCL_OK;
}

```

Tora

```

tora.cc (~ns-allinone-2.35/ns-2.35/tora) - gedit
else if (strcmp(argv[1], "port-dmux") == 0) {
    imepagent = (imepAgent*) TclObject::lookup(argv[2]);
    if (imepagent == 0)
        return TCL_ERROR;
    imepagent->imepRegister((rtAgent*) this);
    return TCL_OK;
}

else if (strcmp(argv[1], "port-dmux") == 0) {
    dmux_ = (PortClassifier *)TclObject::lookup(argv[2]);
    if (dmux_ == 0) {
        fprintf(stderr, "%s: %s lookup of %s failed\n", __FILE__, argv[1], argv[2]);
        return TCL_ERROR;
    }
    return TCL_OK;
}

return Agent::command(argc, argv);
}

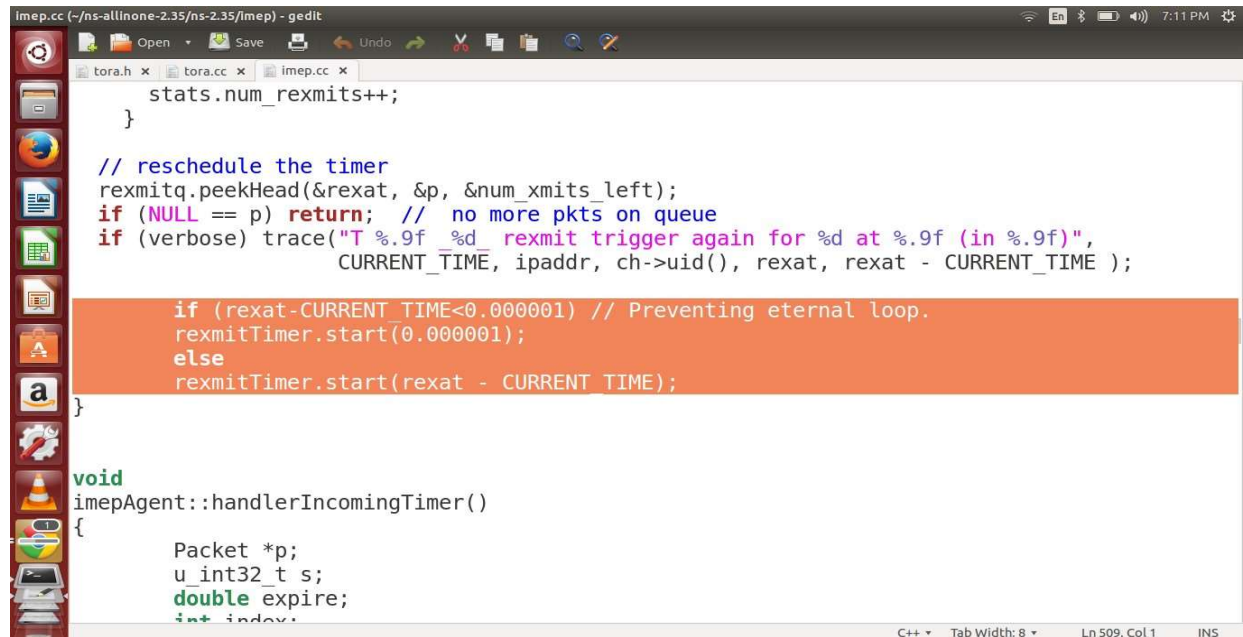
/* =====
Destination Management Functions
===== */

```

## Tora.cc Change

### Change 3: imep.cc

In the file imep.cc, change the following line `rexmitTimer.start(rexat - CURRENT_TIME);` to `if (rexat-CURRENT_TIME<0.000001) // Preventing eternal loop.`  
`rexmitTimer.start(0.000001); else rexmitTimer.start(rexat - CURRENT_TIME); imep`



```
stats.num_rexmits++;
}

// reschedule the timer
rexmitq.peekHead(&rexat, &p, &num_xmits_left);
if (NULL == p) return; // no more pkts on queue
if (verbose) trace("T %.9f %d rexmit trigger again for %d at %.9f (in %.9f)",
    CURRENT_TIME, ipaddr, ch->uid(), rexat, rexat - CURRENT_TIME );

if (rexat-CURRENT_TIME<0.000001) // Preventing eternal loop.
    rexmitTimer.start(0.000001);
else
    rexmitTimer.start(rexat - CURRENT_TIME);
}

void
imepAgent::handlerIncomingTimer()
{
    Packet *p;
    u_int32_t s;
    double expire;
    int index;
```

IMEP Change

### Step 2:

We need to recompile ns2 using the command `make` from the folder `ns-2.35/`

Once the changes are made, Open the Terminal and go to `~ns-2.35` and execute the command `prompt$] make`

### Step 3:

Run the tcl file now

`$] ns TORA.tcl`

This file is generated in Step 1. Copy the `tora.cc` and `tora.h` file in your `ns-2.35/tora` folder then copy the `imep.cc` file in to the `ns-2.35/imep` folder and then copy the `TORA.tcl` file in your home folder or any folder. Then do the compilation `$] make` and then run the `TORA.tcl` file using `$] ns TORA.tcl`

