NAME : Aachal Mishra
Roll No : 03 (A)

Ex. No: 1:

Study of Network Simulator Tools. Date :07/02/23

Aim:

To study of different Network Simulation Tools for constructing and performance testing in different network.

Introduction:

Simulation is a very important modern technology. It can be applied to different science, engineering, or other application fields for different purposes. Computer assisted simulation can model hypothetical and real-life objects or activities on a computer so that it can be studied to see how the system function. Different variables can be used to predict the behavior of the system. Computer simulation can be used to assist the modeling and analysis in many natural systems. Typical application areas include physics, chemistry, biology, and human-involved systems in economics, finance or even social science. Other important applications are in the engineering such as civil engineering, structural engineering, mechanical engineering, and computer engineering. Application of simulation technology into networking area such as network traffic simulation.

Basic concepts in network simulation

In the area of computer and communications networks, simulation is a useful technique since the behavior of a network can be modeled by calculating the interaction between the different network components (they can be end-host or network entities such as routers, physical links or packets) using mathematical formulas. They can also be modeled by actually or virtually capturing and playing back experimental observations from a real production networks. After we get the observation data from simulation experiments, the behavior of the network and protocols supported can then be observed and analyzed in a series of offline test experiments. All kinds of environmental attributes can also be modified in a controlled manner to assess how the network can behave under different parameters combinations or different configuration conditions. Another characteristic of network simulation that worth noticing is that the simulation program can be used together with different applications and services in order to observe end-to-end or other point-to-point performance in the networks.

Type of network simulators

For network protocol designers, it is often difficult to decide which simulator to choose for a particular task. Therefore, we conduct a survey to find a network simulator that provides a good balance between availability of ready to use models, scripting and language support, extendibility, graphical support, easiness of use, etc. The survey is based on a collection of a number of criteria including published results, interesting characteristics and features. From our survey results, we broadly categories network simulators as: "Widely Used" simulators and "Other" simulators. The network simulators taken into consideration as "Widely Used" are Ns-2, Ns-3, GloMoSim, J-Sim, OMNet++, OPNet, and QualNet.

1. **OPNET (Optimized Network Evaluation Tool):**

OPNET 's software environment is called Modeler, which is specialized for network research and development. It can be flexibly used to study communication networks, devices, protocols, and applications. Because of the fact of being a commercial software provider, OPNET offers relatively much powerful visual or graphical support for the users. The

graphical editor interface can be used to build network topology and entities from the application layer to the physical layer. Object-oriented programming technique is used to create the mapping from the graphical design to the implementation of the real systems. An example of the graphical GUI of OPNET can be seen in Figure 1. We can see all the topology configuration and simulation results can be presented very intuitively and visually. The parameters can also be adjusted and the experiments can be repeated easily through easy operation through the GUI.

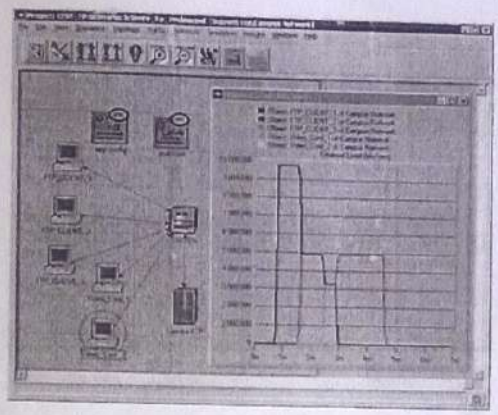


Figure 1. OPNET GUI

Main features:

OPNET inherently has three main functions: modeling, simulating and analysis. **For modeling**, it provides intuitive graphical environment to create all kinds of models of protocols. **For simulating**, it uses three different advanced simulations technologies and can be used to address a wide range of studies. **For analysis**, the simulation results and data can be analyzed and displayed very easily. User friendly graphs, charts, statistics, and even animation can be generated by OPNET for users ' convenience.

2. Network Simulator 2 (NS2)

NS2 is one of the most popular open source network simulators. The original NS is a discrete event simulator targeted at networking research. NS2 is the second version of NS (Network Simulator). NS is originally based on REAL network simulator. The first version of NS was developed in 1989 and evolved a lot over the past few years. The current NS project is supported through DARPA. The current second version NS2 is widely used in academic research and it has a lot of packages contributed by different non-benefit groups. An example of the graphical GUI of NS2 can be seen in Figure 2.

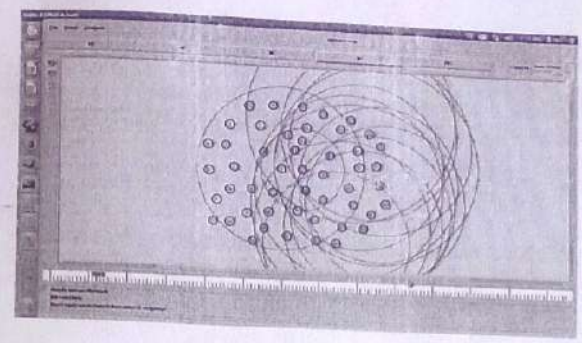


Figure 2. NS2 GUI

Main features:

First and foremost, NS2 is an object-oriented, discrete event driven network simulator which was originally developed at University of California-Berkely. The programming it uses is C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT). The usage of these two programming language has its reason. The biggest reason is due to the internal characteristics of these two languages. C++ is efficient to implement a design but it is not very easy to be visual and graphically shown. It's not easy to modify and assembly different components and to change different parameters without a very visual and easy-to-use descriptive language. Moreover, f or efficiency reason, NS2 separates control path implementations from the data path implementation. The event scheduler and the basic network component objects in the data path are written and compiled using C++ to reduce packet and event processing time. OTcl happens to have the feature that C++ lacks. So the combination of these two languages proves to be very effective. C++ is used to implement the detailed protocol and OTcl is used for users to control the simulation scenario and schedule the events. A simplified user's view of NS2 is shown in Figure 3. The OTcl script is used to initiate the event scheduler, set up the network topology, and tell traffic source when to start and stop sending packets through event scheduler. The scenes can be changed easily by programming in the OTcl script. When a user wants to make a new network object, he can either write the new object or assemble a compound object from the existing object library, and plumb the data path through the object. This plumbing makes NS2 very powerful.

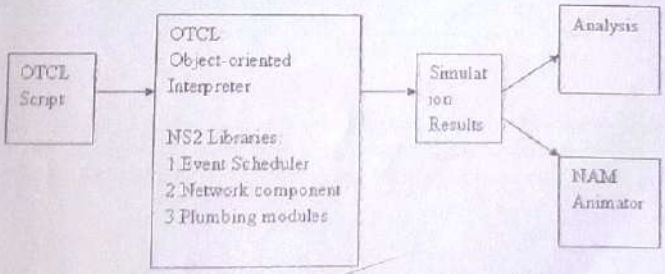


Figure 3. Simplified User's View of NS2

3. **Network Simulator 3 (NS3)**

Similar to NS2, NS3 is also an open soureed discrete-event network simulator which targets primarily for research and educational use. NS3 is licensed under the GNU GPLv2 license, and is available for research and development. NS3 is designed to replace the current popular NS2. However, NS3 is not an updated version of NS2 since that NS3 is a new simulator and it is not backward-compatible with NS2.

4. **OMNeT++**

OMNeT++ has generic and flexible architecture which makes it successful also in other areas like the IT systems, queuing networks, hardware architectures, or even business processes as well. It is similar with NS2 and NS3, OMNeT++ is also a public-source, component-based network simulator with GUI support. Its primary application area is communication networks. Like NS2 and NS3, OMNeT++ is also a discrete event simulator. It is a component-based architecture. Components are also called modules and are programmed in C++. The components are then assembled into larger components and models by using a high-level language. Its function is similar to that of OTcl in NS2 and Python in NS3. OMNeT++ also provides GUI support, and due to its modular architecture, the simulation kernel can be embedded into all kinds of different user s' applications. Figure 5 is an OMNeT++ GUI screenshot.

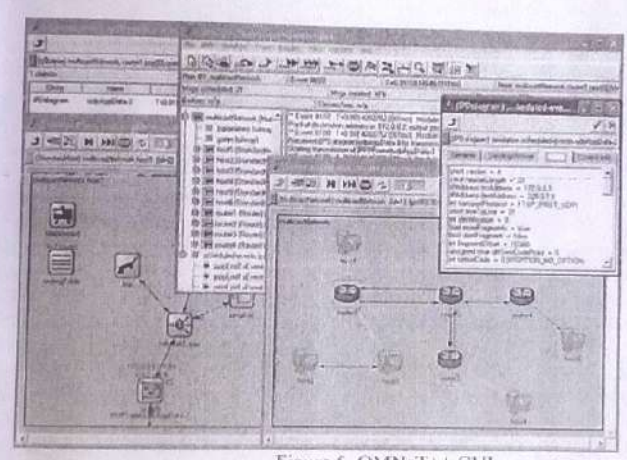


Figure 5. OMNeT++ GUI

Main features:

Since OMNeT++ is designed to provide a component-based architecture, the models or modules of OMNeT++ are assembled from reusable components. Modules are reusable and can be combined in various ways which is one of the main features of OMNeT++.

5. **Global Mobile Information System Simulator (Glo-MoSim)**

It is a parallel discrete event simulation soft-ware[4] that simulates wireless and wired network sys-tems. It is designed as a set of library modules, each of which simulates a specific wireless communication pro-tocol in the protocol stack. It assumes that the network is decomposed into a number of partitions and a single entity is defined to simulate a single layer of the com-plete protocol stack for all the network nodes that be-long to the partition. The parallel implementation of GloMoSim can be executed using variety of conservative synchronization protocols, which include the null mes-sage and conditional event algorithm. The library has been developed using PARSEC, a C based parallel simulation language. It uses the Parsec compiler to com-pile the simulation protocols. It has been designed to be extensible and comprehensible. GloMoSim aims to de-velop a modular simulation environment for protocol stack that is capable of scaling up networks with thou-sands of heterogeneous nodes. GloMoSim currently supports protocols for a purely wireless network.

Features:

- GloMoSim is a library-based sequential and parallel simulator for wireless networks.
- GloMoSim facilitates the ability to use in a parallel environment which distinguishes it from most other wireless network simulators.
- It allows the simulation Scalability to simulate net-works with a hundred and thousand of nodes.
- It supports various layers like Mobility, Radio Propagation, Radio Model, Packet reception models, Data Link, Network (Routing), Transport and Ap-plication. i.e. (It supports almost all the OSI layers with limited benefits).
- GloMoSim supports direct satellite communication, multi-hop wireless communication and most of the traditional internet protocols.
- It facilitates to build a library of parallelized mod-els that can be used for the evaluation of a varie-ty of wireless network protocols.

6. **NetSim**

NetSim is a stochastic discrete event network simula-tiontool used for network lab experimentation and research. Its leading network simulation software for protocol modeling and simulation, allowing us to ana-lyze computer networks with unmatched depth, power and flexibility. NetSim comes with an in-built develop-ment environment, which serves as the interface be-tween User's code and NetSim's protocol libraries and simulation kernel. It provides network performance metrics at various abstraction levels such as Network, sub-network, Node and a detailed packet trace. It has unique features and functionality. NetSimis available as Standard or Academic versions and is built on a com-mon design framework of high level architecture and code. In a word, NetSim is truly a fantastic product that is not only versatile, but also robust and provides those features that are hard to come with any simulators.

Features:

- NetSim modeling and simulation are supported for Aloha, Slotted Aloha, Token Ring/Bus, Ethernet CSMA/CD, Fast and Gigabit Ethernet, WLAN - IEEE 802.11 a/b/g/n and e, X.25, Frame Relay, TCP, UDP, IPv4 and IPv6, Routing - RIP, OSPF, BGP,MPLS, Wi-Max, MANET, GSM, CDMA, Wire-less Sensor Network, Zigbee, Cognitive radio.
- It simulates a wide variety of Cisco routers, including 2500 series, 2600 series, 2800 series, and 3600 series routers, as well as the Cisco Catalyst 1900 series, 2900 series, and 3500 series switches.

- Protocol libraries are available as open C code for user modification. This can help avoid the time consuming process of programming, customization and configuration commercial simulators to meet customer specific needs.
- Along with the Boson Virtual Packet Technology engine NetSim utilizes Boson's proprietary Network Simulator, Router Simulator, and EROUTER soft-ware technologies, to create individual packets. These packets are routed and switched through the simulated network, allowing NetSim to build an appropriate virtual routing table and simulate true networking. Other simulation products on the mar-ket do not support this level of functionality.
- It can be used to create a simulation of the topology of corporate network and help practice trouble-shooting without using devices on the production network.

Comparison of simulators based on General Information

Sl. #	Name	License Type	Language	Supported Operating	GUI port
1	Ns2	Open source	C++ and OTCL	GNU/Linux,FreeBSD, Mac OS X, Windows XP, Windows Vista and Win. 7.	Limited
2	Ns3	Open source	C++and Optional Python Bindings	GNU/Linux,FreeBSD, Mac OS X, Windows XP, Windows Vista and Win. 7.	Yes
3	QualNet	Commercial (Separate license for academicians and others)	C++	UNIX, Window, MAC, Linux	Yes
4	GloMoSim	Open source	C	Windows, Linux, Sun SPARC Solaris	Limited
5	NetSim	Proprietary	C and Java	Windows (7, Vista) and windows XP	Yes
6	OMNET++	Open source (for study and research	C++	Windows XP or Later, Linux, Mac OS X,	Yes

(7)

Comparison of simulators based on the properties of simulators

Sl. #	Name	Simulation Event Type	Available Module	Scalability	Number of node support	Parallelism
1	NS2	Discrete-event	Wired, Wireless, AdHoc and Wireless Sensor Networks	Limited	Up to 3000	No
2	NS3	Discrete-event	Wired, Wireless, Adhoc and Wireless Sensor Networks	Limited	Up to 3000	No
3	QualNet	Discrete- event	Wired &Wireless(like WiFi, Sensor network,MANET,WIMAX)network	Large	500-20000	Yes(SMP/Beowulf
4	GloMoSim	Discrete-event	Wired, Wireless & Ad-Hoc Networks. But currently pure wireless support	Large	Up to 10,000	Yes(SMP/Beowulf
5	NetSim	Stochastic Discrete-	Wired &Wireless, sensor network Event	Large Enough	———	———
6	OMNET++	Discrete-(wireless LAN, WiMAX)	Wired, Wireless, Ad-hoc and WSN.	Enough	———	MPI/P

Result:

Various Network Simulators has been successfully studied.

Done
21/3/23

(15)

Name : Aakal Mishra

Roll NO : 03

Section : A

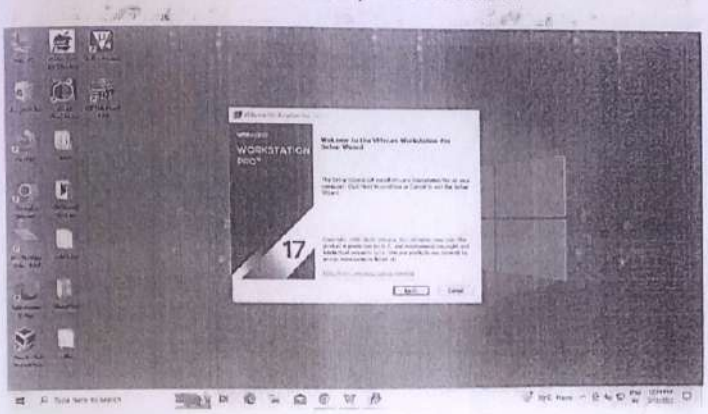
(3) Date : 1/2/23

Experiment No: 2

Aim: To Install Network Simulator 2

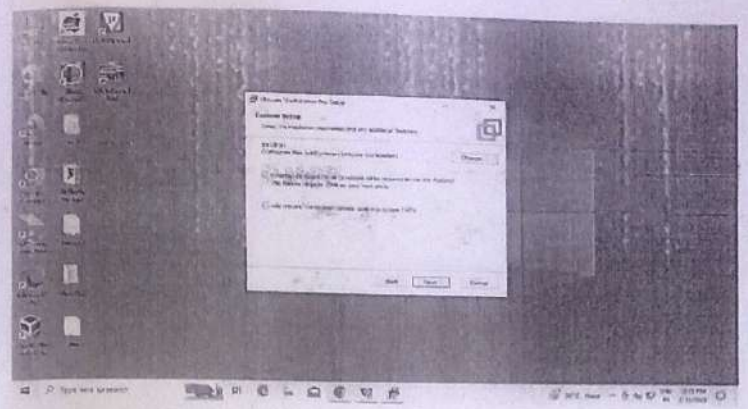
Procedure:

- Click on VMware Workstation Pro Setup Wizard to install it.

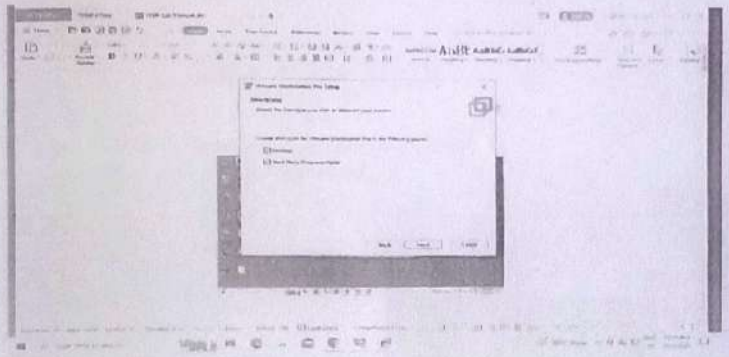


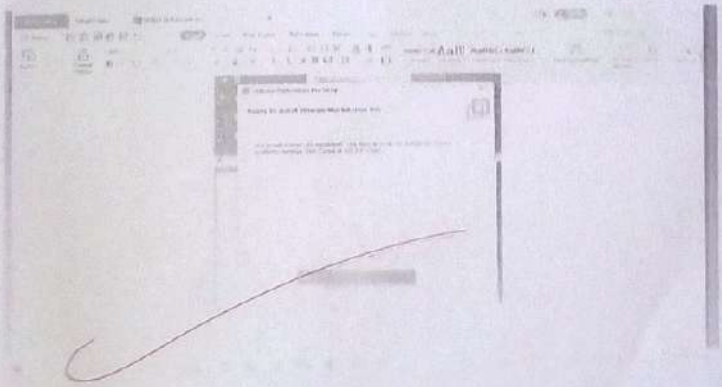
- Then click on the next .

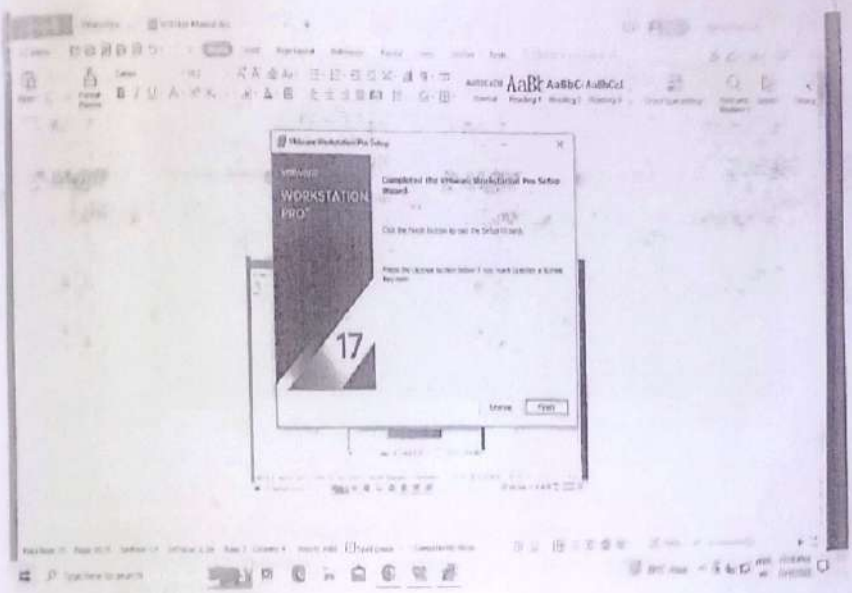
- Then accept the term and conditions , & click on next .



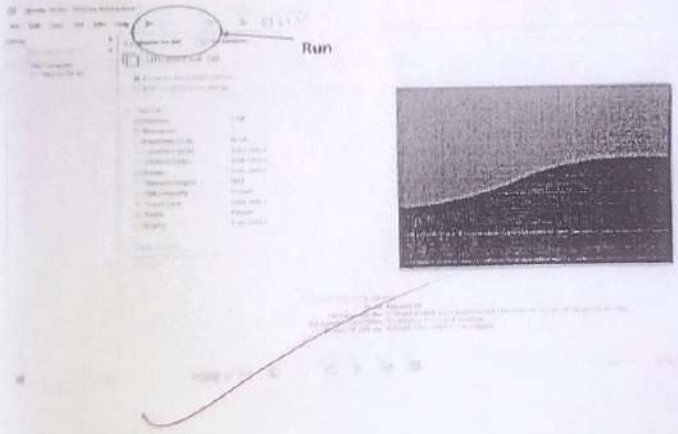
- Allow all the appear conditions &click on next.

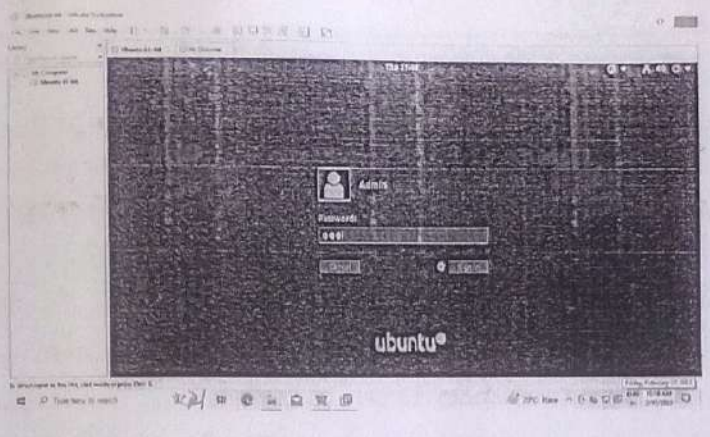






- Click on VM Workstation Pro (icon on the Desktop)
- Click on Run option to start Virtual OS





- Enter password
- Right click on the Ubantu desktop and click on "Open Terminal"
- Type the following commands:

```
sudo apt-get update
Enter pwd:
sudo apt-get install ns2
sudo apt-get install nam
sudo apt-get install tcl
sudo apt-get install gedit
```

- To open the editor window type "gedit name_of_exp.tcl"
- To simulate the program type "ns name_of_exp.tcl"

14

Prog
21/2/23

Date: 2/3/23

Experiment No.3

Aim :- To write TCL script for establishing connection between mobile nodes.

Objective :- To understand the process of creating nodes and connecting link.

Theory :- Network simulation is an important tool in developing, testing and evaluating network protocols. Simulation can be used without the target physical hardware, making it economical and practical for almost any scale of network topology and setup. It is possible to simulate a link of any bandwidth and delay, even if such a link is currently impossible in the real world. With simulation, it is possible to setcach simulated node to use any desired software. This means that meaning deploying software is noton issue. Results are also easier to obtain and analyze, because ~~creating~~ information from important points in the simulated network is as done by simply parsing the generated trace files.

1) Set ns [new Simulator] :- Generates an NS simulator object instance, and assigns it to variable ns.
2) Initializes the packet format
2) Creates a scheduler
3) Selects the default address format.

The "simulator" objects has member functions that do the following
1) Create compound objects such as nodes and links (described
2) Connect n/w components objects created (ex, attach-agent).
3) Set n/w component parameters Create connections between agents.
4) Specify NAM display options.

2) $ns color fid color :→ For eg: $ $ns color 1 Blue
It is to set color of the packets for a flow specified by the flow id (fid). This member functions of "simulator" objects is for the NAM display, and has no effect on the actual simulation.

3) $ns namtrace-all file-descriptor : This member function tells the simulator to record simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command $ns flush-trace. Similarly, the member fun" trace-all is for recording the simulation trace in a general format.

4) Proc finish { } : is called after this simulation is over by the command $ns at 5.0 "finish". In this function, post-simulation processes are specified.

5) Set n0 [$ns node] : for eg: set n1 [$ns node]
The member function node creates a node. A node in NS is Compound object made of address and port classifiers.

6] $ns duplex-link node1 node2 bandwidth delay queue-type :
For eg: $ns duplex-link $n1 $n2 2Mb 10ms DropTail.
It creates two simplex links of specified bandwidth and delay and connects the two specified nodes. In NS, the output queue of a node is implemented as a part of a link, these base uses should specify the queue-type when creating links. In the above simulation script, DropTail queue is used. If the reader wants to use a RED queue, simply replace the word DropTail with RED.

7] $ns queue-limit node1 node2 number : eg. $ns queue-limit $n1 $n2 10
This line sets the queue limit of the two simplex links that connect node 1 & node 2 to the number specified.

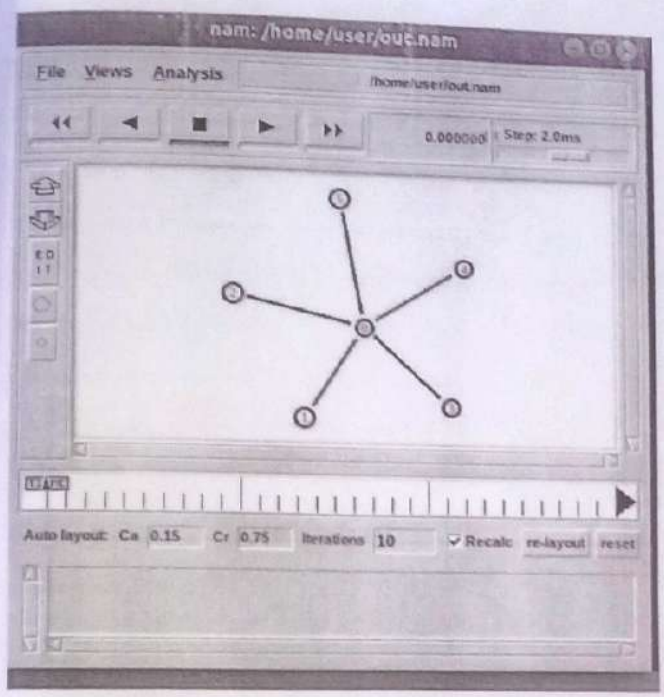
8] $ns duplex-link-op node1 node2 ... : The next couple of lines are used for the NAM display. To see the effects of these lines, users can comment these lines out and try the simulation.
eg: $ns duplex-link-op $n1 $n2 orient right-up.

Code :

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
global ns nf
$ns flush-trace
close $nf
Exec nam out.nam &
exit 0
}
Set n1 [$ns node]
Set n2 [$ns node]
Set n3 [$ns node]
set n4 [$ns node]
Set n5 [$ns node]
set n6 [$ns node]

$ns duplex-link $n1 $n2 10Mb 10ms DropTail
$ns duplex-link $n1 $n3 10Mb 10ms DropTail
$ns duplex-link $n1 $n4 10Mb 10ms DropTail
$ns duplex-link $n1 $n5 10Mb 10ms DropTail
$ns duplex-link $n1 $n6 10mb 10ms DropTail

$ns queue-limit $n1 $n2 20
$ns queue-limit $n1 $n3 20
```

```
$ns queue-limit $n1 $n4 20
$ns queue-limit $n2 $n5 20
$ns queue-limit $n3 $n6 20

$ns at 5.0 "finish"
$ns run
```

Result :- Hence we successfully performe the experiment and establish the connection between mobile nodes.



15

1024
21/3/23

Date:- 21/03/23

Experiment No.4

Aim :- To Write TCL script for transmission between mobile nodes using TCP connection.

Objective :- To understand the process of generating a wireless system using TCL script for wireless transmission.

Theory :- Once the basic network setup is done, the next things to do is to setup traffic agents such as TCP & UDP, traffic sources such as FTP and CBR, & attach them to nodes & agents respectively.

1) Set tcp [new Agent / TCP] :- This line shows how to create a TCP agent. But in general, users can create any agent of traffic sources in this way. Agents and traffic sources are in fact basic objects, mostly implemented in C++ & linked to OTcl. Therefore, there are no specific Simulator object member fun^n that create these objects instances. To create agents or traffic sources, a user should know the class names these objects. This information can be found in the NS documentation or partly in this documentation.

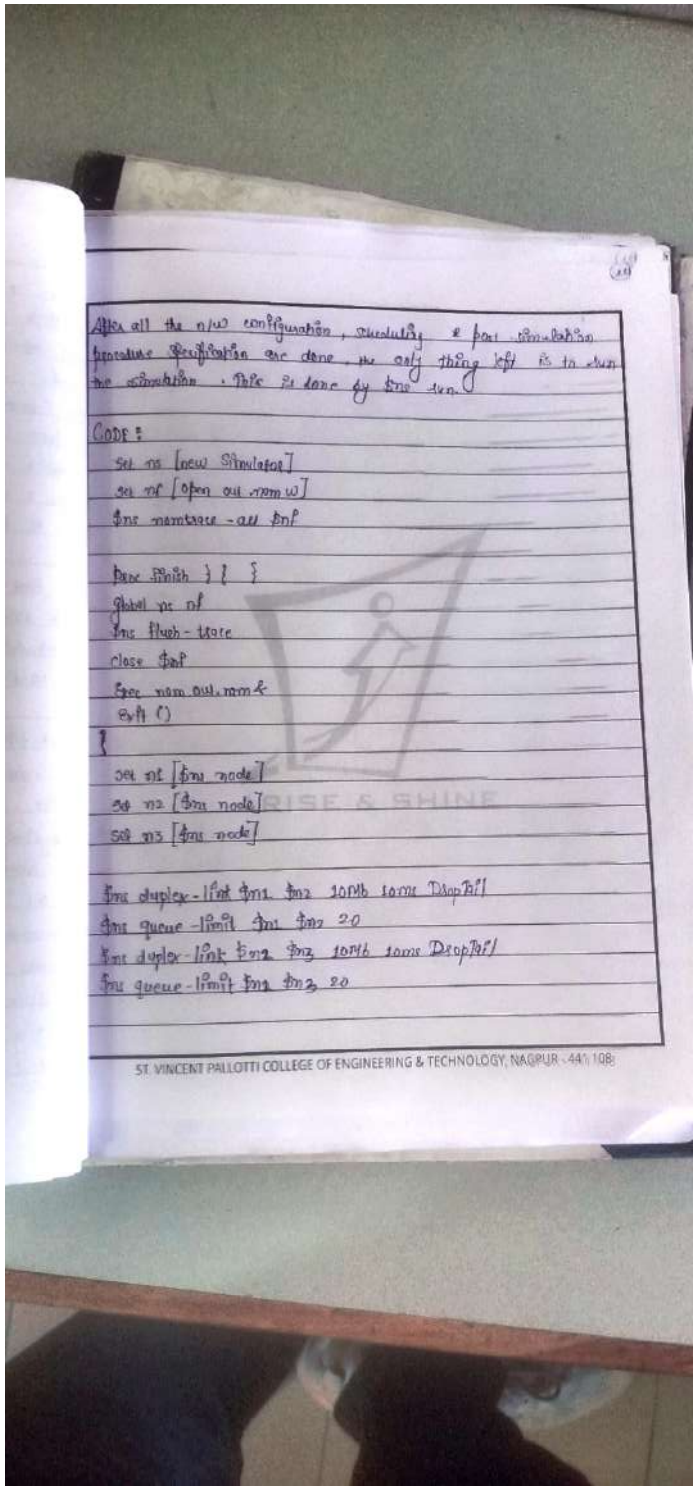
2) $ns attach-agent node agent :- The attach-agent member function attaches an agent object created to a node object. Actually, what this fun^n does is call the attach member fun^n of specified node, which attaches the given agent to itself.

Therefore a user can do the same thing by, for example for attach $tcp. Similarly, each agent object has a member function attach-agent that attaches a traffic source object to itself.

3) $ns Connect agent1 agent2 : After two agents that will communicate with each other are created, the next thing is to establish a logical network connection between them. This line establishes a network connection by setting the destination attributes to each other's ip/w and port address pairs.

Assuming that all the n/w configuration is done, the next thing to do is write a simulation scenario. The simulator object has many scheduling member functions, however, the one that is usually used in the following :

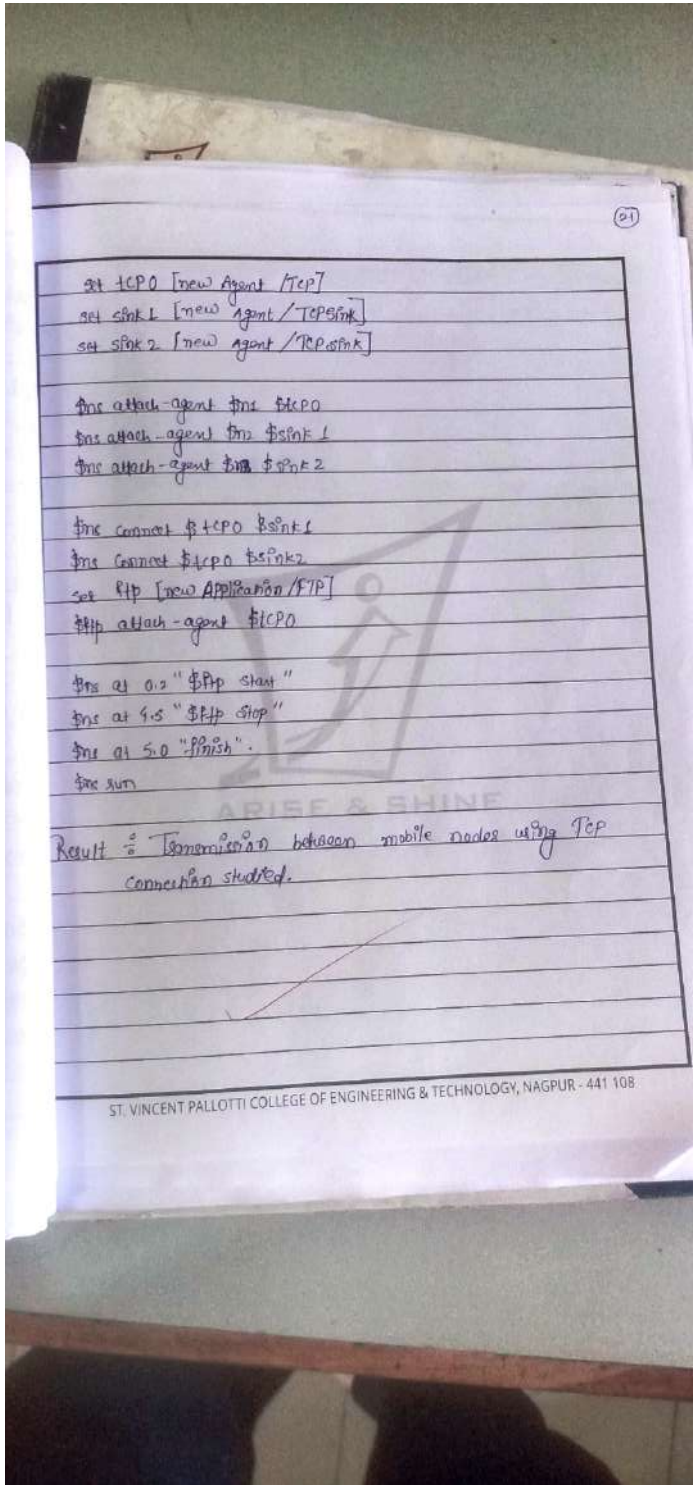
a) $ns.at time "string" :- This member function of a simulator object makes the scheduler (scheduler is the variable that points the scheduler object created by [new scheduler] command at the beginning of the script) to schedule the execution of the specified string at given simulation time. See example $ns at 0.1 "$cbr start" will make the scheduler call a start member function of the CBR traffic source object, which starts the CBR to transmit data. Be it, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, & the agent, just knowing how much of the data to transfer.



After all the n/w configuration, scheduling & port simulation procedure specification are done, the only thing left is to run the simulation. This is done by the run.

Code:

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf

proc finish { } {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit 0
}

set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n1 $n2 10Mb 10ms DropTail
$ns queue-limit $n1 $n2 20
$ns duplex-link $n2 $n3 10Mb 10ms DropTail
$ns queue-limit $n2 $n3 20
```



(21)

set tcp0 [new Agent /Tcp]
set sink1 [new Agent /TCPSink]
set sink2 [new agent /TCPsink]

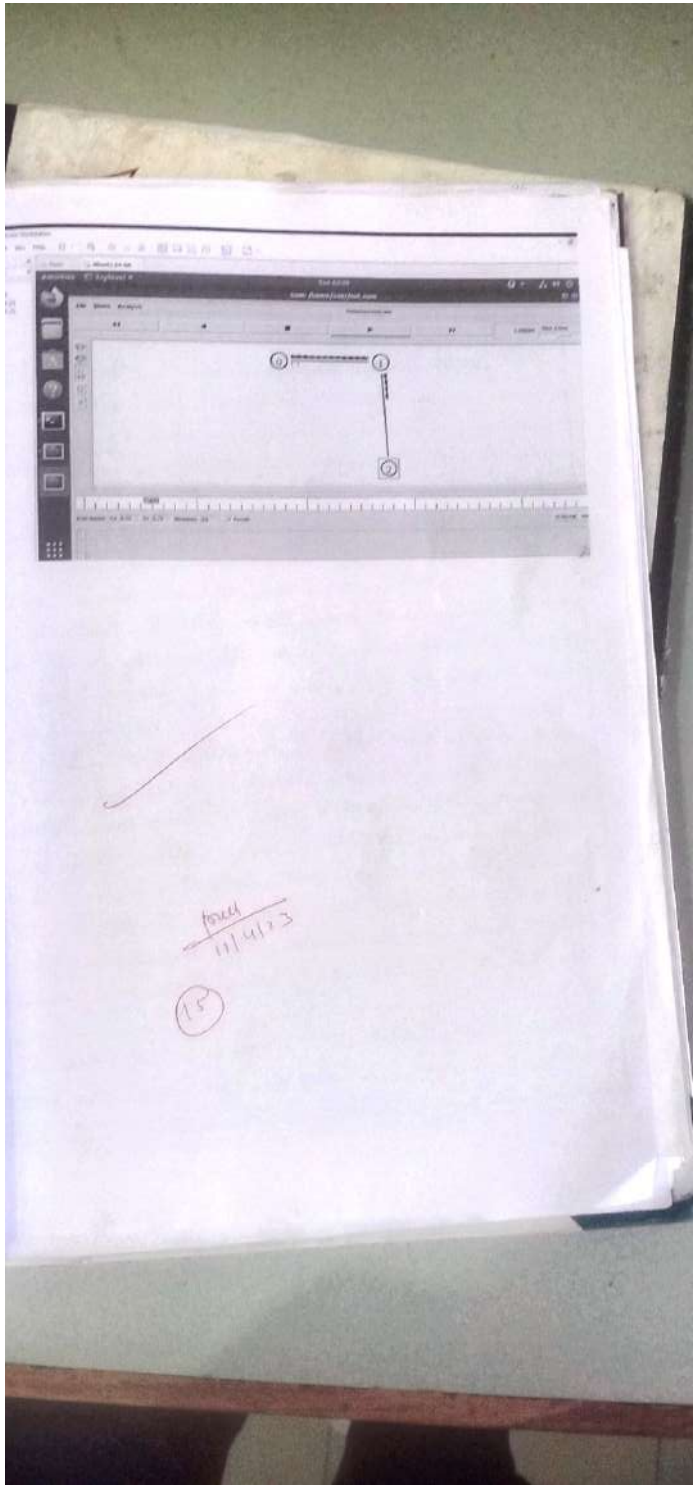
$ns attach-agent $n1 $tcp0
$ns attach-agent $n2 $sink1
$ns attach-agent $n3 $sink2

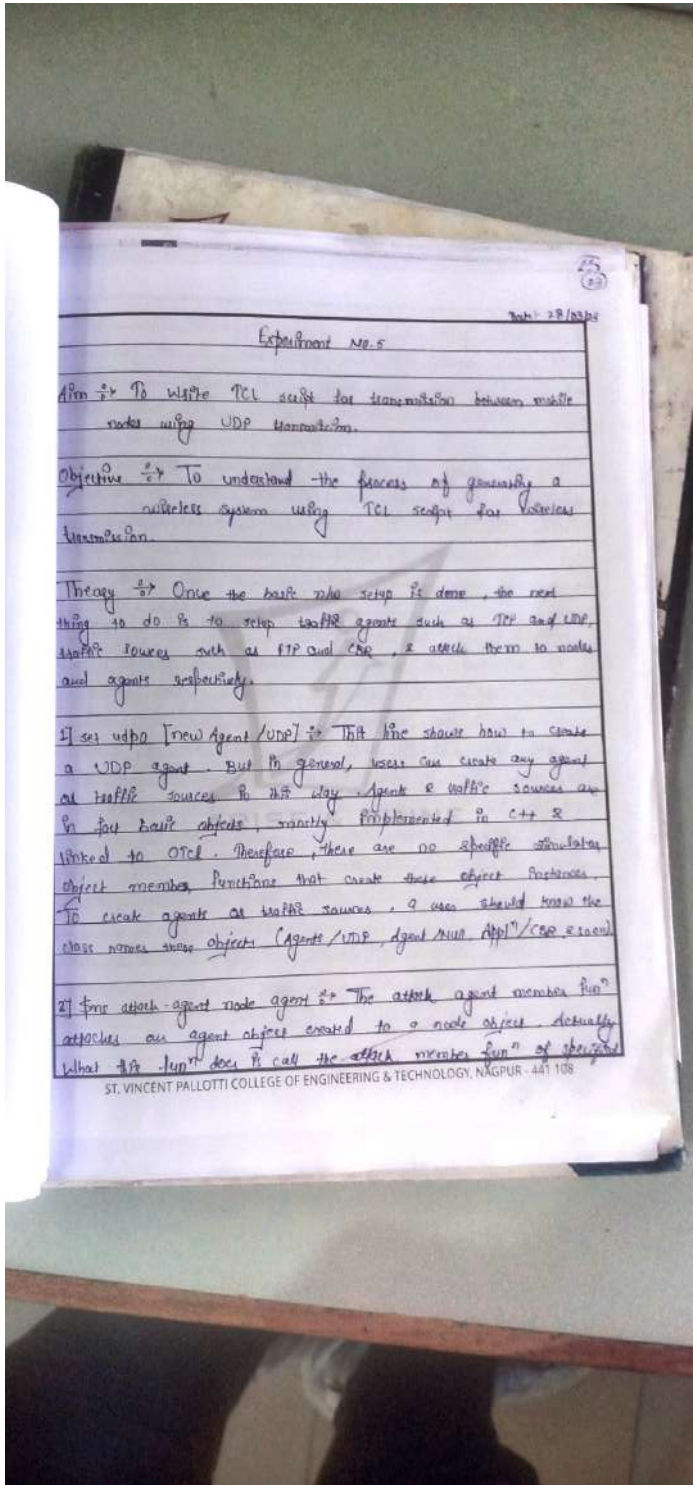
$ns connect $tcp0 $sink1
$ns connect $tcp0 $sink2
set ftp [new Application /FTP]
$ftp attach-agent $tcp0

$ns at 0.2 "$ftp start"
$ns at 4.5 "$ftp stop"
$ns at 5.0 "finish".
$ns run

Result :- Transmission between mobile nodes using Tcp connection studied.

ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING & TECHNOLOGY, NAGPUR - 441 108


force
11/4/23
15



Date: 28/03/24

Experiment No. 5

Aim :- To write TCL script for transmission between mobile nodes using UDP transmission.

Objective :- To understand the process of generating a wireless system using TCL script for wireless transmission.

Theory :- Once the basic n/w setup is done, the next thing to do is to setup traffic agents such as TCP and UDP, traffic sources such as FTP and CBR, & attach them to nodes and agents respectively.

1] set udp0 [new Agent/UDP] :- This line shows how to create a UDP agent. But in general, users can create any agent or traffic sources in this way. Agents & traffic sources are in fact basic objects, mainly implemented in C++ & linked to OTcl. Therefore, these are no specific simulator object members, functions that create these object instances. To create agents or traffic sources, a user should know the class names these objects (Agents/UDP, Agent/TCP, Application/CBR & so on).

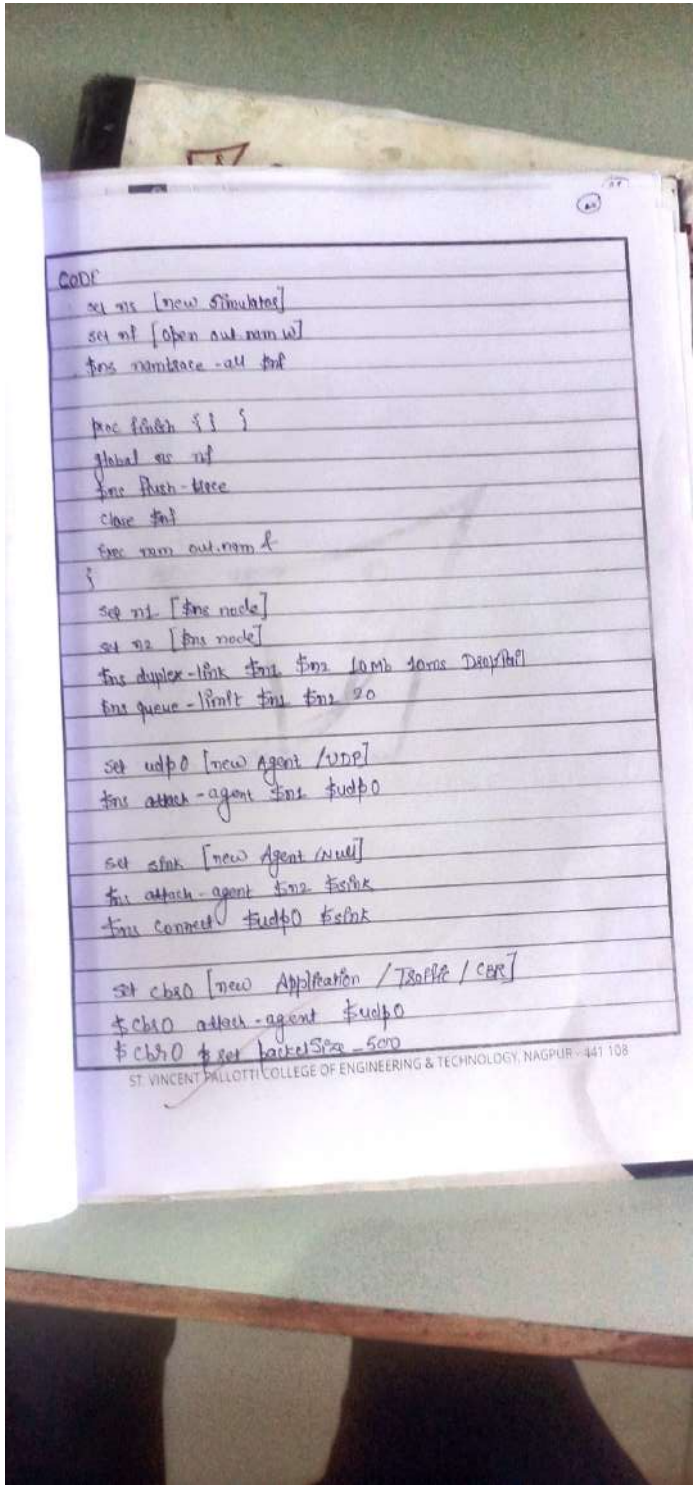
2] $ns attach-agent node agent :- The attach agent member fun^n attaches an agent object created to a node object. Actually what this fun^n does is call the attach member fun^n of the respective

node which attaches the given agent to itself. Therefore, a user can do the same thing by, for example, $tnode attach $udp0. Similarly, each agent object has a member fun" attach-agent that attaches a traffic source object to itself.

3] $ns connect agent1 agent2 :- After two agents that will communicate with each other are created, the next thing is to establish a logical network conn" bet" them. This fun" establishes a n/w conn" by setting the destination address to each others' n/w & port address pair.

4] $ns at time "string" :- This member fun" of a Simulator object makes the scheduler to schedule the execution of specified string at given simulation time. for eg :- $ns at 0.1 "$cbr start" will make the scheduler call a start member function of the CBR traffic source object, which starts the CBR to transmit data. In NS, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, & the agent just knowing how much of the data to transfer creates packets & send them.

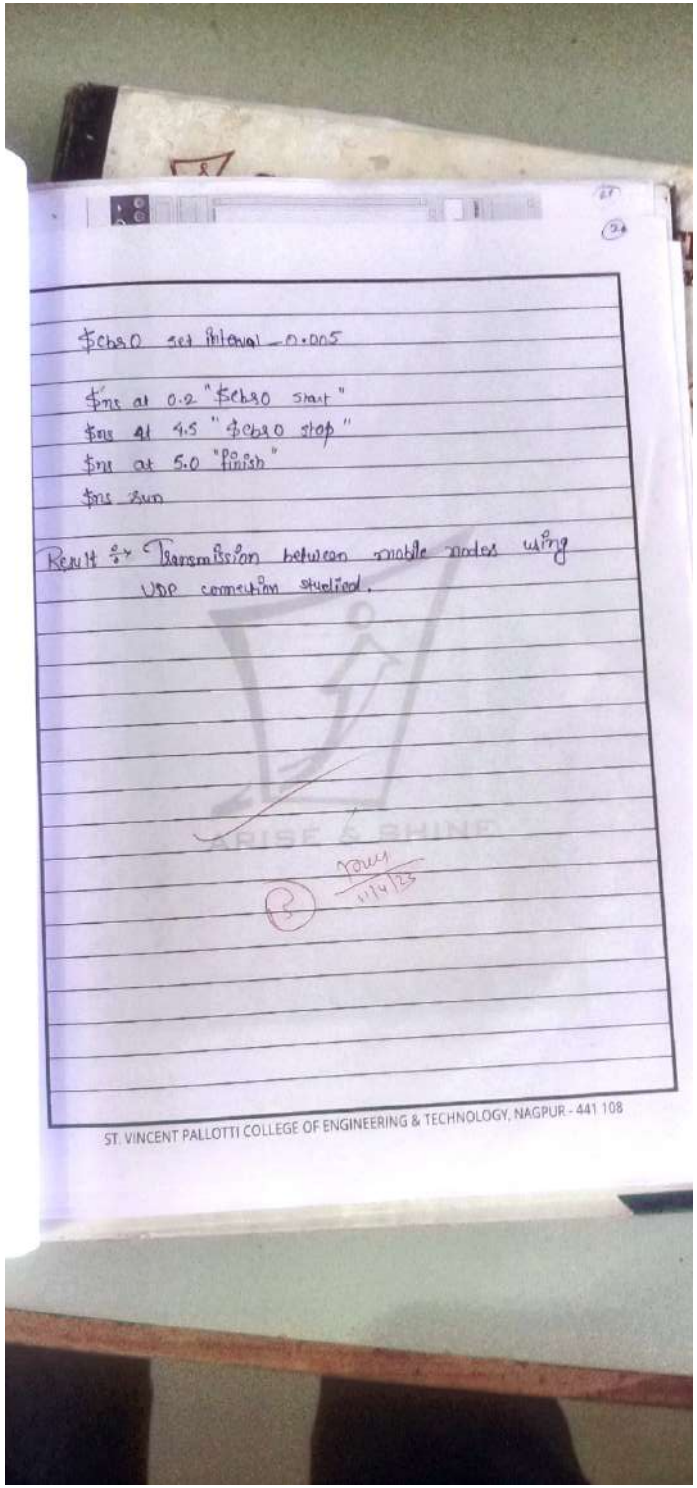
After all n/w configuration, scheduling & post-simulation procedure specifications are done, the only thing left is to run the simulation. This is done by $ns run.



CODE

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
}

set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n1 $n2 10Mb 10ms DropTail
$ns queue-limit $n1 $n2 20

set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0

set sink [new Agent/Null]
$ns attach-agent $n2 $sink
$ns connect $udp0 $sink

set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 500
```

ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING & TECHNOLOGY, NAGPUR - 441 108



$fcb30 set interval 0.005

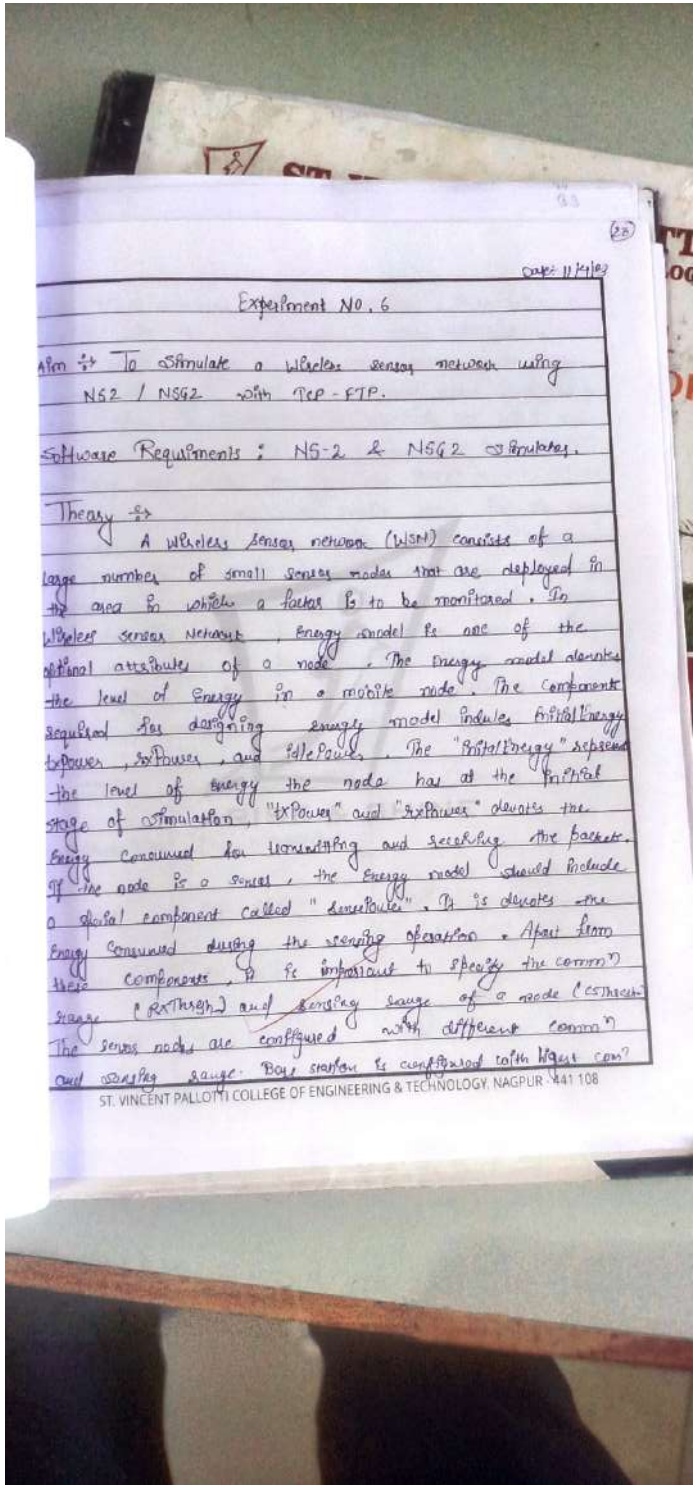
$ns at 0.2 "$fcb30 start"
$ns at 4.5 "$fcb30 stop"
$ns at 5.0 "finish"
$ns run

Result :- Transmission between mobile nodes using UDP connection studied.

Rohit 11/4/23

ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING & TECHNOLOGY, NAGPUR - 441 108

Activities
Top(level
Thu 01:33
nam: /home/user/out.nam
File Views Analysis
0
1
2



(28)

Page No. / /

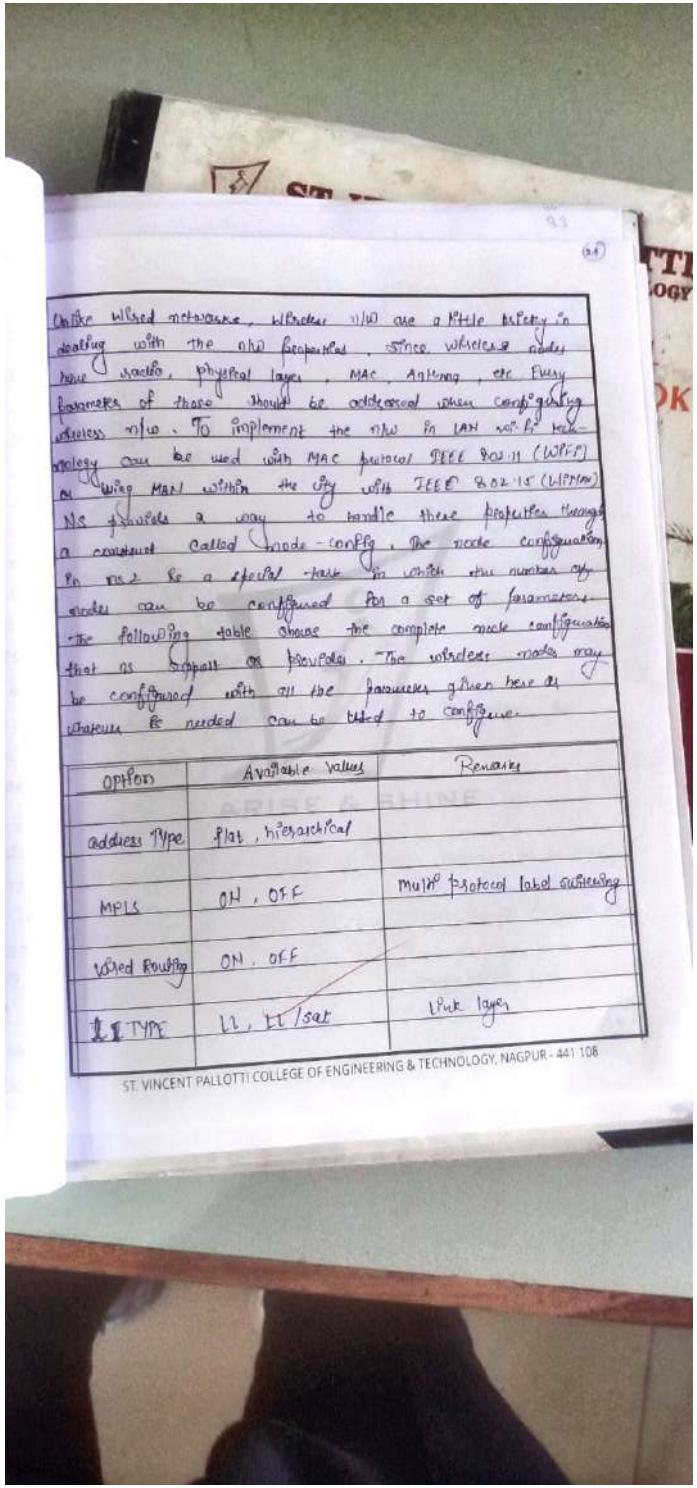
Experiment No. 6

Aim :- To Simulate a Wireless sensor network using NS2 / NSG2 with TCP - FTP.

Software Requirements : NS-2 & NSG2 Simulators.

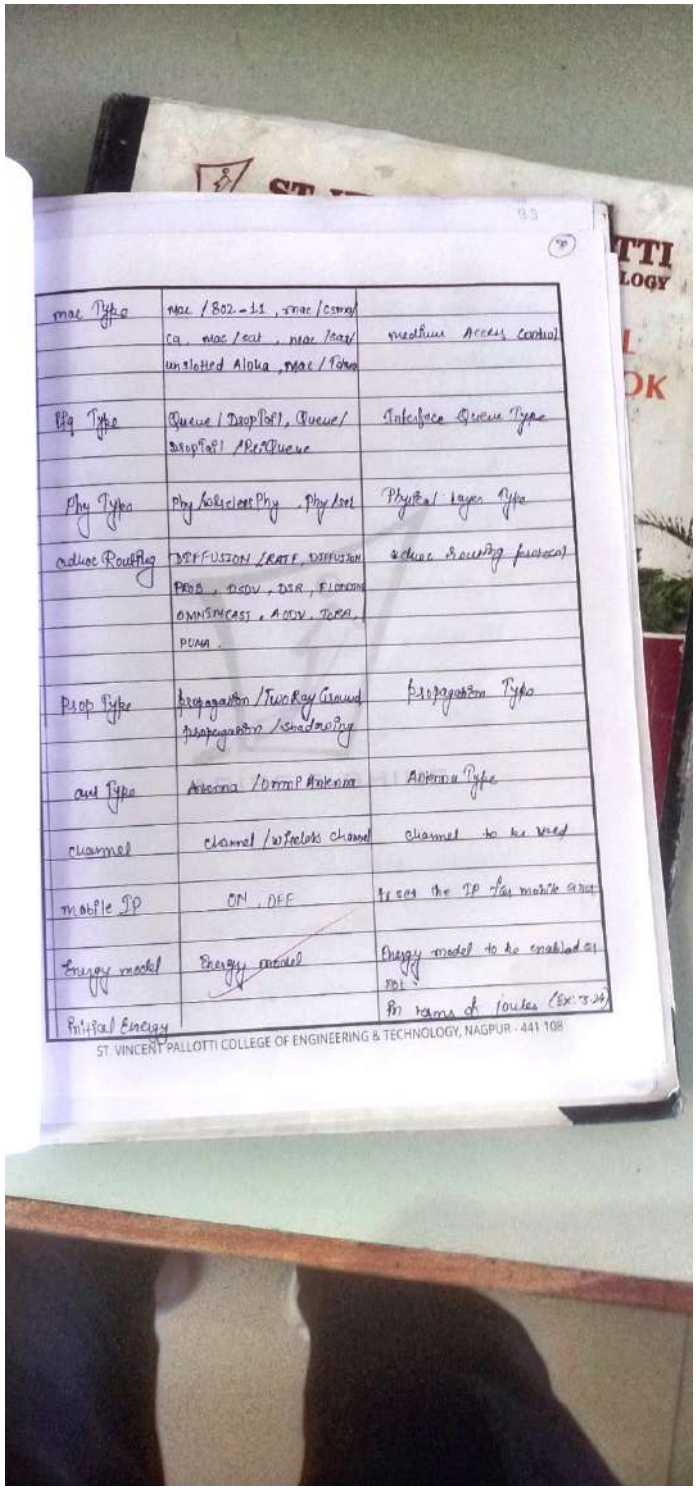
Theory :-

A Wireless Sensor network (WSN) consists of a large number of small sensor nodes that are deployed in the area in which a factor is to be monitored. In Wireless Sensor Network, Energy model is one of the optional attributes of a node. The Energy model denotes the level of Energy in a mobile node. The components required for designing energy model includes Initial Energy, txPower, rxPower, and idlePower. The "Initial Energy" represents the level of energy the node has at the initial stage of simulation, "txPower" and "rxPower" denotes the Energy consumed for transmitting and receiving the packets. If the node is a sensor, the Energy model should include a special component called "sensePower". It is denotes the Energy consumed during the sensing operation. Apart from these components, it is important to specify the common range (RxThresh_) and sensing range of a node (CSThresh_). The sensor nodes are configured with different common and sensing range. Base station is configured with bigger common



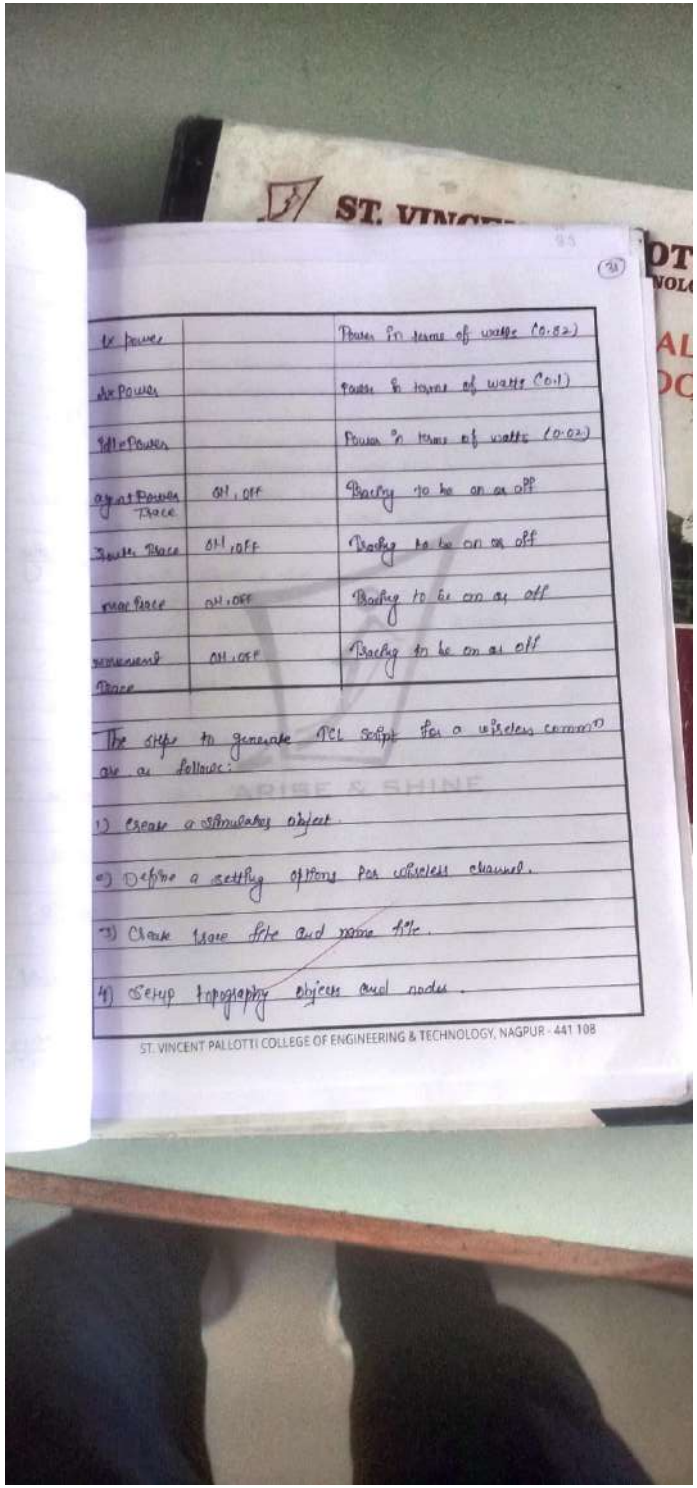
Unlike Wired networks, Wireless n/w are a little tricky in dealing with the n/w properties, since wireless nodes have radio, physical layer, MAC, Antenna, etc. Every parameters of these should be addressed when configuring wireless n/w. To implement the n/w in LAN we Fi technology can be used with MAC protocol IEEE 802.11 (WiFi) or Wireless MAN within the city with IEEE 802.15 (WiMax) NS provides a way to handle these properties through a construct called node-config. The node configuration in ns2 is a special task in which the number of nodes can be configured for a set of parameters. The following table shows the complete node configuration that ns support or provides. The wireless node may be configured with all the parameters given here as whatever is needed can be used to configure.

Option	Available values	Remarks
address Type	flat, hierarchical	
MPLS	ON, OFF	Multi protocol label switching
Varied Routing	ON, OFF	
LL TYPE	LL, LL/sat	Link layer



mac Type	mac / 802_11 , mac / csma/ ca , mac / cat , mac / tdma/ unslotted Aloha , mac / fdma	medium Access control
ifq Type	Queue / DropTail, Queue/ DropTail / PriQueue	Interface Queue Type
Phy Type	Phy / WirelessPhy , Phy / Sat	Physical Layer Type
adhoc Routing	DIFFUSION / RATE, DIFFUSION PROB , DSDV , DSR , FLOODING OMNIMCAST , AODV , TORA , PUMA .	adhoc Routing protocol
Prop Type	Propagation / TwoRay Ground Propagation / Shadowing	Propagation Type
ant Type	Antenna / OmniAntenna	Antenna Type
channel	channel / wireless channel	channel to be used
mobile IP	ON , OFF	To use the IP for mobile node
Energy model	Energy model	Energy model to be enabled or not In terms of joules (Ex: 3.5)
Initial Energy		

ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING & TECHNOLOGY, NAGPUR - 441 108



tx power		Power in terms of watts (0.82)
rx Power		Power in terms of watts (0.1)
Idle Power		Power in terms of watts (0.02)
Agent Power Trace	ON, OFF	Tracing to be on or off
Sent. Trace	ON, OFF	Tracing to be on or off
recv Trace	ON, OFF	Tracing to be on or off
movement Trace	ON, OFF	Tracing to be on or off

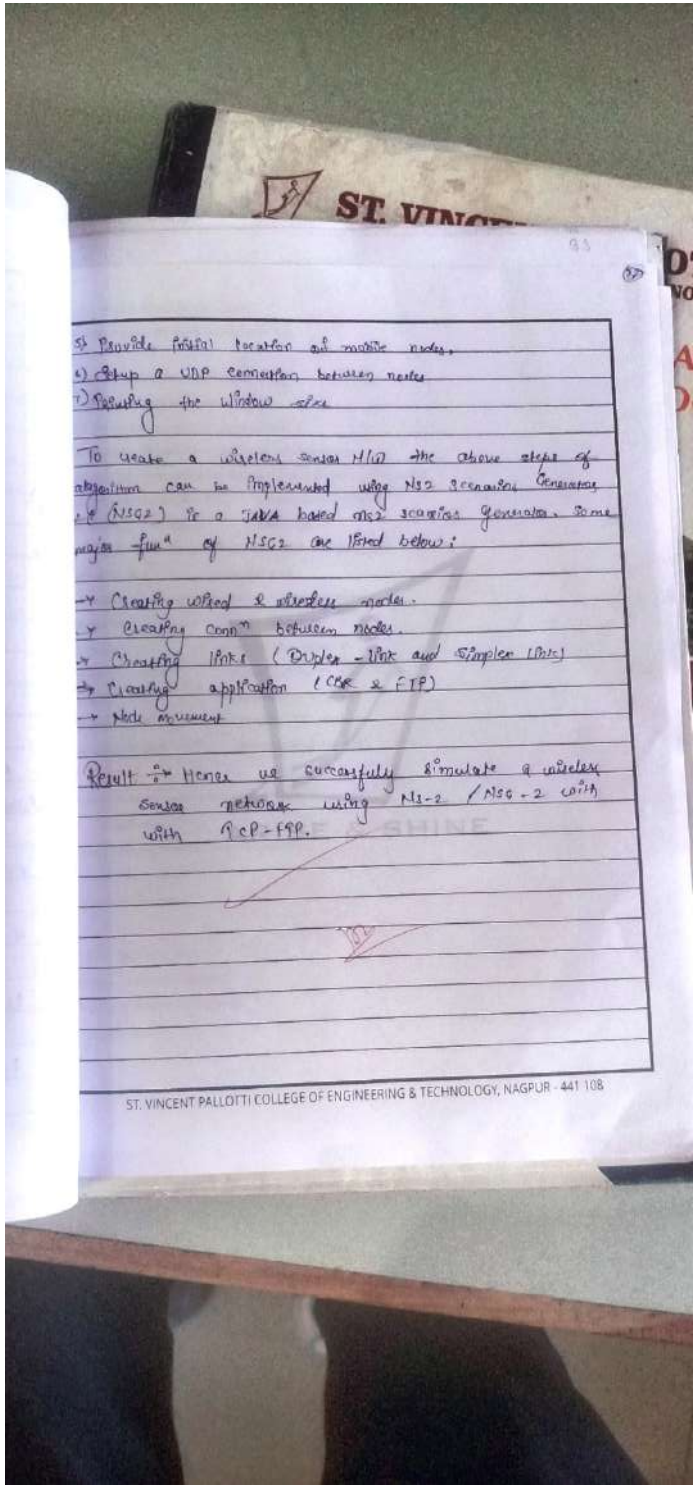
The steps to generate TCL script for a wireless scenario are as follows:

1) Create a simulator object.

2) Define a setting options for wireless channel.

3) Create trace file and name file.

4) Setup topography objects and nodes.



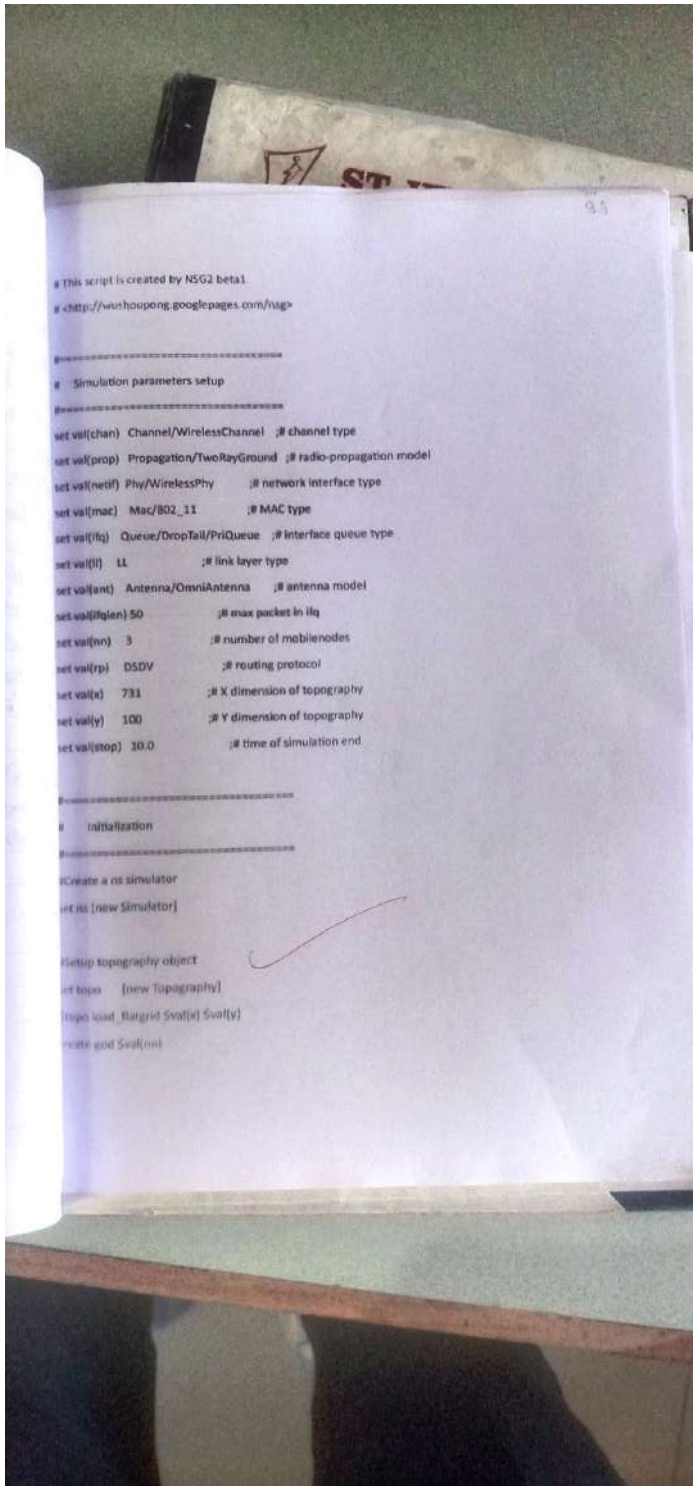
5) Provide initial location of mobile nodes.
6) Setup a UDP connection between nodes
7) Resizing the window size

To create a wireless sensor N/W the above steps of algorithm can be implemented using NS2 scenarios Generator. (NSG2) is a JAVA based ns2 scenario Generator. Some major functions of NSG2 are listed below:

- Creating wired & wireless nodes.
- Creating connection between nodes.
- Creating links (Duplex-link and Simplex link)
- Creating application (CBR & FTP)
- Node movement

Result :- Hence we successfully simulate a wireless sensor network using NS-2 / NSG-2 with with TCP-FTP.

ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING & TECHNOLOGY, NAGPUR - 441 108

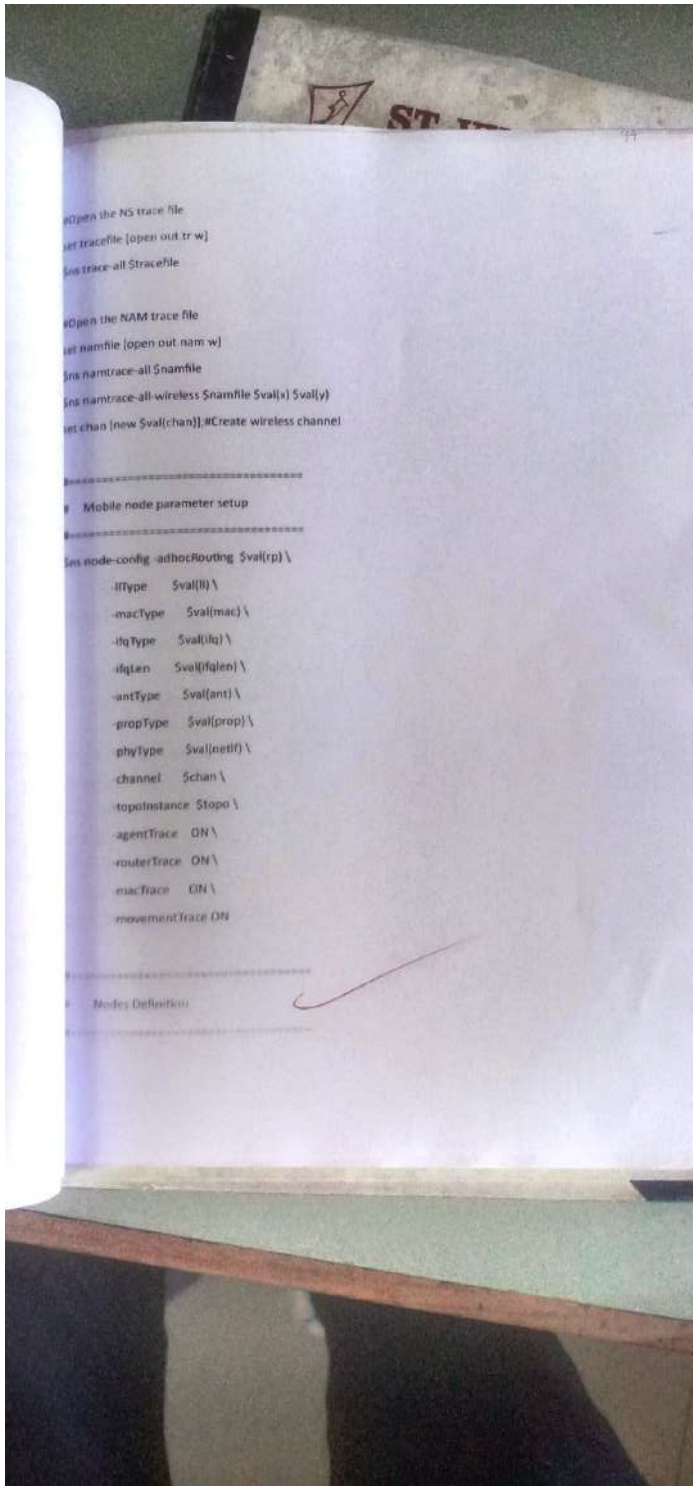


This script is created by NSG2 beta1
<http://wushoupong.googlepages.com/nsg>

#===================================
Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel   ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif) Phy/WirelessPhy   ;# network interface type
set val(mac)   Mac/802_11   ;# MAC type
set val(ifq)   Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)   LL   ;# link layer type
set val(ant)   Antenna/OmniAntenna   ;# antenna model
set val(ifqlen) 50   ;# max packet in ifq
set val(nn)   3   ;# number of mobilenodes
set val(rp)   DSDV   ;# routing protocol
set val(x)   731   ;# X dimension of topography
set val(y)   100   ;# Y dimension of topography
set val(stop)   10.0   ;# time of simulation end

#===================================
Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo   [new Topography]
topo load_flatgrid $val(x) $val(y)
create god $val(nn)

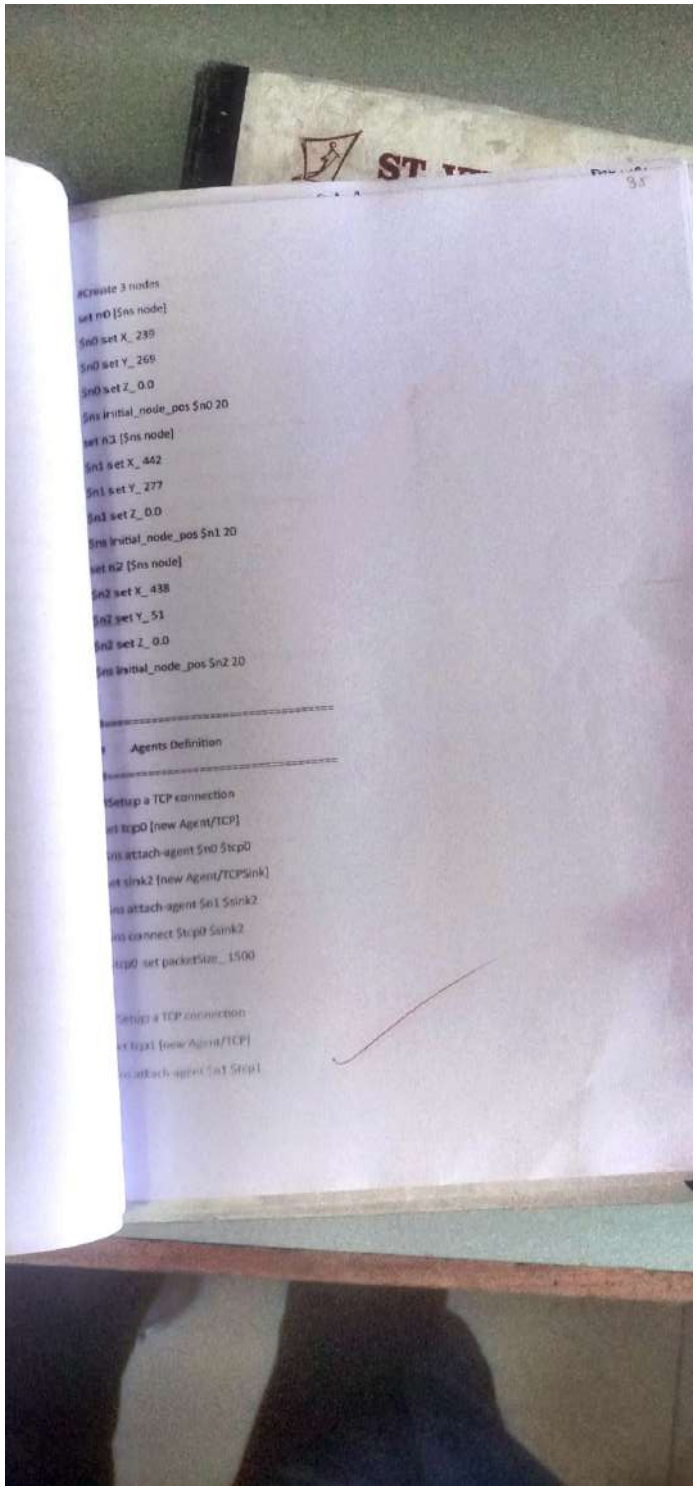


#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#=================================
Mobile node parameter setup
#=================================
$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-channel $chan \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace ON \
-movementTrace ON

#================================
Nodes Definition
#================================

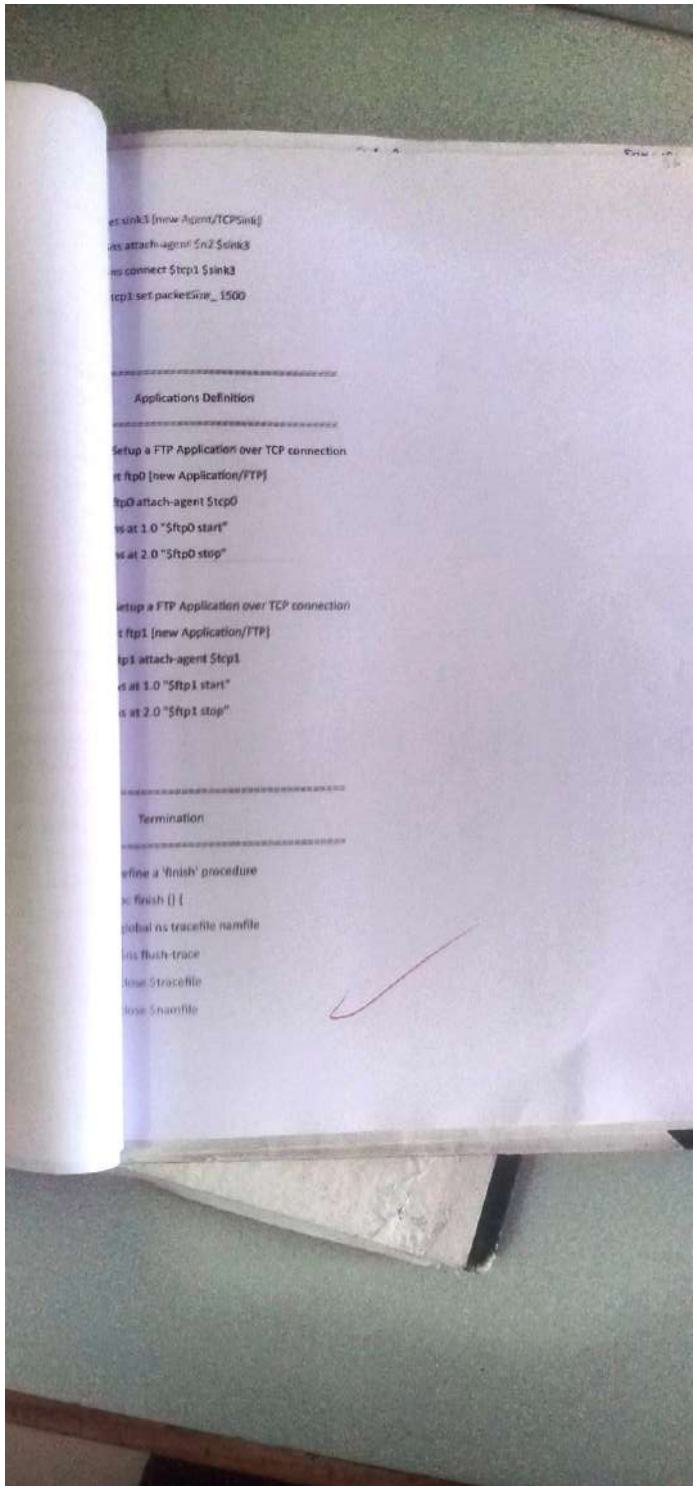


#Create 3 nodes
set n0 [$ns node]
$n0 set X_ 239
$n0 set Y_ 269
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 442
$n1 set Y_ 277
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 438
$n2 set Y_ 51
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20

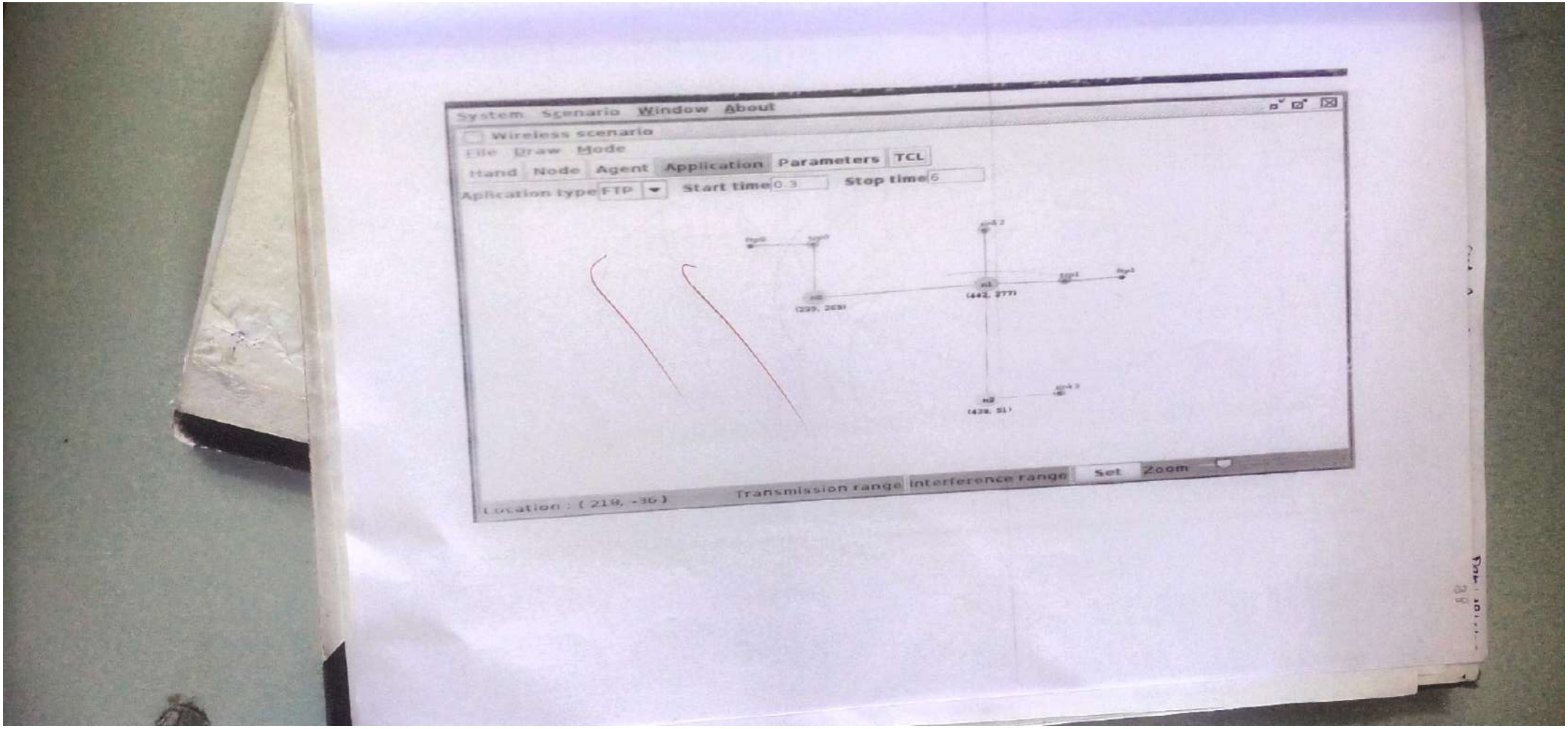
Agents Definition

Setup a TCP connection
set tcp0 [new Agent/TCP]
attach-agent $n0 $tcp0
sink2 [new Agent/TCPSink]
attach-agent $n1 $sink2
connect $tcp0 $sink2
tcp0 set packetSize_ 1500

Setup a TCP connection
tcp1 [new Agent/TCP]
attach-agent $n1 $tcp1


et sink3 [new Agent/TCPSink]
es attach-agent $n2 $sink3
ns connect $tcp1 $sink3
tcp1 set packetSize_ 1500
Applications Definition
Setup a FTP Application over TCP connection
t ftp0 [new Application/FTP]
tp0 attach-agent $tcp0
s at 1.0 "$ftp0 start"
s at 2.0 "$ftp0 stop"
etup a FTP Application over TCP connection
t ftp1 [new Application/FTP]
tp1 attach-agent $tcp1
s at 1.0 "$ftp1 start"
s at 2.0 "$ftp1 stop"
Termination
efine a 'finish' procedure
c finish {} {
lobal ns tracefile namfile
s flush-trace
lose $tracefile
lose $namfile

```
exec nam out.nam &
exit 0

for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$ns$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```


System Scenario Window About
Wireless scenario
File Draw Mode
Hand Node Agent Application Parameters TCL
Aplication type FTP
Start time 0.3
Stop time 6
(442, 277)
(439, 51)
Location : (218, -36)
Transmission range
Interference range
Set
Zoom

File Views Analysis
Step: 398.1ns

Experiment No. 7 Date: 18/4/23

Name :- Aachal Mishra

Year/Section :-3rd year/ A

Roll no. :-03

Aim :- To simulate a wireless network using NSG2 with UDP - CBR.

CODE

```
# This script is created by NSG2 beta1
# <http://wushoupong.googlepages.com/nsg>
#===================================
#   Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel ;          # channel type
set val(prop)   Propagation/TwoRayGround ;         # radio-propagation model
set val(netif)  Phy/WirelessPhy ;                  # network interface type
set val(mac)    Mac/802_11 ;                       # MAC type
set val(ifq)    Queue/DropTail/PriQueue ;          # interface queue type
set val(ll)     LL ;                               # link layer type
set val(ant)    Antenna/OmniAntenna ;              # antenna model
set val(ifqlen) 50 ;                               # max packet in ifq
set val(nn)     4  ;                               # number of mobilenodes
set val(rp)     DSDV ;                             # routing protocol
set val(x)      1243 ;                             # X dimension of topography
set val(y)      100  ;                             # Y dimension of topography
set val(stop)   10.0 ;                             # time of simulation end
#===================================
#       Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]
#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
```

```
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqlen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#===================================
#     Nodes Definition
#===================================
```

```
#Create 4 nodes
set n0 [$ns node]
$n0 set X_ 311
$n0 set Y_ 379
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 312
$n1 set Y_ 33
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 451
$n2 set Y_ 203
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 686
$n3 set Y_ 203
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20

#=================================
#        Agents Definition
#=================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null4 [new Agent/Null]
$ns attach-agent $n2 $null4
$ns connect $udp0 $null4
$udp0 set packetSize_ 1500
```

```
#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set null3 [new Agent/Null]
$ns attach-agent $n2 $null3
$ns connect $udp1 $null3
$udp1 set packetSize_ 1500

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null5 [new Agent/Null]
$ns attach-agent $n3 $null5
$ns connect $udp2 $null5
$udp2 set packetSize_ 1500

#=====================================
#       Applications Definition
#=====================================
#Setup a CBR Application over UDP connection
set cbr9 [new Application/Traffic/CBR]
$cbr9 attach-agent $udp1
$cbr9 set packetSize_ 1000
$cbr9 set rate_ 1.0Mb
$cbr9 set random_ null
$ns at 1.0 "$cbr9 start"
$ns at 2.0 "$cbr9 stop"

#Setup a CBR Application over UDP connection
set cbr10 [new Application/Traffic/CBR]
$cbr10 attach-agent $udp0
```

```
$cbr10 set packetSize_ 1000
$cbr10 set rate_ 1.0Mb
$cbr10 set random_ null
$ns at 1.0 "$cbr10 start"
$ns at 2.0 "$cbr10 stop"

#Setup a CBR Application over UDP connection
set cbr11 [new Application/Traffic/CBR]
$cbr11 attach-agent $udp2
$cbr11 set packetSize_ 1000
$cbr11 set rate_ 1.0Mb
$cbr11 set random_ null
$ns at 1.0 "$cbr11 start"
$ns at 2.0 "$cbr11 stop"

#=====================================
#        Termination
#=====================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
```

```
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```


Fig. connections between UDP-CBR, DSDV Protocol

Fig : OUTPUT

Result → Hence we successfully simulate a wireless sensor Network using NS2-2 with UDP - CBR.

19/4/22

suppose we have two nodes and such that is the next hop of to some destination. Also, suppose the sequence number and hop count of the routes to at and are seq a, hent a respectively.

In AODV, nodes discover routes in request response cycles. A node request a route to a destination by broadcasting an RREQ message to all its neighbors. When a node receives an RREQ message but does not have a route to the requested destination, it in turn broadcast the RREQ message. Also it remembers a reverse - route to the requesting node which can be used to forward process repeats until the RREQ reaches a node that has a valid route to the destination. This node responds with an RREP message. This RREP is unicast along the reverse - routes of the intermediate nodes until it reaches the original requesting node.

Date : 8/

Experiment NO.8

Aim : To simulate a Wireless sensor Network using NS62 with the combination of TCP & UDP using AODV. protocol.

Theory : AODV is stands for Ad-hoc On-demand Distance Vector is loop-free routing protocol for ad-hoc networks. It is designed to be self-starting in an environment of mobile nodes, with standing a variety of network behaviors such as node mobility, link failure and packet losses. This section describe the AODV protocol in brief.

At each node, AODV maintains a routing table. The routing table entry for a destination contains three essential fields : a next hop node, a sequence number and a hop count. All packets destined to the destination are sent to the next hop node. The sequence number acts as a form of time-stamping, and is a measure of the freshness of a route. The hop count represent the current distance to the distance node.

Name : Aachal Mishra

Sec : A (3rd yr VI sem)

Roll no : 03

```
# This script is created by NSG2 beta1
# <http://wushoupong.googlepages.com/nsg>

#===================================
#   Simulation parameters setup
#===================================

set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     4                          ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      952                        ;# X dimension of topography
set val(y)      100                        ;# Y dimension of topography
set val(stop)   10.0                       ;# time of simulation end

#===================================
#        Initialization
#===================================

#Create a ns simulator
set ns [new Simulator]

#Setup topography object
```

```
set topo      [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel
#=====================================
#     Mobile node parameter setup
#=====================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance $topo \
                -agentTrace  ON \
                -routerTrace  ON \
                -macTrace     ON \
                -movementTrace ON
#=====================================
```

```
#       Nodes Definition
#======================================
#Create 4 nodes
set n0 [$ns node]
$n0 set X_ 311
$n0 set Y_ 389
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 323
$n1 set Y_ 33
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 452
$n2 set Y_ 214
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 682
$n3 set Y_ 213
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
#======================================
#       Agents Definition
#======================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null3 [new Agent/Null]
$ns attach-agent $n2 $null3
```

```
$ns connect $udp0 $null3
$udp0 set packetSize_ 1500

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null5 [new Agent/Null]
$ns attach-agent $n3 $null5
$ns connect $udp2 $null5
$udp2 set packetSize_ 1500

#Setup a TCP connection
set tcp6 [new Agent/TCP]
$ns attach-agent $n1 $tcp6
set sink7 [new Agent/TCPSink]
$ns attach-agent $n2 $sink7
$ns connect $tcp6 $sink7
$tcp6 set packetSize_ 1500

#=====================================
#       Applications Definition
#=====================================

#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr0 stop"

#Setup a FTP Application over TCP connection
```

51

53

Result
2/11/23

15

Fig : OUTPUT

Fig : UDP-CBR, AODV protocol

$ns run

```
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp6

$ns at 1.0 "$ftp3 start"
$ns at 2.0 "$ftp3 stop"


#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp2
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 2.0 "$cbr4 stop"

#=================================
#      Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
```

52