Computer Network Simulation in NS2

Basic Concepts and Protocols Implementation

Prof. NEERAJ BHARGAVA
PRAMOD SINGH RATHORE
Dr. RITU BHARGAVA
Dr. ABHISHEK KUMAR

bpb

Computer Network

Simulation

in NS2

Basic Concepts and

Protocols Implementation

---

by

Prof. Neeraj Bhargava

Prof. Neeraj Bhargava

Dr. Ritu Bhargava

Dr. Abhishek Kumar

FIRST EDITION 2020

ISBN: 978-93-88511-827

Distributors:

BPB PUBLICATIONS

20, Ansari Road, Darya Ganj

New Delhi-110002

Ph: 23254990/23254991

MICRO MEDIA

Shop No. 5, Mahendra Chambers,

150 DN Rd. Next to Capital Cinema,

V.T. (C.S.T.) Station, MUMBAI-400 001

Ph: 22078296/22078297

DECCAN AGENCIES

4-3-329, Bank Street,

Hyderabad-500195

Ph: 24756967/24756400

BPB BOOK CENTRE

376 Old Lajpat Rai Market,

Delhi-110006

Ph: 23861747

Published by Manish Jain for BPB Publications, 20 Ansari Road, Darya Ganj, New Delhi-110002 and Printed by him at Repro India Ltd, Mumbai

Dedicated to

Our beloved Parents and Family

About the Authors

◆

Prof. Neeraj Bhargava has done B.Tech (Computer Science) from Central University of Hyderabad (India), M.S (Computer Science from BITS Pillani, (India) completed PhD (Computer Science) from University of Rajasthan, Jaipur (India). He has been consistently delivering his services in teaching and motivating the students for more than 3 decade now. Currently he has been associated with department of Computer Science, School of Engineering and System Sciences, MDS University, Ajmer, Rajasthan, India as a professor & Head. He is also Nodal in-charge for NME-ICT Project, MHRD, and Govt. OF INDIA. He has been guiding many research scholars in Computer Science. He is convener and member of many academic bodies like board of studies, and many reputed Universities in India. He has numerous Books on national and International level and more than 100 prominent research papers in International Journals and contributed as organizing chair in more than 15 National & International conferences. His Area of research includes primarily GIS Spatial data management System, Machine learning, 3 D Computer Graphics, Computer Vision, E-learning and Ad Hoc Network. His work for facerecognition and finger print recognition has been cited in many reputed research work. He has been visited many foreign countries as a key note speakers.

◆

Pramod Singh Rathore is pursuing his Doctorate in computer science&Engineering from Bundelkhand University and research is going on Networking and done M. Tech in Computer Sci. & Engineering from Government engineering college Ajmer, Rajasthan Technical University, Kota India. He has been working as an Assistant professor of Computer Science& Engineering Department at Aryabhatt Engineering College and Research centre, Ajmer, Rajasthan and also visiting faculty in Government University MDS Ajmer. Hehas total Academic teaching experience of more than 8 years with more than 45 publications in reputed, peer reviewed National and International Journals, books & Conferences like Wiley, IGI GLOBAL, Taylor & Francis Springer, Elsevier Science Direct, Annals of Computer Science, Poland, and IEEE. He has co-authored & editedmany books with many reputed publisher like Wiley, CRC Press, USA. His research area includes NS2, Computer Network, Mining, and DBMS. He is in editorial and advisory committee of Global journal group. He is also member of various National and International professional societies in the field of engineering & research like Member of IAENG (International Association of Engineers).

◆

Dr Ritu Bhargava is Assistant Professor at Department of Computer Science,Sohia Girls', Ajmer. She has PhD in Computer Science from Hemchandracharya North Gujarat University Patan, Gujarat, India. She has more than 15 years of active teaching and research experience. She has delivered a series of talks and lectures at reputed University of India. She is a life member of CSI, member of ACM, and IEEE.She has 3Books on national and International publishers and more than 30 prominent research papers in International Journals and contributed as organizing chair in more than 15 National & International conferences. Her Area of research includes Data Mining, GIS Spatial data management System, Machine learning, 3 D Computer Graphics, Computer Vision, E-learning and Ad Hoc Network. Her work for facerecognition and finger print recognition has been cited in many reputed research work.

◆

Dr. Abhishek Kumar is currently working as an assistant professor Research in Chitkara University Punjab, India, he is Doctorate in computer science from University of Madras done M.Tech in Computer Sci. & Engineering from Government engineering college Ajmer, Rajasthan Technical University, Kota India. He has total Academic teaching experience of more than 8 years with more than 55 publications in reputed, peer reviewed National and International Journals, books & Conferences like Wiley, Taylor & Francis Springer, Elsevier Science Direct, Inderscience, Annals of

Computer Science, Poland, and IEEE. His research area includes-Artificial intelligence, Image processing, Computer Vision, Data Mining, Machine Learning. He has authored 5 books published internationally and Editing more than 10 book with, Elsevier, Wiley, IGI GLOBAL Springer, Apple Academic Press and CRC etc. he is also member of various National and International professional societies in the field of engineering & research like Senior Member of IEEE, IAENG(International Association of Engineers), Associate Member of IRED (Institute of Research Engineers and Doctors), Associate Member of IAIP (International Association of Innovation Professionals), Member of ICSES (International Computer Science and Engineering Society), Life Member of ISRD (International Society for research & Development), Member of ISOC (Internet Society),Editorial Board member in IOSRD He has got Sir CV Raman life time achievement national award for 2018 in young researcher and faculty Category.

Acknowledgements

We express our sincere appreciation to all those who were involved in preparation of this book. We thanks to our Hon'ble Vice Chancellors who has fostered excellent academic climate at the respective university which made this endeavour possible.

We express our gratitude to the reviewers for their valuable suggestions and the editorial team for their support throughout the development. We also thank all those who directly or indirectly stand behind the success story of the book even though their names have not been mentioned.

And as always, our families have provided much needed encouragement and understanding; without their support and love we would not have been able to write this book. And above all, we give thanks to God for allowing us to see the completion of our work.

—Prof. Neeraj Bhargava

— Pramod Singh Rathore

— Dr. Ritu Bhargava

— Dr. Abhishek Kumar

Preface

Network Simulation is most sought to research field and it now becomes an integral part of many research projects (commercial applications, academic research). The Networking and communica-tions domain start from finding friends on the social networking sites to medical diagnosis to smart cities implementation and even for satellite processing. In this book, we have made an honest effort to make the concepts of network simulation easy and all the basics programs are explained in an easy and simple manner in NS2 simulator right from the installation part. As the real-time application of networking and communications is endless but the basics concepts and algorithms are discussed by using NS2 simulator so that from graduate students to researchers can get benefited with this.

Who should read this book?

The book is basically meant for all those graduate and research students who find the algorithms & protocols of networking and communications difficult for implementations. In this book, all basic protocols of networking &simulationare discussed in detail with the practical approach. Primarily beginners can find this bookmore effective as the chapters are subdivided in such a way, they will find building and implementing algorithms in NS2 interesting and easy.

Why we wrote this book

The writers for this book teamed up from academicand research domain, so we take care of things that the text and the flow of chapter's content are easy to understand for the beginners. There are many books on Network & communications and WSN for graduate students or for Ph.D. understudies in Computer science and electronics field, and they are brimming with advanced arithmetic. This brings a conspicuous difference to how NS2 is being utilized, as a product device in research and business applications. Today, applying network simulations does not require a Ph.D. Nonetheless, there are a couple of assets out there which completely cover all the essential parts of actualizing networking and communications., without expecting you to take the advanced math courses. We believe that this book will help an individual who need to apply network simulation without studying upon years of analytics, calculus math, and probability hypothesis.

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors if any, occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

Simulation Script

First Simulation Scenario

Second Simulation Scenario

Node Orientation

Third Simulation Scenario

Wired file format

1. Practical Examples for Wired Program in NS2
2. Mobile Networking in NS2

Steps for mobile networking simulation

CHAPTER 1

Introduction to Network Simulation

Network simulation is not just a concept but it's a real time application and practically oriented notion. The book primarily focuses on the basic practical approach to make Network simulation a lucid learning stream for readers. The basic notion behind compiling this book is to make students and scholar familiar with the fundamental details of network simulation concepts.

If you want to do or learn the concept of networking, then you should do a demo and make a small network and learn from it (how it behaves and what happens at each stage). Constructing a small network is quite expensive, apart from this you can do Simulation on internet.

Simulations: It is a software which performs a task.

Network Simulator 2: This is a version for solving a task. There are various versions available in market such as: ns1, ns2, ns3.

Network

Network allows computers to exchange data. Networking in the computer domain includes network simulator, networked devices to exchange data with each other via data connection (wired or wireless media). The optimal example of the networking in computer domain is the Internet .

The networking in computer domain can be defined as a collection of computers which are interconnected to each other and are used for gathering, processing, and distributing information. Networking includes workstation, the network simulator, servers, routers, modems, base station, wireless points, and so on. Computers transfer data through network simulator links such as copper cables, fibre optic cables, and microwave/satellite/radio links.

Network Simulator

The simulator used in networking is hardware or software which predicts the behavior of the networking in computer domain without actual network being present. In the different simulators, the networking in computer domain is typically modelled with devices, dynamic nodes, traffic between; with these parameters the performance of network simulator is analyzed. Users can model the simulator in order to fulfil their specific needs. Simulator used in networking is a combination of tools which simulates the pattern of networking scenario such as creating the networking topologies, log events which happen under any simulation, analyze the events and understand if the simulators is applicable in networking typically come with support for the most popular protocols and networks we use today, such as Wi-Max, WLAN, WSN, cognitive radio, TCP, and so on.

Overview

Network Simulation 2 is an event-driven simulator that simulates various kinds of IP networks. It implements network protocols such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP),File Transfer Protocol (FTP), Telnet, Web, Constant Bit Rate (CBR) and Variable Bit Rate (VBR), queue management methods such as Drop Tail, RED and CBQ, and some routing algorithm. Network Simulation 2also works with multicasting network-oriented programs and some of the Medium Access Control (MAC) layer protocols for local area network simulations. Network Simulation project is currently working for the VINT project which introduce tools for simulation results display, analysis, and converters which convert network topologies generated by the well-known generators to NS formats. At present, Network Simulation (version 2) is developed in C++ and OTcl . This book discusses briefly about the basic construction of NS and explains in detail how to use of NS frequently by giving examples.

As shown in figure 1.1 , Network simulation is an Object-oriented tool script interpreted with simulation event scheduler, with the libraries of network

component object, and with the libraries of network setup (plumbing) module (it is the plumbing modules which is implemented as member functions of the base simulator object). To use NS, we have to do programming in OTcl script language.

To create and run a simulation, OTcl script should be written by the user which creates an event, initiate the network topology, set up the objects of the network, comment the traffic sources, fix the transmission time and the stop time of transmitting packets through the event scheduler.

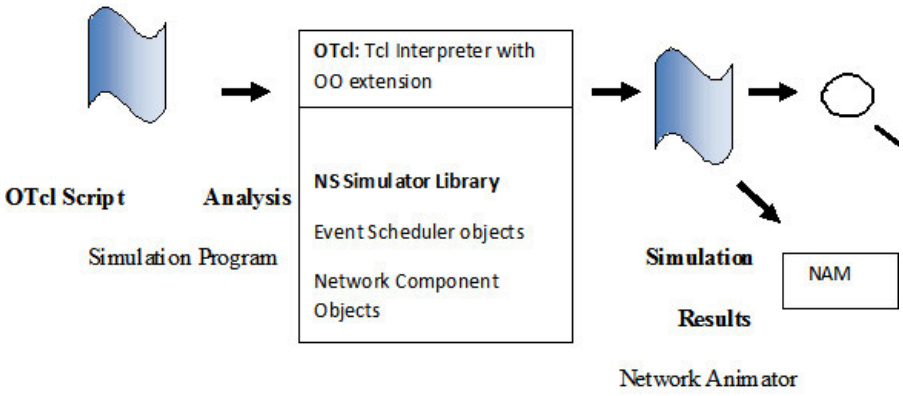


Figure 1.1: Simplified User's View of NS

One more important component of NS beside network objects is the event scheduler . An event in NS is a packet ID which is unique for every packet with the unique scheduled time and the pointer to an object that handles the event. In NS, an event scheduler continuously tracks the simulation time period and fires all the simulation events in the event queue which is programmed for the present time by invoking suitable network components, not are the ones which issued the events in the simulation, and let them to perform the suitable action connected with packet pointed by the event.

Network components communicates with transitory packet; however, this does not devour real simulation time. Each network components spend a little simulation time for handling a packet with an essential delay uses event scheduler by providing an event for the packet. Consider an example, a network switch component which handles the simulation with 20 microseconds of switching delay issues an event for a data packet to be switched to the scheduler as an event 20 microsecond afterward. The scheduler following 20 microseconds handle the process of dequeue the event and fires it to the switch component, which subsequently send the packet to a suitable output link component.

Other control of event scheduler is timer. For example, Transmission Control Protocol (TCP) needs to make the use of a timer to track the transmission time of a packet out for the further transmission (transmission of a packet with the similar TCP packet number but dissimilar network packet identification). Timer measures a time linked with a packet and does a suitable action connected to that packet after a firm time goes by and does not simulate a time delay.

Why TCL

When you write an OTcl script which initiates an event scheduler and devices up the topology parameter of networking by using the item of network and the plumbing feature of the network simulator in the library, and then tells net web page, net page traffics assets. It is done to begin and prevent tracking of the network simulator transmitting packets through the event scheduler. The term " plumbing " is used for a network setup, because of the fact of installing a network is a plumbing viable records paths among an item of the network as the same way of setting the " neighbor " pointer of an item to cope up with the correct item. If a consumer wants to make a brand new network object, she or he may be capable of the consequences that may come while making an object each with the resource of the manner of writing a current item or through the manner of making a compound object from the object library, and plumb the records course through the object. This may seem to be a complex mission, but the plumbing OTcl modules , make the method very clean. The power of the network simulator comes from this plumbing.

The network simulators are no longer written in OTcl, but they are written in C++ additionally. For the overall performance purpose, the network simulator separates the statistics route implementation from the manage path implementation and the network simulator. So, in order to lessen the packet and event processing time (not the simulation time), the occasion scheduler and the fundamental components of object of networking the records direction are written and are compiled in C++. The one compiled gadget are made available to the OTcl interpreter through an OTcl linkage that creates an equal OTcl item for each of the C++ devices. It deploys the network simulator and the configurable variables specific through the way of the C++ object and act as member characteristic of the network simulator and the member variables of the corresponding OTcl item. In this way, the controls of the C++ gadgets are given to OTcl . It is also feasible to feature the member feature of the network simulator and variables to a C++ associated OTcl object.

The devices in C++ that don't need to be managed in a simulation or is internally used by every item is not associated with OTcl. Likewise, an object may be completely achieved in OTcl. Figure 1.2 indicates an object hierarchy instance in C++ and OTcl . One point to be noted here for

C++ devices is that, devices having an OTcl linkage forming a hierarchy, there must be an identical OTcl item hierarchy similar to C++ .

Even as a simulation is completed, the network simulator produces one or extra-textual content; primarily based on the output documents that include simulation statistics. The statistics can be used for simulation evaluation or as an input to a graphical simulation show device called as Network Animator (NaM) . Nam has a pleasing graphical interface just like that of a CD player (play, fast forward, rewind, pause and so forth), and moreover has a show speed controller. Furthermore, it could graphically show an information which incorporates throughput and the quantity of packet drops at each link, regardless of the reality that the graphical information can't be used for proper simulation assessment.

The muse of the hierarchy is the Tcl item class which is the splendid class of all OTcl library devices (scheduler, network components, timers, and the other devices which incorporates nam related ones). As an previous class of Tcl item , the network simulator object elegance is the tremendous class of all easy additives of the object of network that deal with the packets, which can also compose compound item of network together with the nodes present in the network and the links. The primary components of the networking are divided into subclasses, connector, and classifier, are primarily based on the amount of the viable output information paths. The primary item of network that have the handiest one output information course are underneath the connector class and switching devices which have feasible couple of output statistics paths in the classifier class.

Figure 1.2 shows a Network Component

Figure 1.2: Class Hierarchies (Partial)

Figure 1.3 Shows Node and Routing

Figure 1.3: Nodes (Unicast and Multicast)

A node of a network is a compound object, composed by both the node entry object and classifiers as shown in figure 1.3 .

There are two important types of nodes in network simulation:

Unicast node , it has an address classifier with it that does the operation of unicast routing and a port classifier.

Multicast node , in addition to routing and port classification, it has a classifier which classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

In network simulation, Unicast nodes are present by default. In order to create a Multicast node, the user must clearly notify in the input of the OTcl script , after the exact creation of the scheduler object, the nodes which are created will be in the form of multicast nodes. Next process is specification, once it is done then specify the node type. The user can also select an exact routing protocol other than using a predefined one.

Unicast

$ns rtproto type

type: Static, Session, DV, cost, multi-path

Multicast

$ns multicast (right after set $ns [new Scheduler])

Link

Figure 1.4: Link

Firstly, it notifies that a node's output queue is implemented in the form of simplex link object. After completing the process of dequeued; packets from a queue are passed to the Delay object that again simulates the link delay, and then dropped packets at a queue are transferred to a Null Agent and are made freed there. Finally, the TTL object calculates Time To Live (TTL) parameters for each received packets and updates.

Tracing

In NS2, activities of the network are traced around the simplex links. If the simulator is intended to trace network activities (with the use of $ns trace-all file or $nsnamtrace-all file), the links are created after this command and will be followed by the upcoming inserted trace objects as shown in figure 1.5 . Users creates a trace object of type type between the given source and destination nodes by using the create-trace { type file srcdst } command.

Figure 1.5: Inserting Trace Objects

Queue Monitor

Tracing objects are designed to document the packet arrival time at which they're located. Although a person gets enough records from the trace, he or she is probably interested in knowing what is going on the network simulator identification. Consideran example, a user inquisitive about red queue behavior may additionally need to degree the dynamics of common queue size and the present-day queue length of a particular red queue

(i.e. Want for queue monitoring). Queue tracking technique may be achieved through the use of objects of queue display and snoop queue objects as shown in figure 1.6 .

Figure 1.6: Monitoring Queues

When a packet arrives, a snoop queue object acknowledges the queue reveal item of this occasion. The queue displays the use of this statistics video display units in the queue. A red queue monitoring instance is proven within the red queue monitor instance segment.

Packet Flow Example

Till now, the two most vital components of the networking (node and hyperlink) have been analyzed. Determine (packet waft examples) suggests internals of instance of a setup of simulation network and packet float. The networking parameters of the network simulator are two nodes present in the network (n0 and n1) of which the network addresses are zero and one respectively. A TCP agent attached to n0, use port 0 to communicates with a tcp sink object connected to n1 port zero. Subsequently, an FTP utility (or traffic supply) is connected to the TCP agent, asking to ship some amount of data. (figure 1.7)

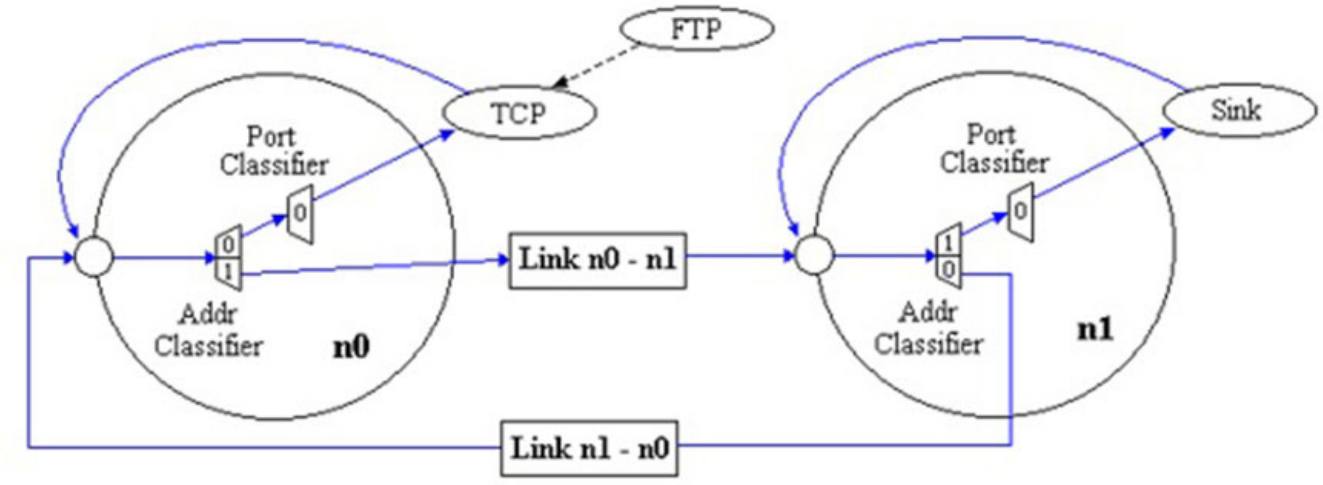


Figure 1.7: Packet Flow Examples

Features of NS-2

NS-2 implementation consists of the following features:

Multicasting is employed here

Simulation of various kinds of wireless networks

Terrestrial (cellular, Adhoc, GPRS, WLAN, BLUETOOTH) satellite network is used

IEEE 802.11 standard can be simulated

Mobile Internet Protocols and Ad hoc protocols such as DSR, TORA, DSDV and AODV Routing are simulated

Software Tools used with NS-2

In the simulation, there are two tools used:

NAM(Network Animator)

xGraph

NS Architecture

Object-oriented (C++,OTCL)

Modular approach

Fine–grained object composition

Reusability

Maintenance

Performance(speed and memory)

Careful planning of modularity

NS Programming

Create the event scheduler

Turn on tracing

Create network

Setup routing

Insert errors

Create transport connection

Create traffic

Transmit application-level data

TCL Interpreter

TclCL is the language used to provide a linkage between C++ and OTcl.Toolkit Command Language (Tcl/OTcl) scripts are written to set up/configure network topologies. TclCL provides linkage for class hierarchy, object instantiation, variable binding, and command dispatching. OTcl is used for periodic or triggered events

The following is written and compiled with C++:

Events Scheduler

NAM: The Network Animator

Xgraph: For plotting

Pre-Processing: Traffic & Topology generator

Post Processing: Simple Trace Analysis often used TCL and Pearl

Characteristics

NS-2 implements the following features

Router queue Management Techniques Drop Tail, RED,CBQ

Multicasting

Simulation of wireless networks

Developed by Sun Microsystem + UC Berkeley (Daedalus project)

Terrestrial (Cellular, Ad-hoc, GPRS, WLAN, BLUETOOTH)Satellite

NAM (Network Animator)

NAM provides a visual interpretation of the network topology created. The application was developed as a part of the VINT project. (figure 1.8) Its feature is:

Provides a visual interpretation of the network created

Can be executed directly from a Tcl script

Controls include play; stop fast forward, rewind, pause, a display speed controller button and a packet monitor facility

Presented information such as throughput, number packets on each link

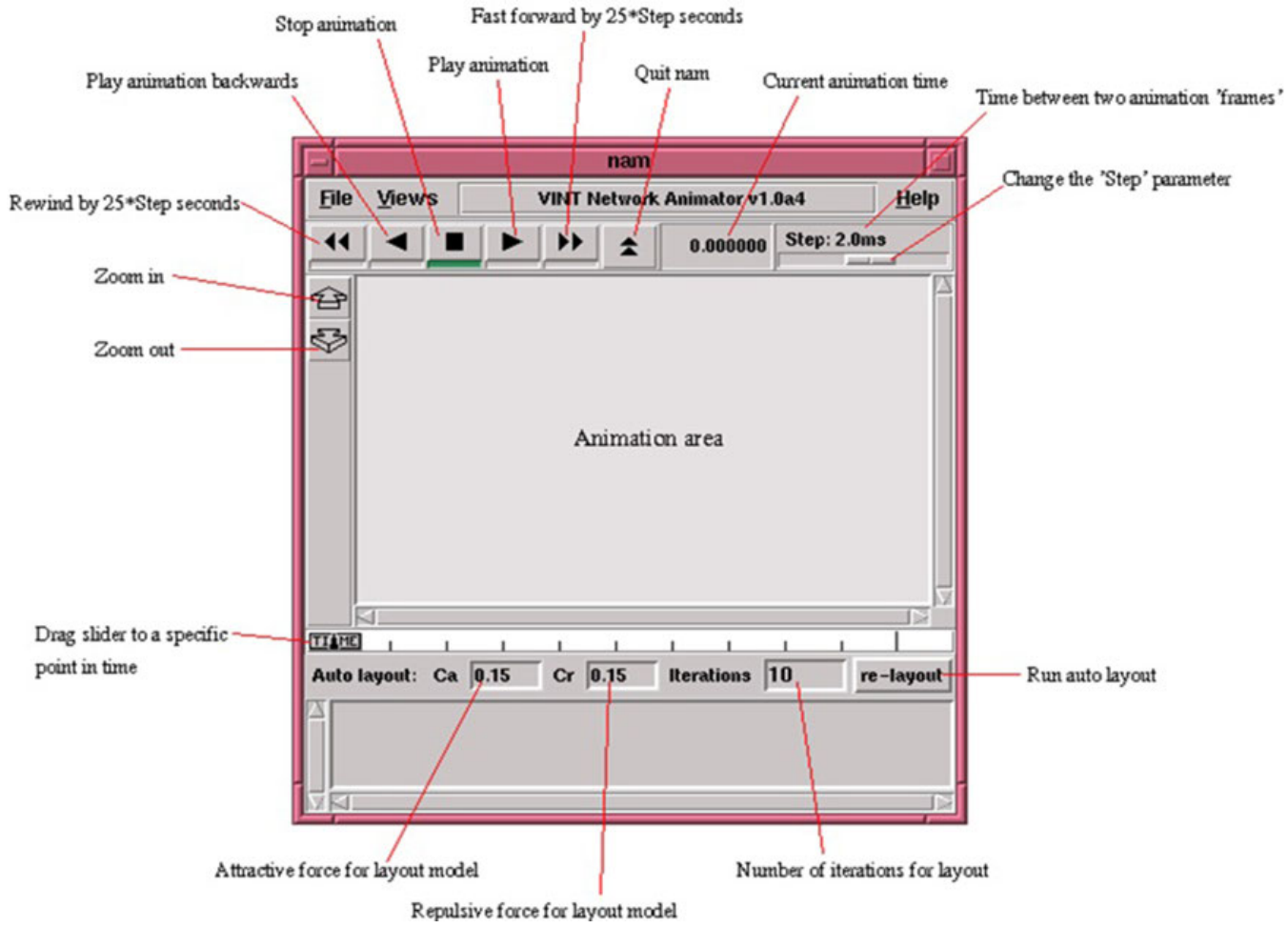


Figure 1.8: NAM

X Graph

X-Graph is an X-Window application that includes:

Interactive plotting, graphing Animated and derivatives.

To use Graph in NS-2 the executable can be called within a TCL script . This will then load a graph displaying the information visually displaying the information of the file produced from the simulation.

The output is a graph of size 800 x 400 displaying information on the traffic flow and time.

Simulation tool

NS2 are often growing to include new protocols. LANs need to be updated for new wired/wireless support. NS is an object-oriented simulator, written in C++, with an OTcl interpreter as a front-end. The simulator supports a class hierarchy in C++ and a similar class hierarchy within the OTcl interpreter (interpreted hierarchy). The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between classes in the interpreted.

Installation

Ns2 is a free simulation tool. It keeps running on different stages including UNIX (or Linux), Windows, and Mac frameworks.

Ns2 source codes are circulated in two structures:

all-in-one suite

the component-wise

With the all in one suite, clients get all the required parts along with some discretionary segments. This is essentially a suggested decision for the components. The package provides an " install " script which configures the NS2 environment and creates NS2 executable file using the " make " utility. The current all-in-one suite consists of the following main components:

NS release 2.30

Tcl/Tk release 8.4.13

OTcl release 1.12

TclCL release 1.18

And the following are the optional components:

NAM release 1.12: NAM is an animation tool for viewing network simulation traces and packet traces

Zlib version 1.2.3: This is the required library for NAM

Xgraph version 12.1: This is a data plotter with interactive buttons for panning, zooming, printing, and selecting display options

The idea of the component-wise approach is to obtain the above components and install them individually. This option saves considerable amount of downloading time and memory space. However, it could be troublesome for the beginners, and is therefore recommended only for experienced users.

Ns2 Installation on Ns2.35 In Ubuntu

Step 1:

Make sure that internet connectivity is good.

Download NS2.35 from http://www.isi.edu/nsnam/ns/

Step 2:

Make a new folder [ns] in /home/

Extracts downloaded files in above folder

Step 3:

Copy downloaded ns-allinone-2.35.tar.gz in /home/ns/

Step 4:

Open terminal

Step 5:

Run following commands:

cd ns-allinone-2.35

$ sudo apt-get update

$ sudo apt-get install build-essential

autoconfautomakelibxmu-dev gcc-4.4 g++4.4

Step 6:

Run following command

. /install

Step 7:

Run following command to install X – graph

$ sudo apt-get install xgraph

Run following command to install GNUPLOT

$ sudo apt-get install gnuplot

Run following command to install GEANY

$ sudo apt-get install geany

Step 8:

Set environment variables

run this command

gedit ~/.bashrc

Step 9:

run following command

source ~/.bashrc

Step 10:

Now you can run your ns with ns command:

The “%” symbol appears on the screen.

Type “ exit ” to quit.

CHAPTER 2

Tool Command Language

Tool command language (TCL) was developed by John Ousterhout in 1988. It is same as Python, in a wayas python is interpreted similarly TCL language is interpreted. When you lookat a timeline TCL seems to be appearing a way before Python came up. TCL is licensing BSD style.

TK: It is an extension to Tool command language.

Graphical user interface (GUI) is the framework for the TCL. Just like TCL, TK is interpreted for all platforms. It gives a GUI interface for tcl with support over tk. Tk support various languages such as:

Wish program

Windows Shell

NS2 is written in C and C++. If you do not know C++, then you can simulate a code by using a scripting language called as Tool Command Language (TCL) . Figure 2.1 , shows NS-2 Directory Structure.

Figure 2.1: NS-2 Directory Structure

Linux Commands

cd: Change directory

Syntax: cd directory name

ls: List the files in current directory

Syntax: ls

rm: Remove a file from directory

Syntax: rm filename

cp: Copying file from one directory to another

Syntax: cp filename directory name

pwd: For checking current directory

Syntax: pwd

ps: For viewing currently running processes on system

Syntax: ps

kill: For killing a process

Syntax: kill process_id

cat: For viewing file contents on terminal

Syntax: cat filename

clear: Clear the contents on terminal

Syntax: clear

gcc: For compiling C and C++ programs

Syntax: gccprogram_name.c

gedit: Create and open the file in text editor

Syntax: gedit filename

./ : For running object file

TCL Basics

TCL is an interpreted language and each instruction is a command in TCL program. TCL programming is used to write simulation script in NS2.

Variables in TCL

Set command is used to make a variable and assign value to them. This command is also used to fetch the value of variable.

Syntax: set

When you type the above command, it will create a variable with given name and assign value. If the value is not specified, then the above command will show the value stored in variable.

Consider an example:

set x 10

set name “MDSU”

set price 100.12

First command will create a variable name ‘ x ’ and value assign a value of 10 .

Second command will assign ‘MDSU’ to variable name.

Third command assign a float value to variable price.

Note: For deleting a variable ‘unset’ command is used.

Example: unset x

Print variable’s value

To use the Value of any variable, $ symbol is used. Consider an example, assign a value of x variable to another variable y.

Command: set y $x

Above command, will first replace $x by its value which is 10 then 10 will be assigned to variable y.

To print the value:

Command: puts “ HELLO MDSU ”

Output: HELLO MDSU

Above command will take cursor to a new line, if you want to print next message on the same line, then –no new line option has be specifies

Example 2.1:

puts “HELLO STUDENTS”

puts –nonewline “WELCOME”

puts “TO AJMER”

Output:

HELLO STUDENTS

WELCOME TO AJMER

Program 2.1:

```
set x 50

set y 100

puts “value of x is $x”

puts “value of y is $y”
```

Output 2.1:

Value of x is 50

Value of y is 20

Performing basic operations

In TCL, expr command used to perform operations. Expr command takes entire expression as an argument. See program 2.2 to add two variable x and y.

Program 2.2:

```
set x 10

set y 20

expr $x+$y
```

Output 2.2:

30

Expr command add both variable and print their value.

If you are assigning result to third variable z, then use the brackets. Command written inside the bracket are then replaced by its result.

Program 2.3:

```
Set x 10

Set y 20

Set z [expr $x+$y]

Puts “result of addition: $z”
```

Output 2.3:

Result of addition: 30

Question 2.1: How to execute TCL program in Linux.

Solution:

Open gedit text editor

```
puts “ Hello MDSU ”
```

Save this file with:.tcl extension (one.tcl)

Open terminal and change directory to desktop.(cd Desktop/)

Run the following command: tclshone.tcl

Output: Hello MDSU

Question 2.2: How to install and run TCL on windows.

Solution:

Install tcl .

Open the browser.

Search for tcl download.

Click on link which is active to download tcl .

Check your system configuration if it is 32 bits, then download first or in case of 64 bit download second.

Once download is done.

Double click on the download file and install it.

tcl will install on your system.

TCL is installed on windows.

Now run your program on windows, by opening command prompt

Write tclsh command

If you can see % symbol, it means tcl interpreted is running

% puts “Hello MDSU”

Output: Hello MDSU

You can write your program in a file with the .tcl \ extension and then run that file by using tclsh <file-name>

Example 2.2:

Open notepad

set a 10

puts “ variable value is $a ”

Save the file with one.tcl

Open command prompt and write cd Desktop/

Write tclshone.tcl

Output: variable value is 10

Example 2.3:

Check for variable exit

set x 1

puts [info exist x]

run tclshshow.tcl

Output: 1

Example 2.4:

To Unset a variable after assigned in a variable

```
set x 10
unset x
puts [info exist x]
```

run tclshshow.tcl

Output: 0

Example 2.5: Program to calculate the area of circle

```
set radius 7
set pie [expr 22/7]
set area [expr $pie*$radius+$radius]
puts " Area of circle is : $area "
```

Output: Area of circle is: 154

If-else statement

Syntax:

```
if {condition} {
if-body
}
```

Or

```
If [expr condition] {
If-body
}
```

Example 2.6:

```
set x 10
if {$x < 50}
{
puts "x is less than 50"
}
if [expr $x ==20]
{
puts "x is equal to 20"
}
else
{
puts "x is not equal to 20"
```

}

% set a 5

5

% puts $a

5

% set a [expr 5+5]

10

% puts $a

10

% if {$a<15}{puts “True”} else {puts “False”}

True

% if {$a>15}{puts “True”} else {puts “False”}

False

Loop

In TCL, the syntax of while loop is in similar to C, with different condition specification.

While loop

Syntax:

```
while {condition} {
while-body
}
```

Note: Opening braces for while must be in the same line of while condition, if you place opening brace at new line it will create error.

Example 2.7:

```
set s 0
while {$s < 100} {
puts $s
set s [expr $s+2]
}
```

Example 2.8:

```
set i 0    //created variable i and assign it 0 value
while {$i< 10} {
puts $i
set i [expr $i+1]       // incri
}
```

Save this file as show.tcl

Run as tclshshow.tcl

Output:

0

1

2

3

4

5

6

7

8

9

For loop

Syntax:

for {initialization} {condition} {increment/decrement} {

}

Example 2.9:

for {set i 0} {$i<10} {incri} {

puts $i

}

Example 2.10:

% for {set b 1} {$b<10} {incr b} {

puts “b is $b”

}

Output:

b is 1

b is 2

b is 3

b is 4

b is 5

b is 6

b is 7

b is 8

b is 9

In the above program:

In the first part of setting value, the value of variable b is 1

Second part has a condition in which the value of b is less than 10

In the third part we increment the value of b

List

List is an arrangement of elements in a sequential manner.

Example 2.11:

```
set a {12 23 34}

foreach t $a {

puts $t

}
```

Array

Array is a collection of homogenous data. TCL support named array in which string is given as named variable

Example 2.12:

```
Set a(0) 10

Set a(1) 20

Puts $a(0)
```

Output:

10

Procedure

Multiple commands can be combined to make a new command, this can be done by making a procedure. Procedure is similar to the function in 'C'

Syntax:

```
proc { }{

Procedure-body

}
```

Example 2.13:

```
proc show { } {     //opening braces in the same line

puts "welcome to AJMER"

}
```

Run: tclshshow.tcl

Output: welcome to AJMER

Example 2.14:

```
proc show { a } {     //procedure argument

puts "variable value"

puts $a;

}
```

show 10 //procedure call

Run: tclshshow.tcl

Output: variable value 10

Example 2.15: Procedure to add two numbers

proc show { a b } {

set c [expr $a+$b] // In tcl programming, in any operation expr command is used. As, by default in tcl everything is string. In order to make them integer and perform operation on them, we use expr command. [ ] bracket are used to replace the entire expression [expr $a + $b] to output i.e. to c

return $c

}

puts [show 10 20]

Run: tclshshow.tcl

Output: 30

Example 2.16: Program to make procedure for factorial

proc fact { a } {

set fact 1

for {set i 1} {$i<=$a} {incri} {

set fact [expr $fact *$i]

}

puts $fact

}

fact 5 #calling procedure

Output: 120

Example 2.17: For default value

proc show { a {b 4} } {

puts $a

puts $b

}

show 10 20

show 10

Run: tclshshow.tcl

Output: 10

20

10

Global variable in Procedure

In procedure all the variable is known as local. If you want to access some variable which are created outside of the procedure, then the global command is used.

Example 2.18:

```
set x 20

proc demo { } {

global x

set x 30

}

demo

puts $x
```

Output: 20

Example 2.19: To check the preference of global variable over local variable

```
set x 10     //global variable

proc demo { } {

set x 20     //local variable

}

demo

puts $x
```

Run: tclshshow.tcl

Output: 10

Example 2.20: To make global variable as a local

```
set x 10

proc demo { } {

global x

set x 20

}

demo

puts $x
```

Run: tclshshow.tcl

Output: 20

Example 2.21: To use Color (::) symbol for a global

```
set x 10

Proc demo { } {

Set ::x 20     //use:: for global variable

}
```

demo

Puts $x

Run: tclshshow.tcl

Output: 20

Example 2.22: How different value can be show in output.

set x 10

x=10

set y 120

set z 230

puts $x

puts $y

puts $z

Run: tclshshow.tcl

Output: 10

120

230

Example 2.23: To use integer, sting, float value and print them

set x 10

x=10

set y “MDSU”

set z 5.777

puts $x // puts [set x]

puts $y //puts [set y]

puts $z

Run: tclshshow.tcl

Output: 10

MDSU

5.77

File Handling

Before performing any operation on file, that file need to be opened. In tcl, open command is used to open a channel with file to perform the read/write operation. File can be opened in various modes according to which an operation to be performed is allowed. These modes are

- R ➢ For reading // *file should exist*
- R+ ➢ To read and write // *file should exist*
- W ➢ For writing // *file created or replaced*
- W+ ➢ To read and write // *file created or replaced*
- A ➢ To append // *file created or appended*
- A+ ➢ To read and append // *file created or appended*

Example 2.24: For opening file

Open

Open one.txt r

For assigning open file handle to variable

Set f in [open one.txt r]

Above code will open " one.txt " file in read mode, which means only read operation can be performed on this file. Last command in above code open " one.txt " in read mode and then assign it a variable f, which can be used further for file processing.

Writing file

puts command is used for writing data in file

Syntax:

puts <data|variable value>

set f [open one.txt w]

puts $f "Welcome to tcl"

close $f

Reading file

gets command is used for reading data from file

Syntax:

gets

set f [open one.txt r]

gets $f data

puts $data

close $f

CHAPTER 3

Writing and Executing a TCL Scripting with NS2

Simulation Script

Create Simulator Class Object

Every simulation script in NS2 requires a simulator object. It works as scheduler which is used to schedule events.

Code:

Create scheduler

set ns [new Simulator]

Above command will create a Simulator object and assign it a variable ns.

Store results in File

After creating a Simulator object, we need to store its results. For this purpose, open command is used to open a file. For tracing, trace-all and namtrace-all procedures of Simulator class are called.

Code:

Tracing simulation results

set fin [open result.dat w]

$ns trace-all $fin

Tracing for Network Animator (NAM)

set nfin [open namtrace.nam w]

$ns namtrace-all $nfin

First line opens a file result.dat in write mode (W) and assigns it to variable fin. This variable can be used to perform operation on file result.dat . Next line calls an inst proc trace-all and pass reference to file as a parameter. Trace-all procedure stores all simulation result in result.dat file. Similarly, out.nam file is opened which is used by network animator to show packet transmission.

Create Nodes

After storing the result, we need to create required number of nodes. NS2 support wired and wireless types of node. In our example we are considering wired node.

Code:

Create a node

set n0 [$ns node]

Create another node

set n1 [$ns node]

If there is need to create large number of nodes, then loop can be used. Consider an example:

Create 10 Nodes

for {set i 0} { $i<10 } { incri} {

set n($i) [$ns node]

}

Create Agent (TCP or UDP)

In NS2, there are various agents which we need to attach with nodes. Some of them are TCP, TCP/Reno, TCPSink, and UDP . For our program, as we want to send data over TCP so one node will be Sender and another node will be receiver. Sender needs to be attached with TCP and receiver needs to be attached with TCPSink .

Code:

Create TCP agent and attach it to first node

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

Create TCP receiver and attach it to second node

set tcp1 [new Agent/TCPSink]

$ns attach-agent $n1 $tcp1

Connect Nodes

Nodes need to be connected to each other, and the connections among them are known as links . These links can be simplex or duplex . For each link bandwidth, delay, and type of queue need to be specified.

Code:

Connect both nodes with 1.5 Mbps bandwidth, 5 milli seconds delay, and DropTail queue. DropTail works according to First In First Out (FIFO) pattern .

$ns duplex-link $n0 $n1 1.5Mb 5ms DropTail

Connect Agents on Both Nodes

Every node can have any number of agents. We have to connect these agents to all nodes. In our case Sender agent (TCP) need to be connected with Receiver (TCPSink).

Code:

Connect tcp0 and tcp1 agents

ns connect $tcp0 $tcp1

Setup a Ftp Application over TCP

Till now, we have created node, connected them, setup a TCP connection. Now we need to associate a traffic generator to see the packet transmission, for which CBR, FTP are available.

Code:

Create a FTP object

set ftp0 [new Application/FTP]

Attach Ftp Application with Agent and Schedule Events

In this we will attach FTP application with TCP agent.

Code:

Attach ftp application with agent tcp0

$ftp attach-agent $tcp0

Schedule events

$ns at 0.2 "$ftp start"

$ns at 2.0 "$ftp stop"

In the above code, FTP application is attached with TCP agent of the first node and events are scheduled. Event can be scheduled by using the following syntax.

Syntax:

at

In our case we have started FTP application at 0.2 milli second and stopped it at 2milli second.

Flush Buffer and Start NAM

At the end we need to flush buffers and start network animator(Optional). For this purpose, we will make a function and run that function at the end.

Code:

Finish procedure to perform operation at the end of simulation

proc finish { } {

Mapping of global variable into local variable

global ns fin nfin

Flush all buffers

$ns flush-trace

close file objects

close $fin

close $nfin

execute nam process in background

exec namout.nam&

Terminate current process

exit 0

}

Now, we will schedule execution of procedure to finish on time when we want simulation to terminate.

Calling function:

$ns at 2.2 "finish"

Start Simulation

Finally, start the simulation by using scheduler.

$ns run

First Simulation Scenario

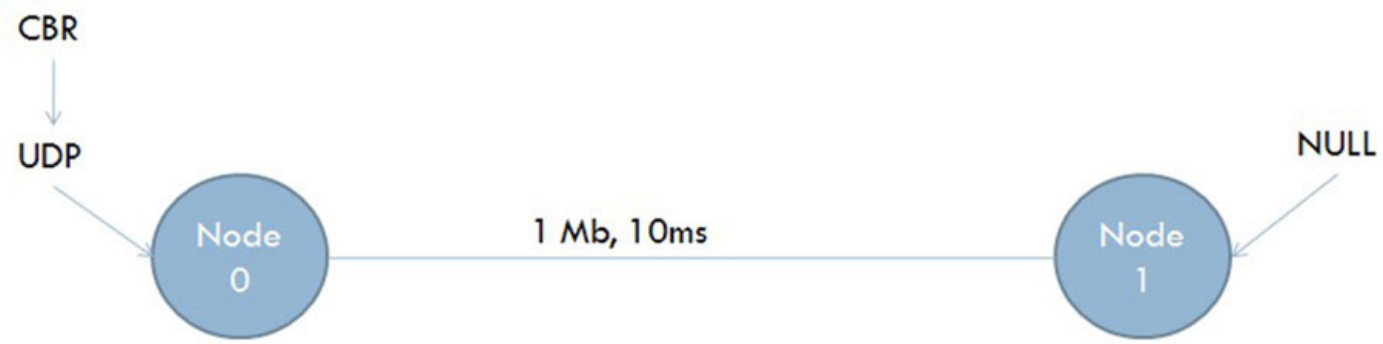


Figure 3.1: First Simulation Scenario

Simulation Script

Create a simulator object

set ns [new Simulator]

Open the nam trace file

set nf [open s1.nam w]

$ns trace-all $nf1

Define a finish procedure

Proc finish { }

{

global ns nf

$ns flush-trace

Close the nam trace file

close $nf

Close the event trace file

Close $nf1

Execute nam

Exec nam s1.nam &

exit 0

}

Create two nodes

Set n0 [$ns node]

Set n1 [$ns node]

Create a duplex link between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms Drop Tail

Create a UDP agent

set udp0 [new Agent/UDP]

Attach udp agent to node n0

$ns attach-agent $n0 $udp0

Create a CBR traffic source and attach it to udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0,005

$cbr0 attach-agent $udp0

Create a Null agent (a traffic sink) and attach it to node n1

set null0 [new Agent/Null]

$ns attach-agent $n1 $null0

Connect the traffic source with the traffic sink

$ns connect $udp0 $null0

Schedule events for the CBR agent

$ns at 0.5 “$cbr0 start”

$ns at 4.5 “$cbr0 stop”

Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 “finish”

Run the simulation

$ns run

Save the simulation script in specific folder.

Open the terminal and go up to specific folder.

Run the simulation script,

ns: command to run simulation script.

Syntax: ns filename.tcl

Example: ns First*script*wired.tcl

Run the nam file,

nam: command to run animation file

Syntax: nam filename.nam

Example: nam s1.nam

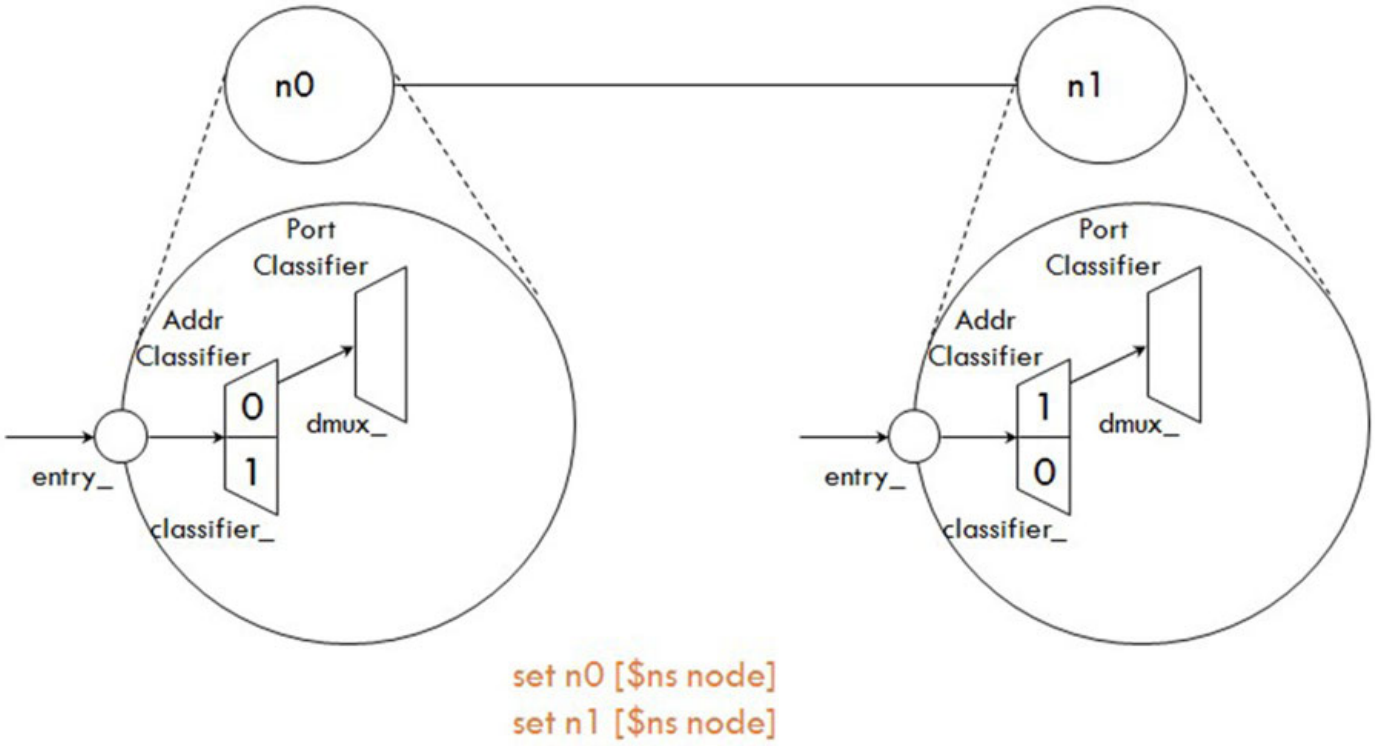


Figure 3.2: Flow of Simulation (NS-Node)

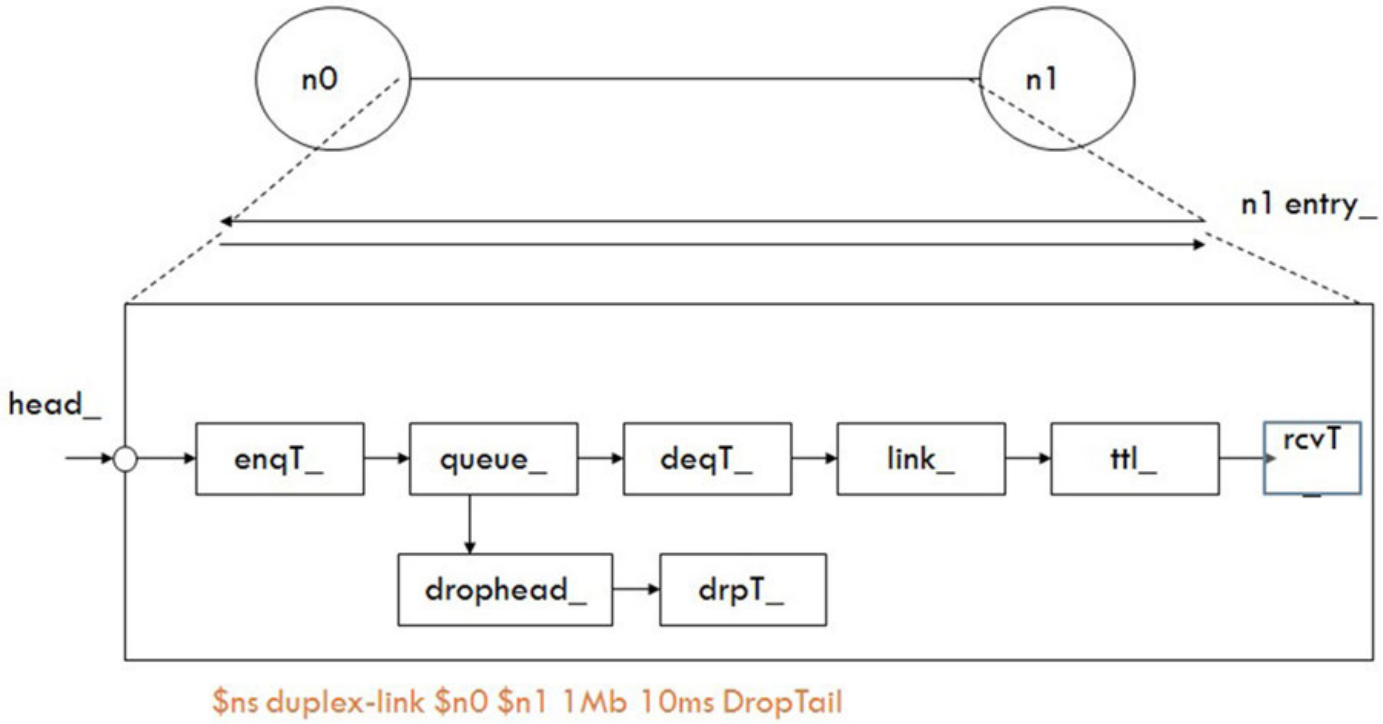


Figure 3.3: Flow of Simulation (Network Topology – Link)

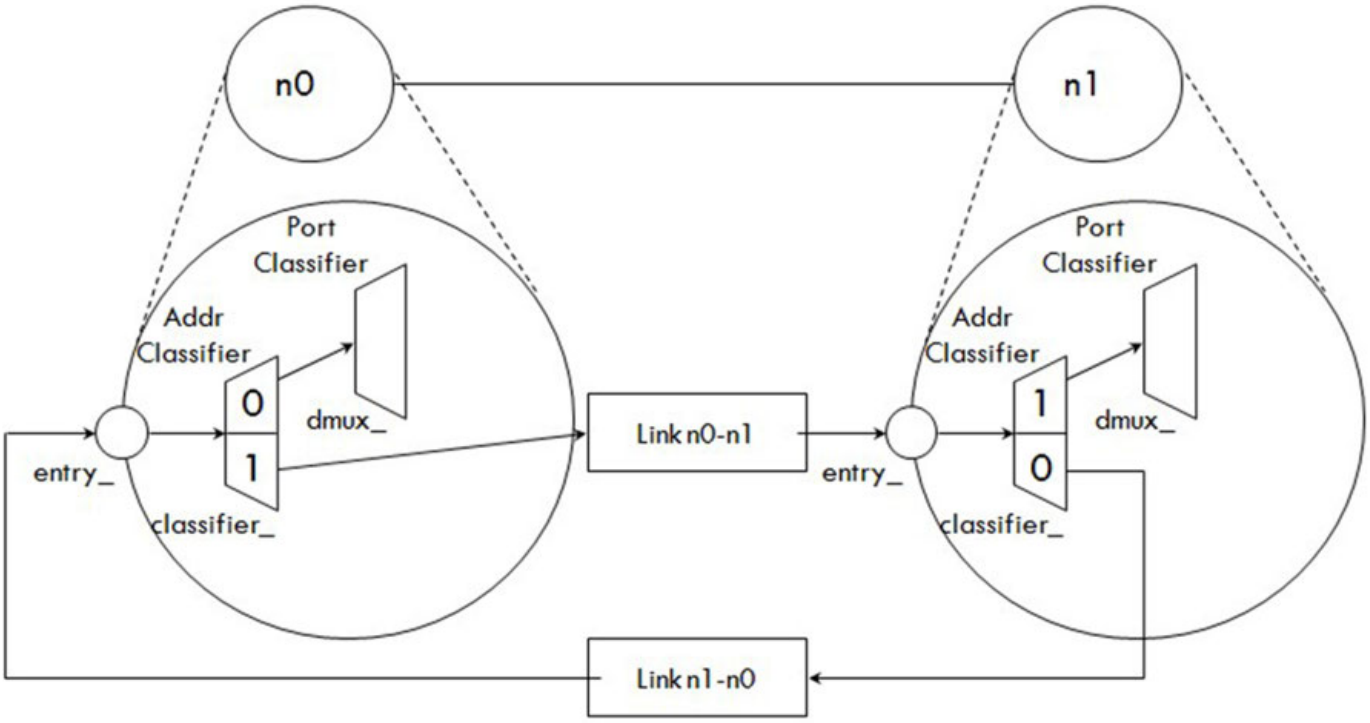


Figure 3.4: Flow of Simulation (Routing)

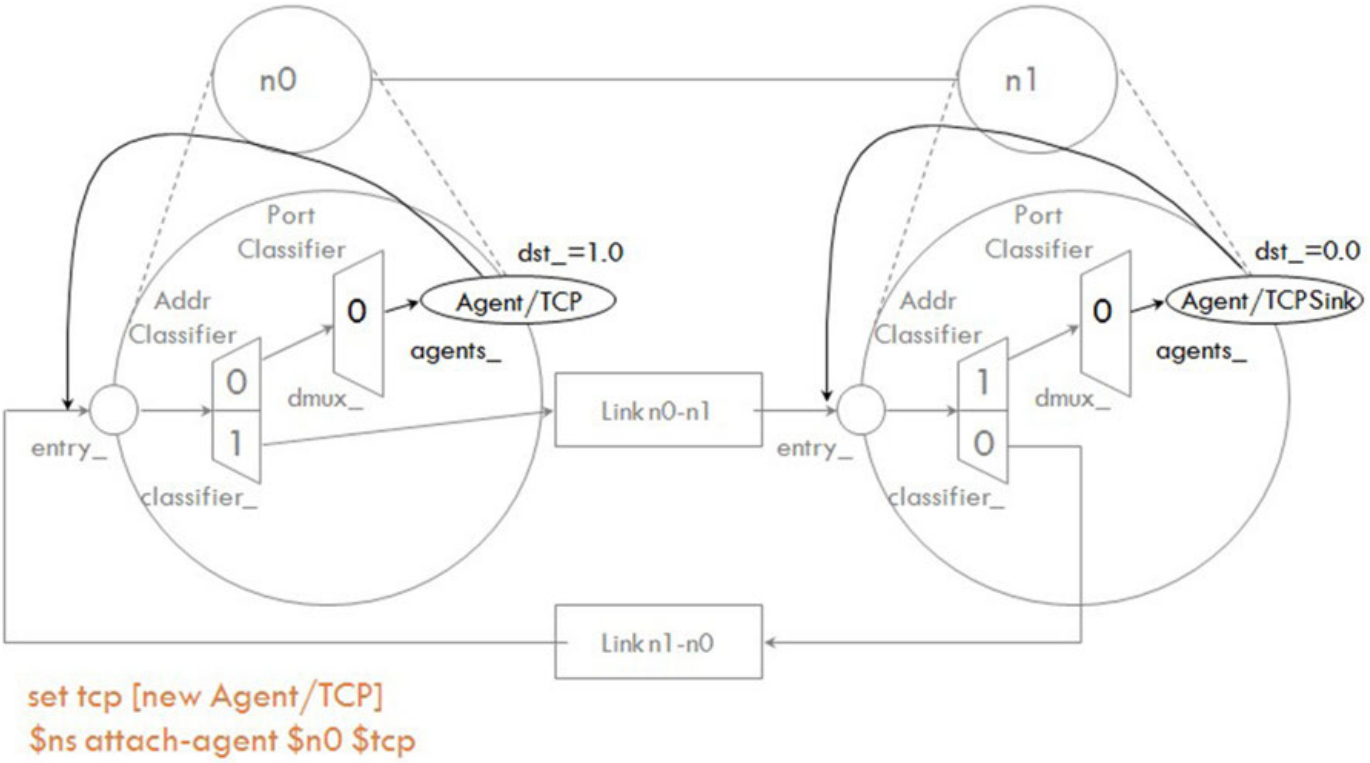


Figure 3.5: Flow of Simulation (Transport)

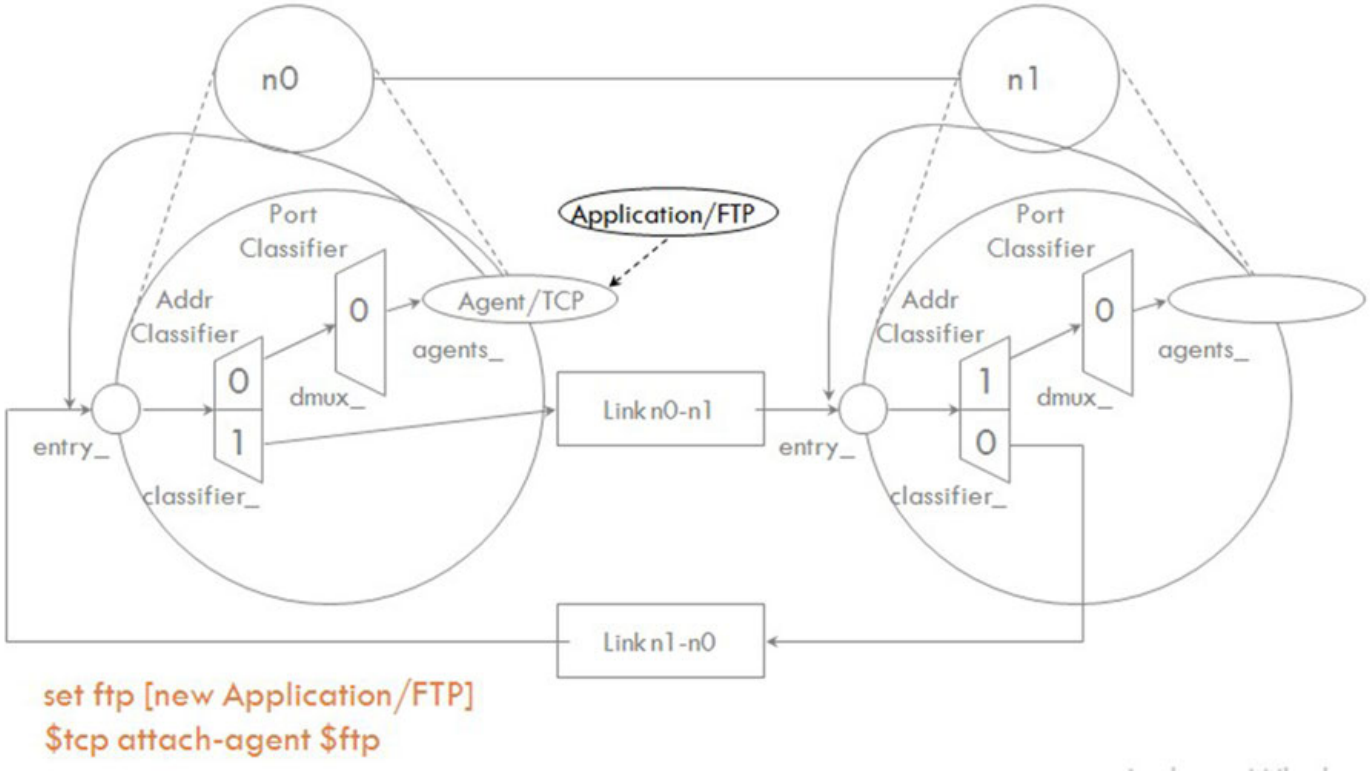


Figure 3.6: Flow of Simulation (Application)

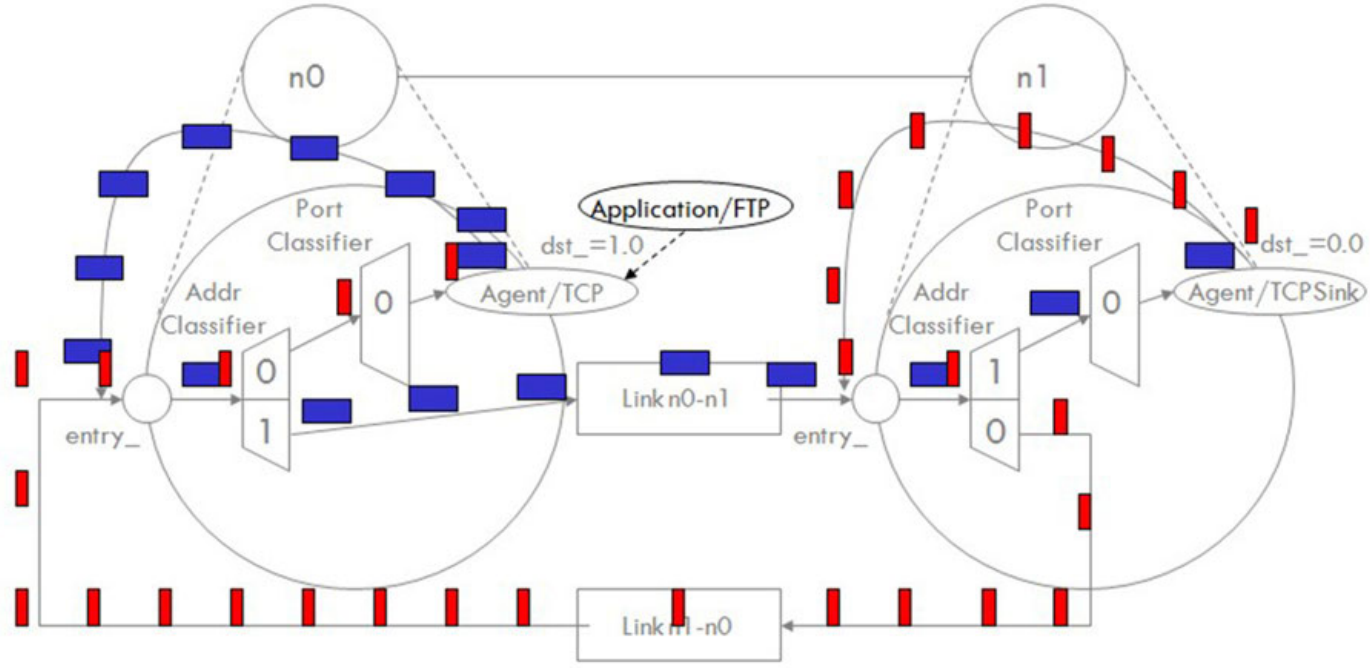


Figure 3.7: Flow of Simulation (Packet Flow)

Second Simulation Scenario

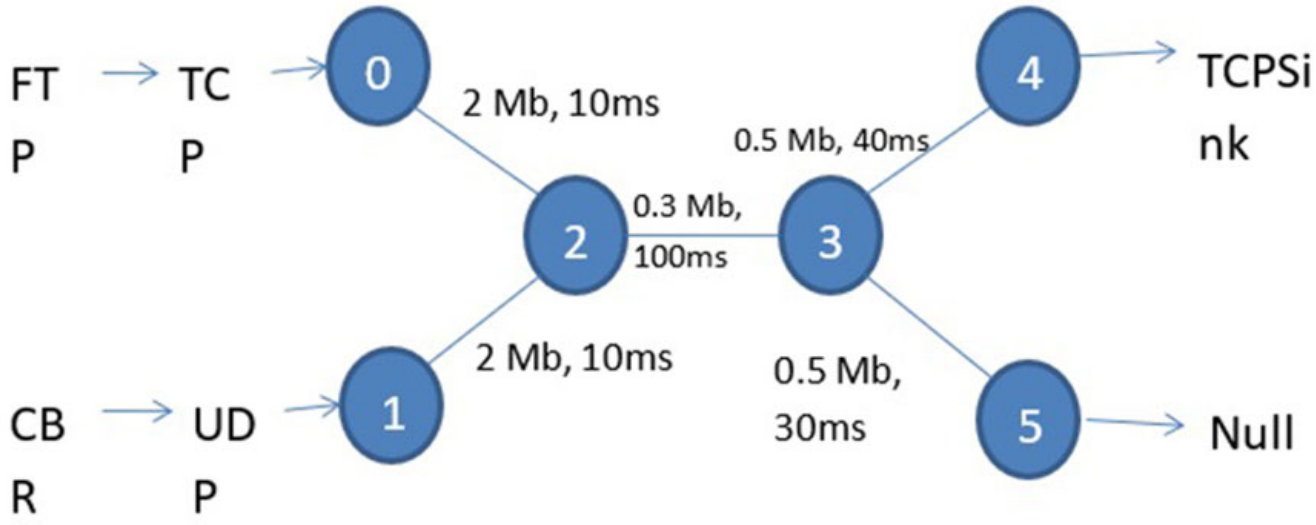


Figure 3.8: Second Simulation Scenario

Simulation Script 2

Create a simulator object

set ns [new Simulator]

Open the nam trace file

$ns color 1 Blue

$ns color 2 Red

Open the Event trace files

set file1 [open out tr w]

$ns trace-all $file1

Open the NAM trace file

Set file2 [open outman w

$ns namtrace-all $file2]

Define a finish procedure

proc finish { }

{

global ns file1 file2

$ns flush-trace

close $file1

close $file2

exec nam out nam& exit 0

}

Create six nodes

Set n0 [$ns node]

Set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

Set n5 [$ns node]

Label to the node n1 and node n2

$ns at 0.1 “$ns label “CBR””

$ns at 1.0 “$ns label “FTP””

Create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 0.3Mb 100ms DropTail

$ns duplex-link $n3 $n2 0.3Mb 100ms DropTail

$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail

$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

Give node position (for NAM)

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n3 $n2 orient left

$ns duplex-link-op $n3 $n4 orient right-up

$ns duplex-link-op $n3 $n5 orient right-down

Set Queue Size of link (n2-n3) to 40

$ns queue-limit $n2 $n3 40

Set up a TCP connection

Set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

Set sink [new Agent/TCP Sink]

$ns attach-agent $n4 $sink

$ns connect $tcp $sink

$tcp set fid_1

$tcp set window_ 8000

$tcp set packetSize_ 552

Set up a FTP over TCP connection

Set ftp [new Application FTP]

$ftp attach-agent $tcp

$ftp set type_ FTP

Set a UDP connection

Set upd[new Agent/UDP]

$ns attach-agent $n1 $upd

Set null [new Agent $n5 $null]

$ns connect $udp $null

$udp set fid_ 2

Setup a CBR over UDP conTraffic CBR]

$cbr attach-agent $upd

$cbr set type_ CBR

$cbr set packet*size* 1000

$cbr set rate_ 0.01 mb

$cbr set random_ false

scheduling the event

$ns at 0.1 “$cbr start”

$ns at 1.0 “ftp start”

$ns at 624.0 “$ftp stop”

$ns at 624.5 “$cbr stop”

Trace Congestion Window and RTT

Set file [open cwnd_rtt tr w]

$tcp attach $file

$tcp trace cwnd_

$tcp trace rtt_

Call finish procedure

$ns at 625.0 “finish”

Run the simulation

$ns run

Node Orientation

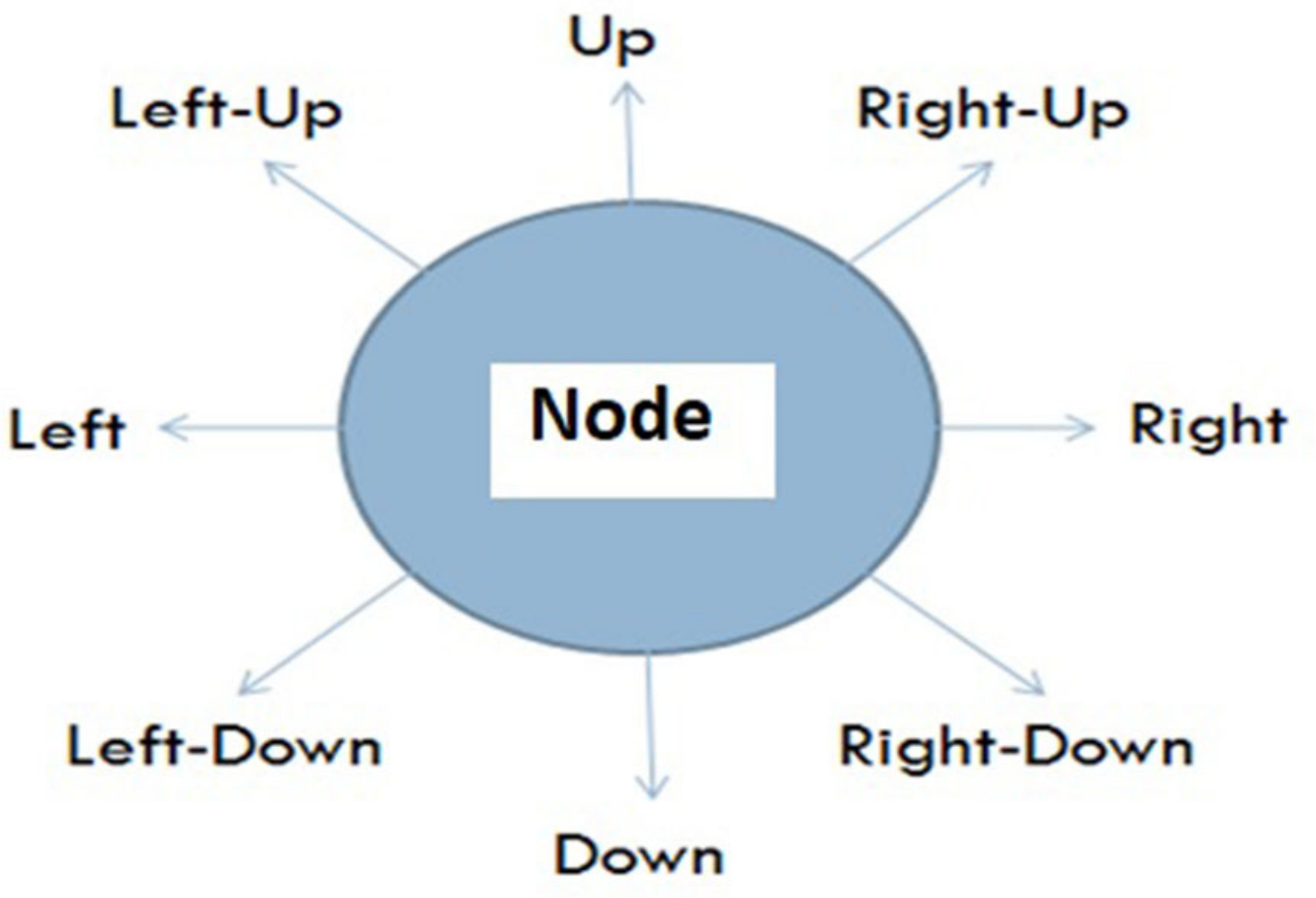


Figure 3.9: Node Orientation

Node Commands

$ns node [ ]

$ns node-config -<config-parameters><optional-val>

$node id

$node node-addr

$node reset

$node agent

$node entry

$node attach

$node detach

$node neighbors

$node add-neighbor

$node add-route

$node alloc-port

$node incr-rgtable-size

Link Commands

$ns simplex-link

$ns duplex-link args>

$ns simplex-link-op

$ns duplex-link-op

$ns lossmodel

$link head

$link link

$link add-to-head

$link queue

$link cost

$link cost?

$link if-lable?

$link up

$link down

$link up?

$link all-connectors

Simulator Commands

set ns [new Simulator]

set now [$ns now]

$ns halt

$ns run

$ns at

$ns cancel

$ns flush-trace

$ns use - scheduler

$ns after

$ns clearMemTrace

$ns is-started

$ns dumpq

Trace Related Commands

$ns trace-all <trace-file>

$ns namtrace-all

$ns namtrace-all-wireless

$ns nam-end-wireless

$ns flush-trace

$ns create-trace <optional:op>

$ns trace-queue <optional: file>

$ns namtrace-queue <optional: file>

$ns drop-trace

$ns monitor-queue <optional: sampleinterval>

$link trace-dynamics

NAM Commands

$ns color <color-id>

$ns trace-annotate

$ns set-animation-rate

Third Simulation Scenario

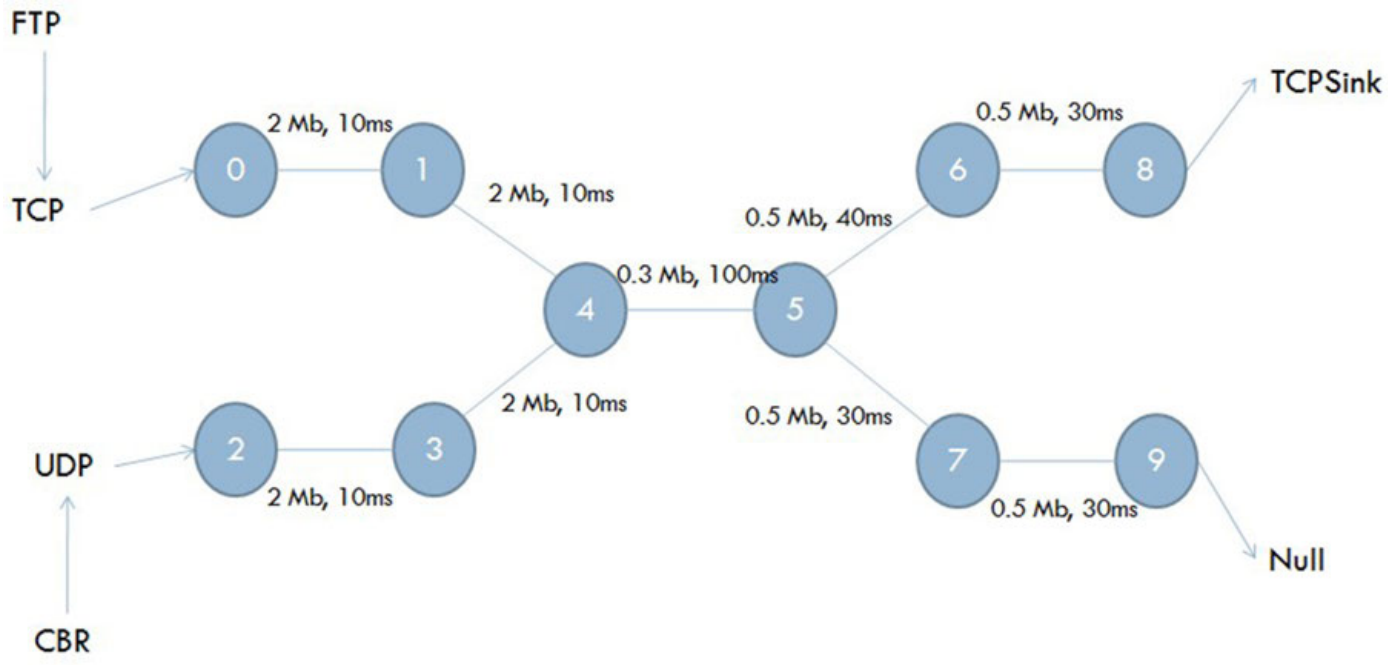


Figure 3.10: Third Simulation Scenario

Simulation Script 3

Create a simulator object

set ns [new Simulator]

Open the nam trace file

$ns color 1 Blue

$ns color 2 Red

Open the Event trace files

set file1 [open out tr w]

$ns trace-all $file1

Open the NAM trace file

Set file2 [open outman w

$ns namtrace-all $file2]

Define a finish procedure

proc finish { }

{

global ns file1 file2

$ns flush-trace

close $file1

close $file2

exec nam out. nam&

exit 0

}

Create ten nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

Set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

Set n9 [$ns node]

Label to the node n1 and node n2

$ns at 0.1 “$ns label \“CBR\””

$ns at 1.0 “$ns label \“FTP\””

Create links between the nodes

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link $n1 $n3 2Mb 10ms DropTail

$ns duplex-link $n1 $n4 2Mb 100ms DropTail

$ns duplex-link $n3 $n4 2Mb 100ms DropTail

$ns duplex-link $n4 $n5 0.3Mb 100ms DropTail

$ns duplex-link $n5 $n4 0.3Mb 100ms DropTail

$ns duplex-link $n5 $n6 0.5Mb 100ms DropTail

$ns duplex-link $n6 $n8 0.5Mb 40ms DropTail

$ns duplex-link $n5 $n7 0.5Mb 30ms DropTail

$ns duplex-link $n7 $n9 0.5Mb 30ms DropTail

Set Queue Size of the link (n4-n5) to 10

$ns queue-limit $n4 $n5 10

Set up a TCP connection

Set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

Set sink [new Agent/TCP Sink]

$ns attach-agent $n8 $sink

$ns connect $tcp $sink

$tcp set fid_1

$tcp set window_8000

$tcp set packetSize_ 552

Set up a FTP over TCP connection

Set ftp [new Application FTP]

$ftp attach-agent $tcp

$ftp set type_ FTP

Set a UDP connection

Set upd[new Agent/UDP]

$ns attach-agent $n2 $udp

Set null [new Agent/Null]

$ns attach-agent $n9 $null

$ns connect $udp $null

$udp set fid_ 2

Setup a CBR over UDP connection

Set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type_ CBR

$cbr set packet*size* 1000

$cbr set rate_ 0.01 mb

$cbr set random_ false

scheduling the event

$ns at 0.1 “$cbr start”

$ns at 1.0 “ftp start”

$ns at 624.0 “$ftp stop”

$ns at 624.5 “$cbr stop”

Trace Congestion Window and RTT

Set file [open cwnd_rtt tr w]

$tcp attach $file

$tcp trace cwnd_

$tcp trace rtt_

Call finish procedure

$ns at 625.0 “finish”

Run the simulation

$ns run

Wired file format

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)
- : dequeue (at queue)
d : drop    (at queue)

src_addr : node.port (3.0)
dst_addr : node.port (0.0)
```

Event	Time	From-Node	To-Node	Pkt-Type	Pkt-Size	Flags	Fid	Source-address	Destination address	Sequence number	Pkt-id
-	1.06	0	2	tcp	1040	----	1	0.0	3.0	2	124
r	1.07	1	2	cbr	1000	----	2	1.0	3.1	120	122
+	1.07	2	3	cbr	1000	----	2	1.0	3.1	120	122
d	1.07	2	3	cbr	1000	----	2	1.0	3.1	120	122

CHAPTER 4

Practical Examples for Wired Program in NS2

Program 4.1: Write a Program to create a node in NS2.

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec namout.nam&
exit 0
}
set n0 [$ns node]
$n0 set X_ 200
$n0 set Y_ 400
$ns at 5.0 "finish"
$ns run
```

Output 4.1:

Figure 4.1: One node creation

Program 4.2: Write a Program to create two nodes in NS2.

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec namout.nam&
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
$n0 set X_ 200
$n0 set Y_ 400
$n1 set X_ 300
$n1 set Y_ 400
$ns at 5.0 "finish"
$ns run
```

Output 4.2:

Figure 4.2: Two node creation

Program 4.3: Write a Program to create three nodes in NS2.

```
set ns [new Simulator]
```

```
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec namout.nam&
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$n0 set X_ 200
$n0 set Y_ 400
$n1 set X_ 300
$n1 set Y_ 400
$n2 set X_ 500
$n2 set Y_ 400
$ns at 5.0 "finish"
$ns run
```

Output 4.3:

Figure 4.3: Three node creation

Program 4.4: Write a Program to create three node and join two of them in NS2.

```
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 namout.nam&

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

$ns duplex-link $n0 $n1 100Mb 100ms DropTail

$n0 set X_ 200

$n0 set Y_ 400

$n1 set X_ 300

$n1 set Y_ 400

$n2 set X_ 500

$n2 set Y_ 400

$ns at 5.0 "finish"

$ns run
```

Output 4.4:

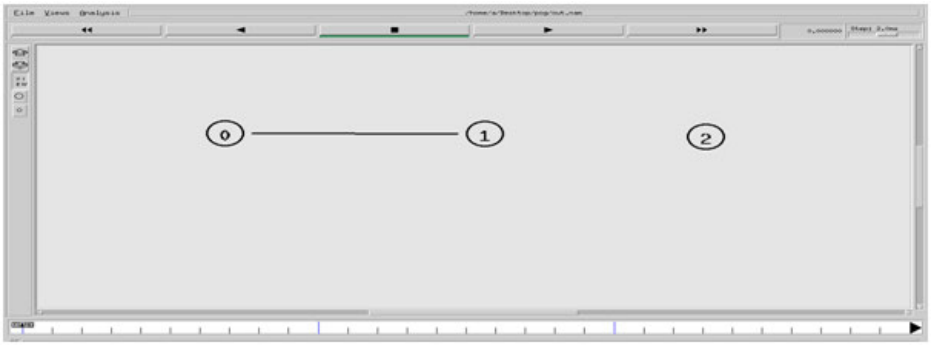


Figure 4.4: Four node creation

Program 4.5: Write a Program to create three nodes with multiple color and join three nodes in NS2.

```
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 namout.nam&

exit 0

}

set n0 [$ns node]

$n0 color "Red"

set n1 [$ns node]

$n1 color "Blue"

set n2 [$ns node]

$n2 color "Green"

$ns duplex-link $n0 $n1 100Mb 100ms DropTail

$ns duplex-link $n1 $n2 100Mb 100ms DropTail

$n0 set X_ 200

$n0 set Y_ 400

$n1 set X_ 300

$n1 set Y_ 400

$n2 set X_ 500

$n2 set Y_ 400

$ns at 5.0 "finish"

$ns run
```

Output 4.5:

Figure 4.5: Join three node creation

Program 4.6: Write a Program to create two node and data transfer between two nodes inNS2.

```
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 namout.nam&

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$ns duplex-link $n0 $n1 10Mb 10ms DropTail

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n1 $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize_ 500

$cbr1 set interval_ 0.005

$cbr1 attach-agent $udp1

set null1 [new Agent/Null]

$ns attach-agent $n0 $null1

$ns connect $udp1 $null1

$ns at 0.5 "$cbr0 start"

$ns at 2.5 "$cbr0 stop"

$ns at 0.5 "$cbr1 start"

$ns at 2.5 "$cbr1 stop"

$ns at 5.0 "finish"

$ns run
```

Output 4.6:

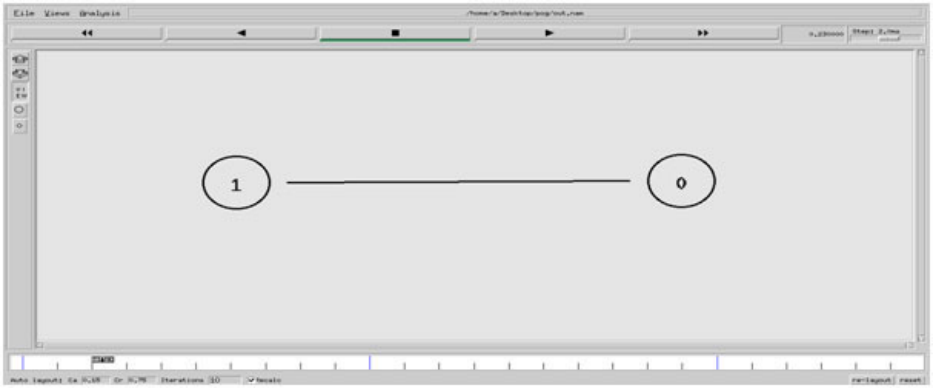


Figure 4.6 (a): Two node creation data transfer (I)

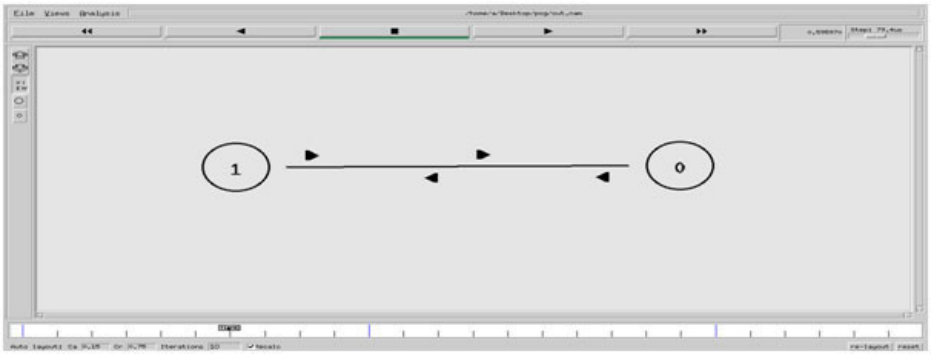


Figure 4.6 (b): Two node creation data transfer (II)

Program 4.7: Write a Program to create three nodes and data transfer between them in NS2.

```
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 namout.nam&

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$ns duplex-link $n0 $n1 10Mb 10ms DropTail

$ns duplex-link $n1 $n0 10Mb 10ms DropTail

set udp0 [new Agent/UDP]

set udp1 [new Agent/UDP]

$ns attach-agent $n0 $udp0
```

```
$ns attach-agent $n1 $udp1

set cbr0 [new Application/Traffic/CBR]

set cbr1 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr1 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr1 set interval_ 0.005

$cbr0 attach-agent $udp0

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]

set null1 [new Agent/Null]

$ns attach-agent $n1 $null0

$ns attach-agent $n0 $null1

$ns connect $udp0 $null0

$ns connect $udp1 $null1

$ns at 0.5 "$cbr0 start"

$ns at 0.5 "$cbr1 start"

$ns at 2.5 "$cbr0 stop"

$ns at 2.5 "$cbr1 stop"

$ns at 5.0 "finish"

$ns run
```

Output 4.7:

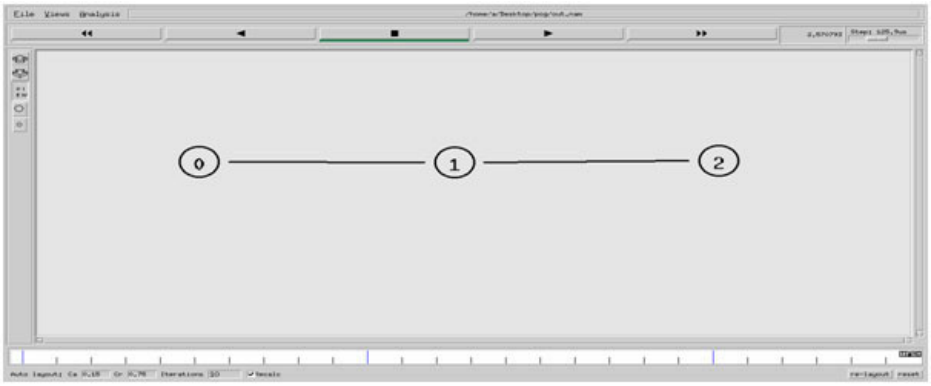


Figure 4.7(a): Data transfer between three nodes (I)

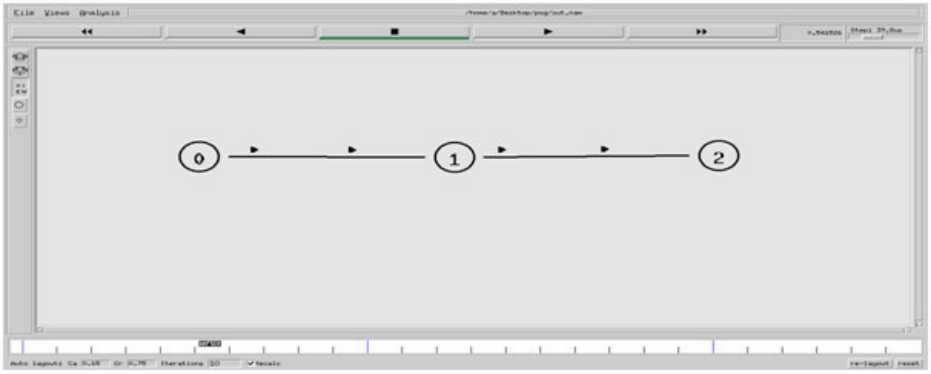


Figure 4.7(b): data transfer between three nodes (II)

Program 4.8: Write a program to data transfer between four nodes in NS2.

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec namout.nam&
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set proto rlm
$ns color 1 red
$ns color 2 blue
$ns color 3 yellow
$ns color 4 green
$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n2 10Mb 10ms DropTail
$ns duplex-link $n2 $n3 10Mb 10ms DropTail
$ns duplex-link $n3 $n0 10Mb 10ms DropTail
$ns duplex-link $n0 $n3 10Mb 10ms DropTail
$ns duplex-link $n3 $n2 10Mb 10ms DropTail
```

```
$ns duplex-link $n2 $n1 10Mb 10ms DropTail

$ns duplex-link $n1 $n0 10Mb 10ms DropTail

set udp0 [new Agent/UDP]

set udp1 [new Agent/UDP]

set udp2 [new Agent/UDP]

set udp3 [new Agent/UDP]

$ns attach-agent $n0 $udp1

$ns attach-agent $n1 $udp0

$ns attach-agent $n1 $udp2

$ns attach-agent $n2 $udp1

$ns attach-agent $n2 $udp3

$ns attach-agent $n3 $udp2

$ns attach-agent $n3 $udp0

$ns attach-agent $n0 $udp3

$ns attach-agent $n0 $udp1

$ns attach-agent $n1 $udp2

$ns attach-agent $n2 $udp3

$ns attach-agent $n3 $udp0

$ns attach-agent $n0 $udp3

$ns attach-agent $n3 $udp2

$ns attach-agent $n2 $udp1

$ns attach-agent $n1 $udp0

set cbr0 [new Application/Traffic/CBR]

set cbr1 [new Application/Traffic/CBR]

set cbr2 [new Application/Traffic/CBR]

set cbr3 [new Application/Traffic/CBR]

$ cbr0 set packetSize_ 500

$cbr1 set packetSize_ 500

$cbr2 set packetSize_ 500

$cbr3 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr1 set interval_ 0.005

$cbr2 set interval_ 0.005

$cbr3 set interval_ 0.005

$cbr0 attach-agent $udp0
```

```
$cbr1 attach-agent $udp1

$cbr2 attach-agent $udp2

$cbr3 attach-agent $udp3

set null0 [new Agent/Null]

set null1 [new Agent/Null]

set null2 [new Agent/Null]

set null3 [new Agent/Null]

$ns attach-agent $n0 $null1

$ns attach-agent $n1 $null2

$ns attach-agent $n2 $null3

$ns attach-agent $n3 $null0

$ns connect $udp0 $null0

$ns connect $udp1 $null1

$ns connect $udp2 $null2

$ns connect $udp3 $null3

$ns at 0.5 "$cbr0 start"

$ns at 0.5 "$cbr1 start"

$ns at 0.5 "$cbr2 start"

$ns at 0.5 "$cbr3 start"

$ns at 2.5 "$cbr0 stop"

$ns at 2.5 "$cbr1 stop"

$ns at 2.5 "$cbr2 stop"

$ns at 2.5 "$cbr3 stop"

$ns at 5.0 "finish"

$ns run
```

Output 4.8:

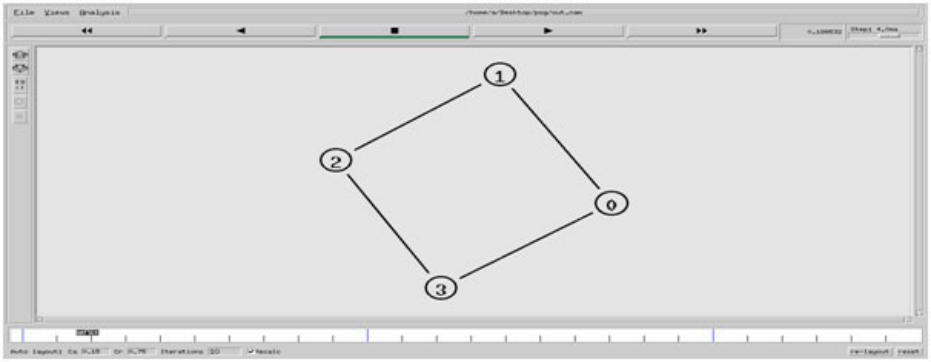


Figure 4.8 (a): Data transfer between four nodes (I)

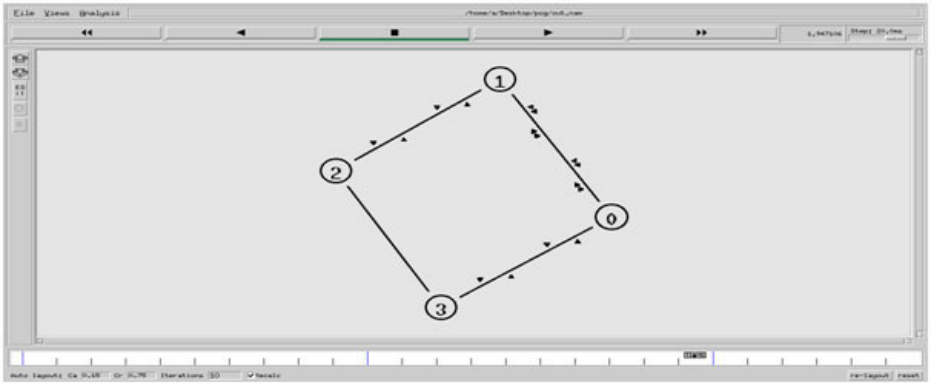


Figure 4.8 (b): Data transfer between four nodes (II)

Program 4.9: Write a Program to create a triangle and data transfer between them in NS2.

```
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 namout.nam&

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

$ns duplex-link $n0 $n1 100Mb 100ms DropTail

$ns duplex-link $n1 $n2 100Mb 100ms DropTail

$ns duplex-link $n2 $n0 100Mb 100ms DropTail

$ns at 5.0 "finish"

$ns run
```

Output 4.9:

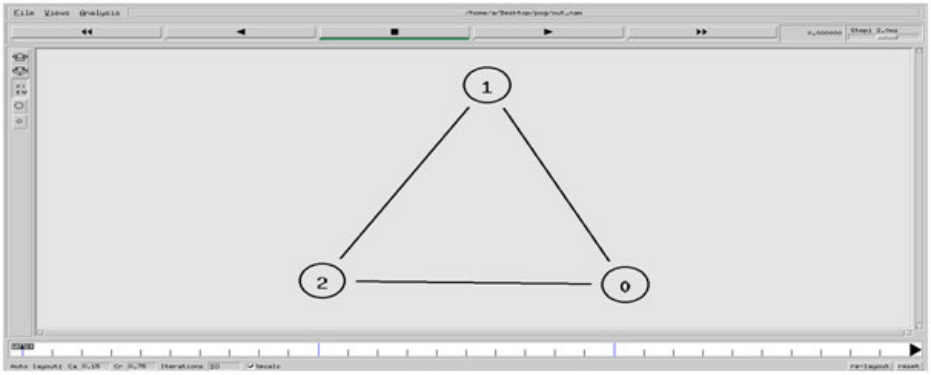


Figure 4.9 (a): Data transfer between three nodes (I)

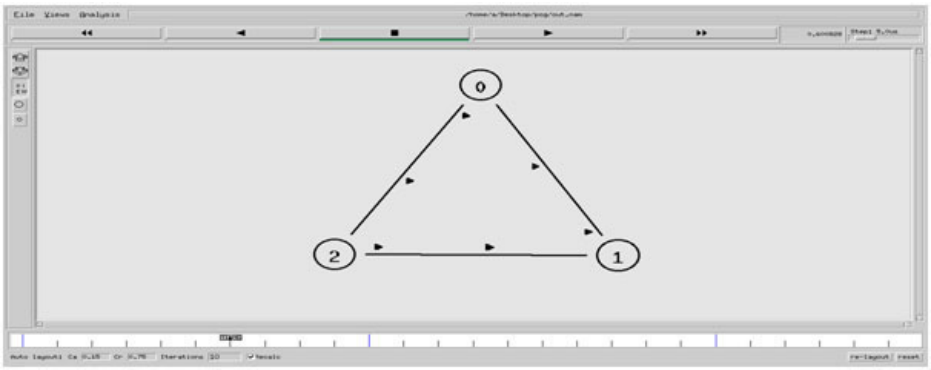


Figure 4.9 (b): Data transfer between three nodes (II)

Program 4.10: Write a Program to create a pentagon or star topology in NS2.

```
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 namout.nam&

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

$ns duplex-link $n0 $n1 100Mb 100ms DropTail

$ns duplex-link $n1 $n2 100Mb 100ms DropTail
```

$ns duplex-link $n2 $n3 100Mb 100ms DropTail

$ns duplex-link $n3 $n4 100Mb 100ms DropTail

$ns duplex-link $n4 $n0 100Mb 100ms DropTail

$ns at 5.0 "finish"

$ns run

Output 4.10:

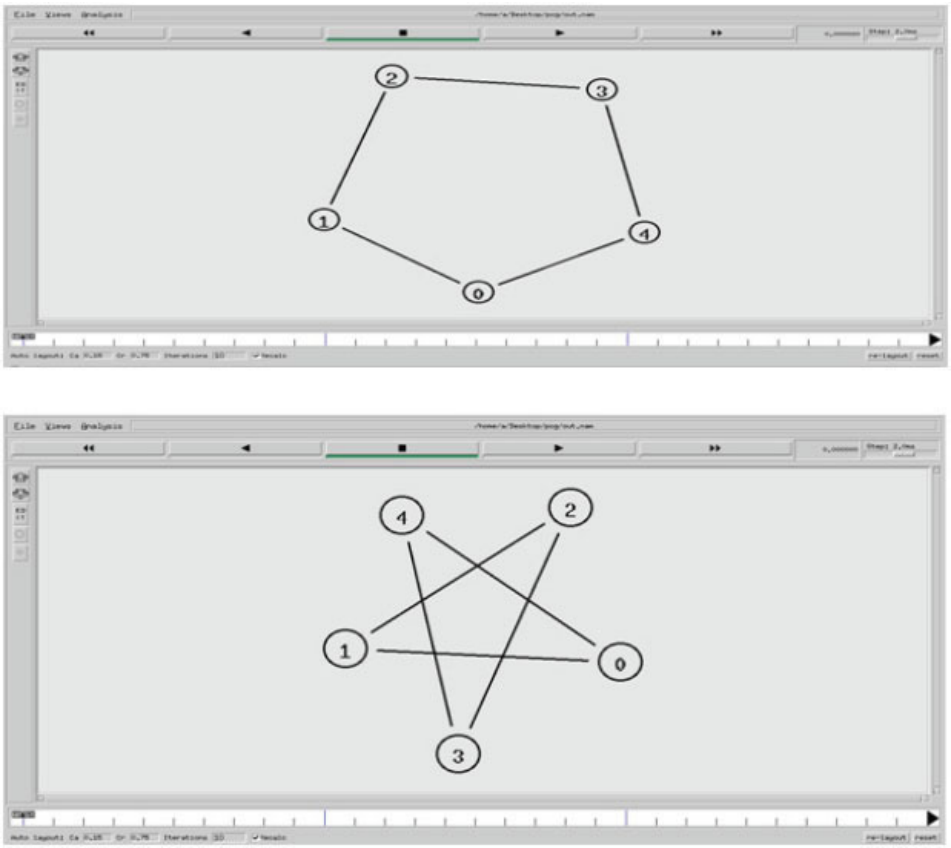


Figure 4.10: Star topology

Program 4.11: Write a Program to create a hexagon in NS2.

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 namout.nam&

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$ns duplex-link $n0 $n1 100Mb 100ms DropTail

$ns duplex-link $n1 $n2 100Mb 100ms DropTail

$ns duplex-link $n2 $n3 100Mb 100ms DropTail

$ns duplex-link $n3 $n4 100Mb 100ms DropTail

$ns duplex-link $n4 $n5 100Mb 100ms DropTail

$ns duplex-link $n5 $n0 100Mb 100ms DropTail

$ns at 5.0 "finish"

$ns run

Output 4.11:

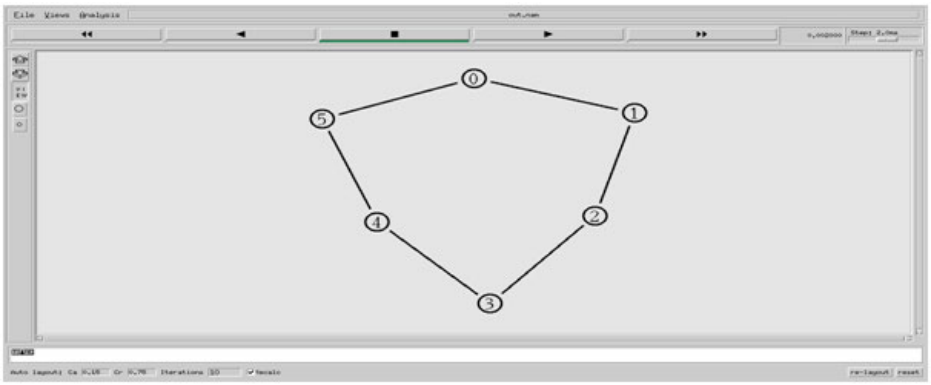


Figure 4.11: Hexagon

Program 4.12: Create a star topology and transfer data between wireless nodes.

Create a simulator object

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

Open the nam trace file

set nf [open out.nam w]

$ns namtrace-all $nf

Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

Close the trace file

close $nf

Executenam on the trace file

exec namout.nam&

exit0

}

Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n0 shape box

$n0 color green

$n4 color red

$n2 color red

$n1 color blue

$n3 color blue

Create links between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n2 $n0 1Mb 10ms DropTail

$ns duplex-link $n3 $n0 1Mb 10ms DropTail

$ns duplex-link $n4 $n0 1Mb 10ms DropTail

$ns duplex-link $n5 $n0 1Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient left-up

$ns duplex-link-op $n2 $n0 orient left-down

$ns duplex-link-op $n0 $n3 orient up

$ns duplex-link-op $n0 $n4 orient left-down

$ns duplex-link-op $n0 $n5 orient right-down

Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$ns attach-agent $n1 $tcp0

Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

Connect the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

Create a TCP agent and attach it to node n0

set tcp1 [new Agent/TCP]

$ns attach-agent $n4 $tcp1

Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink1 [new Agent/TCPSink]

$ns attach-agent $n2 $sink1

Connect the traffic sources with the traffic sink

$ns connect $tcp1 $sink1

$tcp0 set fid_ 1

$tcp1 set fid_ 2

Create a FTP and attach it to tcp0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp1

Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 attach-agent $tcp0

Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 0.5 "$ftp0 start"

$ns at 3.5 "$ftp0 stop"

$ns at 4.5 "$cbr0 stop"

Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

Run the simulation

$ns run

Output 4.12:

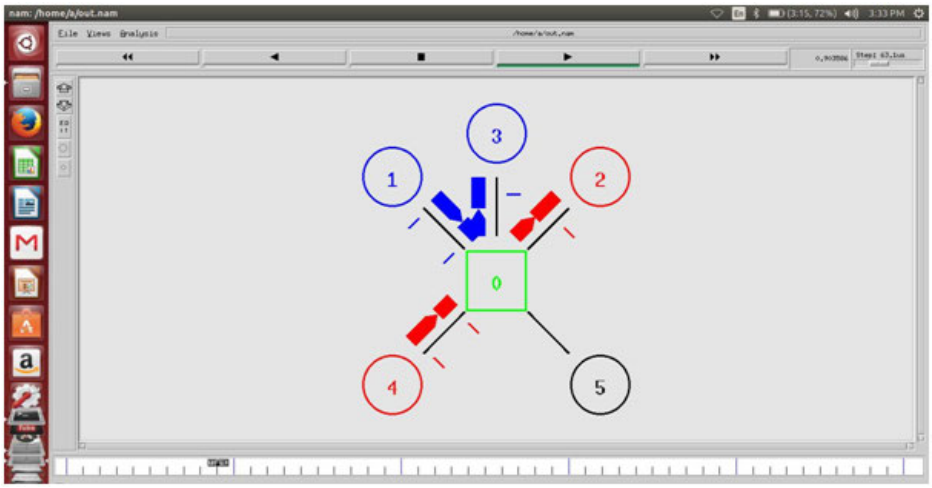


Figure 4.12: Transfer data between wireless nodes

Program 4.13: Create a Mesh topology and transfer data between wireless nodes.

set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

Close the trace file

close $nf

Executenam on the trace file

exec namout.nam&

exit0

}

```
set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

$ns duplex-link $n0 $n1 1Mb 2ms DropTail

$ns duplex-link $n0 $n2 1Mb 2ms DropTail

$ns duplex-link $n0 $n3 1Mb 2ms DropTail

$ns duplex-link $n0 $n4 1Mb 2ms DropTail

$ns duplex-link $n1 $n2 1Mb 2ms DropTail

$ns duplex-link $n1 $n3 1Mb 2ms DropTail

$ns duplex-link $n1 $n4 1Mb 2ms DropTail

$ns duplex-link $n2 $n3 1Mb 2ms DropTail

$ns duplex-link $n2 $n4 1Mb 2ms DropTail

$ns duplex-link $n3 $n4 1Mb 2ms DropTail

$ns duplex-link-op $n0 $n4 orient right

$ns duplex-link-op $n1 $n0 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns duplex-link-op $n2 $n3 orient right-down

$ns duplex-link-op $n3 $n4 orient left-down

$ns duplex-link-op $n3 $n2 orient right-down

set tcp0 [new Agent/TCP]

set ftp0 [new Application/FTP]

set tcp1 [new Agent/TCPSink]

$ns attach-agent $n1 $tcp0

$ns attach-agent $n4 $tcp1

$ftp0 attach-agent $tcp0

$ns connect $tcp0 $tcp1
```

Schedule events for the CBR agents

```
$ns at 0.5 "$ftp0 start"

$ns at 4.5 "$ftp0 stop"
```

Call the finish procedure after 5 seconds of simulation time

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

Run the simulation

$ns run

Output 4.13:

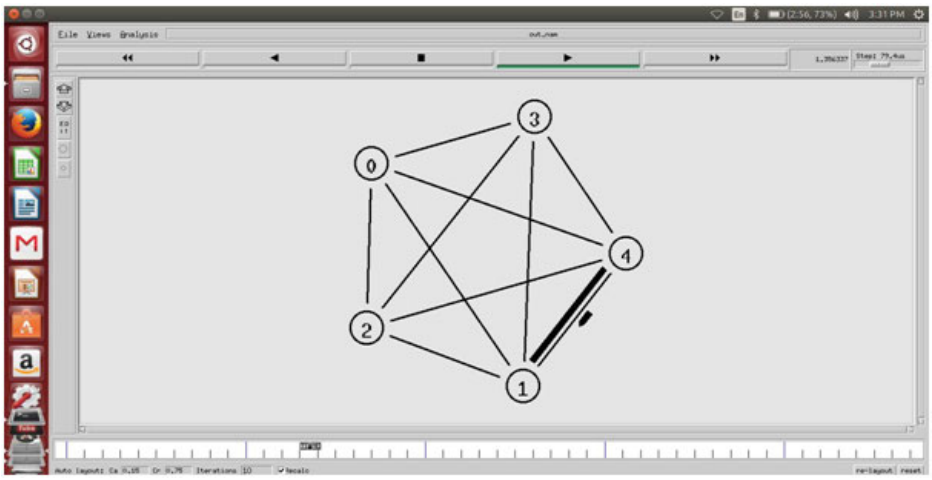


Figure 4.13: Mesh topology and transfer data between wireless nodes

CHAPTER 5

Mobile Networking in NS2

In NS2, the node can be configured to support the wireless transmission. The node-config function of the Simulator class is used to configure the node. For node configuration, parameters need to assign some value.

Steps for mobile networking simulation

Create Simulator object

Create General Operation Director (GOD)

Create a topography object

Configure attributes for a wireless node

Attach agent and application

Connect them

General Operation Director

Two nodes can transmit data to each other only if they lie in the range of each other. If a node moves away from the range of another node, then the connection will be terminated. For this purpose, NS2 has General Operation Director (GOD) .

General Operation Director (GOD) keeps information about each node in wireless simulation. For every wireless simulation, GOD needs to be created. At the time of GOD creation, several nodes need to be specified, so that GOD can reserve memory space for those nodes for keeping their information.

Syntax:

create-god

Topology

To determine area in which mobile host can move, topography class is used.

Syntax:

Create object of Topography class.

set topo [new Topography]

Specify boundary of area in which node can move.

$topo load_Flatgrid&tl;Y> res

Size of grid is specified in x and y direction and res passed as grid resolution.

Mobile Node Movement

For moving a node during a simulation following commands are used.

Random Motion: Node can move randomly if their random-motion property is on.

Syntax:

$[node-instance] random-motion 0 or 1

Random motion will be turned on if 1 is specified, in case of 0 it will be off.

Setting Position: Position(X, Y, Z) of node can be set in NS2. Z coordinates of position is 0. Following command is used for setting position.

Syntax:

$[node-instance] set X_ <x-coordinates>

$[node-instance] set Y_ <y-coordinates>

$[node-instance] set Z_ <z-coordinates>

Setting Movement: Future position can be set by using the command.

Syntax:

$[simulator-instance] at $[node-instance] setdest

Example

$ns at 1.2 $n0 setdest 300 200 20

In above command node, start moving to position 300, 200 at time 1.2 with 20 speed.

When above command executed, node starts moving to specified position with sp speed.

Setting Radius of Node: Range of wireless node can be changed by using this command.

Syntax:

[node-instance] radius

$n0 radius 20

above command will set radius for node 0 as 20.

The radius denotes the node's range. All mobile nodes that fall within the circle of radius with the node at its center are considered as neighbors.

Setting distance between nodes: Distance between nodes can be changed for routing protocols by using this command.

Syntax:

[god-instance] set-dist

$god set-dist 0 1 2

Above command will set the distance between node 0 and 1 as 2 hops.

First Wireless Simulation script

Create Simulator class object:

set ns [new Simulator]

Assign properties for wireless connection. Here, you can access properties for wireless connection.

Code:

$ns node-config -adhocRouting DSR \

-llType LL \

-macType Mac/802_11 \

-ifqType Queue/DropTail \

-ifqLen 5 \

-antType Antenna/OmniAntenna \

-propType Propagation/TwoRayGround \

-phyTypePhy/WirelessPhy \

-topoInstance $topo \

-channel [new Channel/WirelessChannel] \

-agentTrace ON \

-movementTrace OFF \

-agentTrace ON \

-macTrace ON

Create GOD and topology instance:

create GOD for two nodes

create-god 2

create topology instance and set boundary

set topo [new Topography]

$topo load_flatgrid 600 600

Create nodes:

set n0 [$ns node]

set n1 [$ns node]

Set coordinates for both nodes:

$n0 set X_ 20.0

$n0 set Y_ 20.0

$n0 set Z_ 0.0

$n1 set X_ 500.0

$n1 set Y_ 500.0

$n1 set Z_ 0.0

Energy Model in NS2

In wireless network, energy model maintains total energy at each node. During simulation, energy model in NS2 is used to access energy of node. In NS2, energy model for any node can be assigned as following:

[Simulator-instance] node-config

-energyModel "EnergyModel" \

-idlePower 1.0 \

-rxPower 1.0 \

-txPower 1.0 \

-sleepPower 0.001 \

-transitionPower 0.2 \

-transitionTime 0.005 \

-initialEnergy 1000 \

Energy Model's attributes

Energy analysis

After simulation, energy is stored in the following format in trace file.

[energy 998.999217 ei 1.000 es 0.000 et 0.000 er 0.001]

In above format, first the name of the attribute and thenthe value of the attribute is given.

Meaning of each attribute:

energy: total remaining energy

ei: energy consumption in IDLE state

es: energy consumption in SLEEP state

et: energy consumed in transmitting packets

er: energy consumed in receiving packets

Wireless network programs

Program 5.1: Simulation program for LAN network

set ns [new Simulator]

Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

Open the Trace files

set file [open out.trw]

set winfde [open WmFile w]

$ns trace-all $file1

Open the NAM trace file

set file2 [open out.nam w]

$ns namtrace-all$file2

Define a 'finish' procedure

proc finish {} {

globalns file1 file2

$ns flush-trace

close $file1

close $file2

exec namout.nam&

exit 0

}

Create six nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n1 color red

$n1 Shapebox

Create links between the nodes

$ns duplex-link $n1 $n2 2Mb 10ms Drop Tail

$ns duplex-link Sn1 $n2 2Mb 10ms Drop Tail

$ns simplex-link $n2 $n3 0.3Mb 100ms Drop Tail

$ns simplex-link $n3 $n2 0.3Mb 100ms Drop Tail

set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/Drop Tail MAC/Csma/Cd Channel]

$ns duplex-link $n3 $n4 0.5Mb 40ms Drop Tail

$ns duplex-link $n3 $n5 0.5Mb 30ms Drop Tail

Give node position (for NAM)

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns simplex-link-op $n2 $n3 orient right

$ns simplex-link-op $n3 $n2 orient left

$ns duplex-link-op $n3 $n4 orient right-up

$ns duplex-link-op $n3 $n5 orient right-down

Set Queue Size of link (n2-n3) to 10

$ns queue-limit $n2 $n3 10

Setup a TCP connection

set tcp [newAgent/TCP/Newreno]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink/DelAck]

$ns attach-agent $n4 $sink

$ns connect $tcp $sink

$tcp set fid_ 1

$tcp set window_8000

$tcp set packetSize_552

Setup a FTP over TCP connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type_ FTP

Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $nl $udp

set null [new Agent/Null]

$ns attach-agent $n5 $null

$ns connect $udp $null

$udp set fid_ 2

Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type_CBR

$cbr setpacket*size*1000

$cbr set rate_ 0.01mb

$cbr set random_ false

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 124.0 "$ftp stop"

$ns at 124.5 "$cbr stop"

next procedure gets two arguments: the name of the

tcp source node, will be called here "tcp",

and the name of output file.

proc plotWindow {tcp Source file) {

gobalns

set time 0.1

set now [$ns now]

set cwnd [Stcp Source set cvmd_]

set wnd [$tcpSource set window_]

puts $ file "nowcwnd"

$ns at [expr $now+Stime] "plotWindow $tcpSource $ file" }

$ns at 0.1 "plotWindow$tcp $winfile"

$ns at 5 "$ns trace-annotate\"packet drop\""

PPP

$ns at 125.0 "finish"

$ns run

Output 5.1:

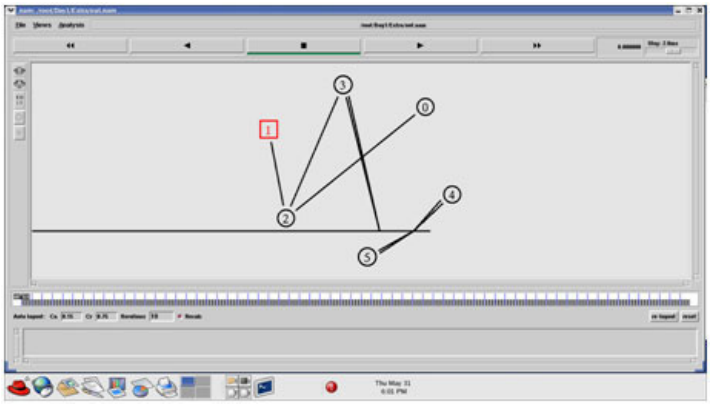


Figure 5.1: Network formation

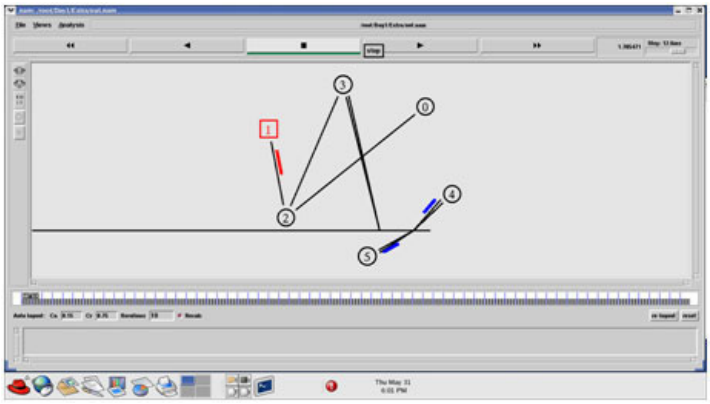


Figure 5.2: Data transmission

Program 5.2: Unicast program

set ns [new Simulator]

Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

Open the Trace file

set filel [open unicastDV.trw]

$ns trace-all S filel

0pen the NAM trace file

set fi1e2 [open unicastDV .namw]

$ns namtrace-all $flLe2

Define a 'finish' procedure

proc finish {} {

global ns file1 file 2

$ns flush-trace

close $file1

close $file2

exec namunicastDV.nam&

exit 0

}

Next line should be commented out to have the static routing

$ns rtproto DV

Create six nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [4ns node]

set n5 [$ns node]

Create links between the nodes

$ns duplex-link $n0 $n1 0.3Mb 10ms Drop Tail

$ns duplex-link $n1 $n2 03Mb 10ms Drop Tail

$ns duplex-link $n2 $n3 0.3Mb 10ms Drop Tail

$ns duplex-link $n1 $n4 0.3Mb 10ms Drop Tail

$ns duplex-link $n3 $n5 0.5Mb 10ms Drop Tail

$ns duplex-link $n4 $n5 0.5Mb 10ms Drop Tail

Give node position (for NAM)

$ns duplex-link-op $n0 $n1 orient right

$ns duplex-link-op $n1 $n2 orient right

$ns duplex-link-op $n2 $n3 orient up

$ns duplex-link-op $n1 $n4 orient up-left

$ns duplex-link-op $n3 $n5 orient left-up

$ns duplex-link-op $n4 $n5 orient right-up

Setup a TCP connection

set tcp [newAgent/TCP/Newreno]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink/DelAek]

$ns attach-agent $n5 $sink

$ns connect $tcp $sink

$tcp set fid_ 1

Setup a FTP over TCP connection

set ftp [newApplication/FTP]

$ftp attach-agent $tcp

$ftp set type_ FTP

$ns rtmodel-at 1.0 down $n1 $n4

$ns rtmodel-at 4.5 up $n1 $n4

$ns at 0.1 "$ftp start"

$ns at 6.0 "finish"

$nsnm

Output 5.2:

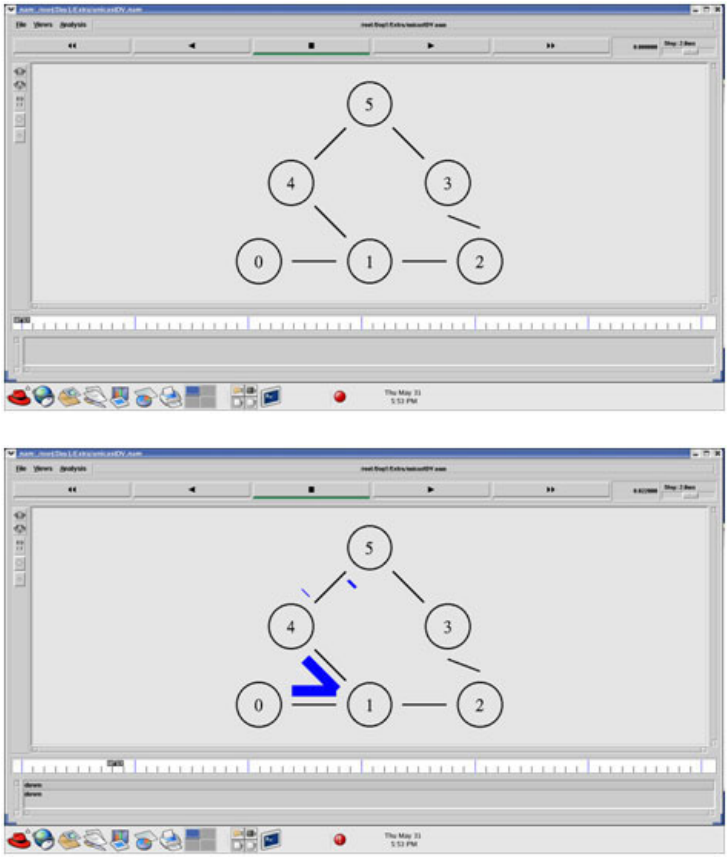


Figure 5.3: Data transfer

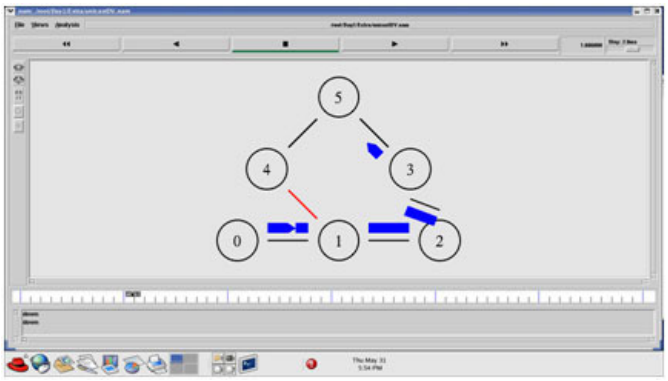


Figure 5.4: Path change due to link failure

Program 5.3: Multicast program 1

set ns [new Simulator]

$ns multicast

set f [open out.tr w]

$ns trace-all $f

$ns namtrace-all [open out.nam w]

$ns color 1 red

the nam colors for the prune packets

$ns color 30 purple

the nam colors for the graft packets

$ns color 31 green

allocate a multicast address;

set group [Node allocaddr]

nod is the number of nodes

set nod 6

create multicast capable nodes;

for {set i l} ($i<= $nod} (incri) {

set n($i) [$ns node]

}

Create Links between the nodes

$ns duplex-link $n(1) $n(2) 0.3Mb 10 msDropTail

$ns duplex-link $n(2) $n(3) 0.3Mb 10 msDropTail

$ns duplex-link $n(2) $n(4) 0.5Mb 10 MsDropTail

$ns duplex-link $n(2) $n(5) 0.3Mb 10 MsDropTail

$ns duplex-link $n(3) $n(4) 0.3Mb l0 MsDropTail

$ns duplex-link $n(4) $n(S) 0.5Mb l0 MsDropTail

$ns duplex-link $n(4) $n(6) 0.5Mb l0 MsDropTail

$ns duplex-link $n(S) $n(6) 0.5Mb l0 MsDropTail

configure multicast protocol;

set mprotoDM

all nodes will contain multicast protocol agents;

set mrthandle[$ns mrtproto$mproto]

set udpl[new Agent/ UDP]

set udp2 [new Agent/UDP]

$ns attach-agent $n(l) $udpl

$ns attach-agent $n(2) $udp2

set src1 [new Application Traffic/ CBR]

$src1 attach-agent $udp1

$udp1 set dst_addr$group

Sudp 1 set dst_port0

$src1 set random_ false

set src2 [new Application Traffic/ CBR]

$sr c2 attach-agent $udp2

$udp2 set dst*addr* $group

Sudp2 set dst_port1

$sr c2 set random_ false

create receiver agents

set rcvr [new Agent /LossMonitorj

joining and leaving the group;

$ns at 0.6 “$n(3) join-group $rcvz $gro”

$ns at 1.3 “$n(4) join-group Srcvr$group”

$ns at 1.6 “$n(5) join-group $rcvr$group”

$ns at 1.9 “$n(4) leave-group $rcvr$group”

$ns at 2.3 “$n(6) join-group $rcvr$group”

$ns at 3.5 “$n(3) leave-group $r$vr $group”

$ns at 0.4 “$src1 start”

$ns at 2.0 “$src2 start”

$ns at 4.0 “finish”

proc finsh{} {

global ns

$ns flush-trace

exec nain out. nam &

exit 0

}

$ns run

Output 5.3:

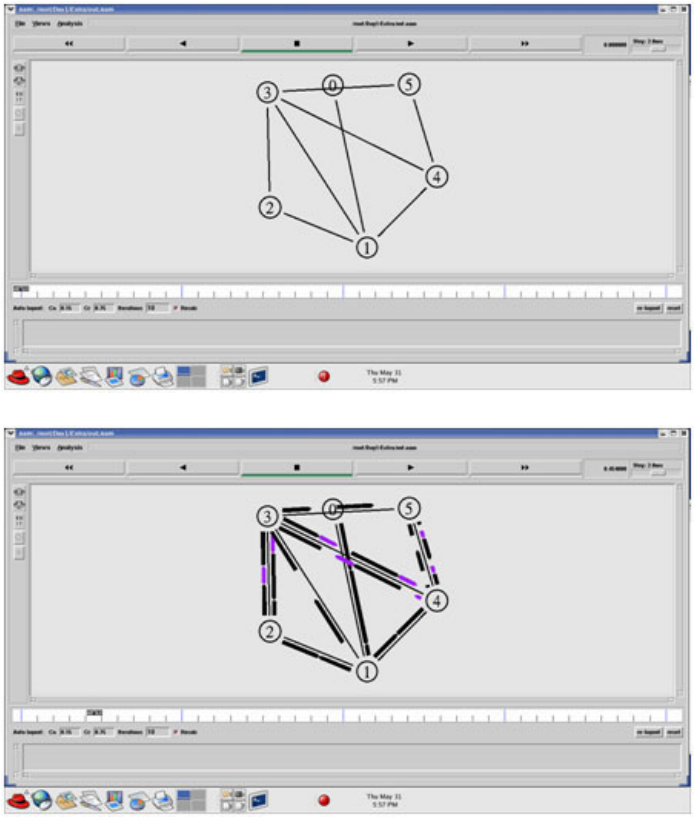


Figure 5.5: Data transmission

Program 5.4: Multicast program 2

set ns [new Simulator]

$ns multicast

set f [open out.tr w]

$ns trace-all $f

$ns namtrace-all [open out.nam w]

$ns color 1 red

the nam colors for the prune packets

$ns color 30 purple

the nam colors for the graft packets

$ns color 31 green

allocate a multi cast address;

set group [Node allocaddr]

nod is the number of nodes

set nod 6

create multicast capable nodes;

for (set i l) {Si <= $nod} {incri} {

set n($i) [$ns node]

Create links between the nodes

$ns duplex-link $n(1) $n(2) 03Mb l0msDropTail

$ns duplex-link $n(2) $n(3) 03Mb l0ms DropTail

$ns duplex-link $n(2) $n(4) 0.5Mb l0ms DropTail

$ns duplex-link $n(2) $n(S) 0.3Mb l0msDropTail

$ns duplex-link $n(3) $n(4) 0.3Mb l0ms DropTail

$ns duplex-link $n(4) $n(5) 0.5Mb l0msDropTail

$ns duplex-link $n(4) $n(6) 05Mb l0ms DropTail

$ns duplex-link $n(S) $n(6) 0.5Mb l0ms DropTail

configure multicast protocol;

DM set CacheMissModedvmrp

set mprtoto DM

all nodes will contain multicast protocol agents;

set mrthandle [$ns mrtproto $mproto]

set udp 1 [new Agent / UDP]

set udp2 [new Agent /UDPJ]

$ns attach-agent $n(1) $udp1

$ns attach-agent $n(2) Sudp2

set src1 [new Application / Traffic/CBR]

$src1 attach-agent $udp1

$udplset dst*addr*$ group

$udpl set dst*port*0

Ssrc 1 set random_ false

set src2 [new Application /Traffic/CBR]

$src2 attach-agent $udp2

$udp2 set dst*addr* $group

$udp2 set dst*port* 1

$src2 set random_ false

create receiver agents

set rcvr [new Agent/LossMonitor]

joining and leaving the group;

$ns at 0.6 $n(3) join-group Srcvr $group”

$ns at 1.3 $n(4) join-group $rcvr $group”

$ns at 1.6 ‘$n(5)join-group$rcvr$group”

$ns at 1.9 ‘in(4) leave-group SrcvrSgroup

$ns at 2.3 $n(6) join-group $rcvr$ Sgroup ”

$ns at 3.5 in(3) leave-group $rcvr Sgroup ”

$ns at 0.4 “$src1 start”

$ns at 2.0 “$src2 start”

$ns at 4.0 “finish”

proc finish{}{

global ns

$ns flush-trace

exec narnout.nam&

exit 0

$ns run

Output 5.4:

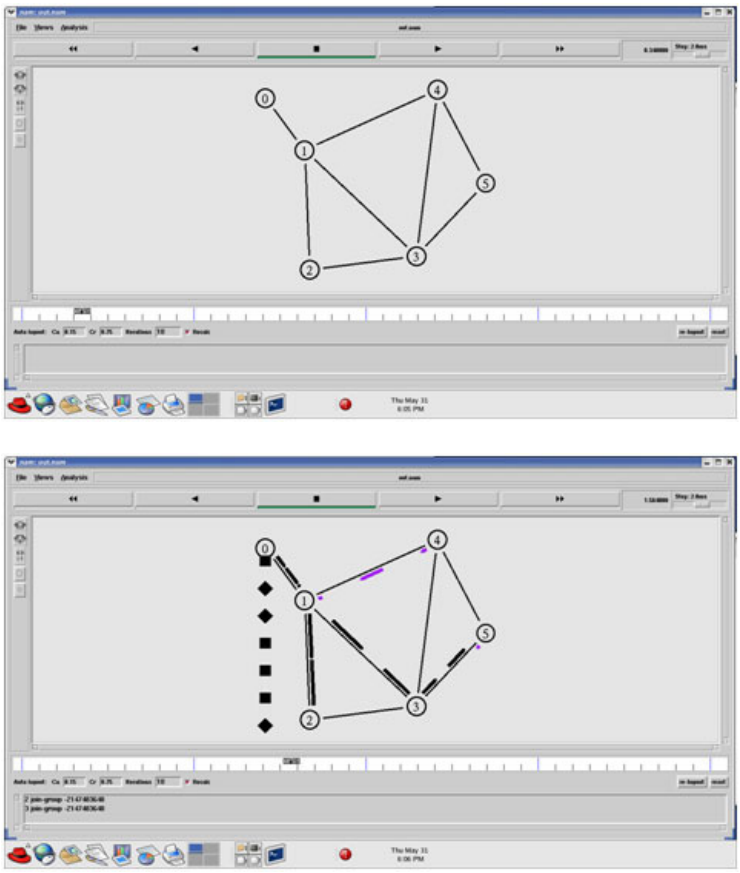


Figure 5.6: Data transmission

Mobile/Wireless Node Structure

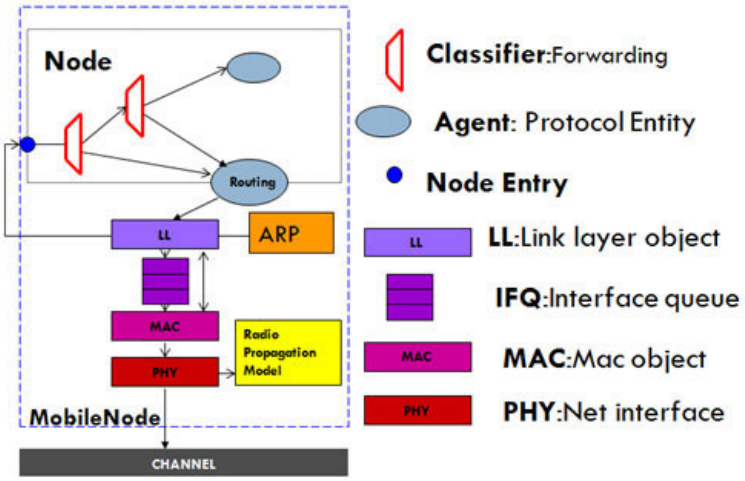


Figure 5.7: Node structure

Examples for Wireless Program in NS2

Program 5.5: Wireless program 1

```
#Define options
set val(chan) Channel/Wireless/Channel    ;# channel type
set val(prop) Propagation /TwoRayGround   ;#; radio-propagation
set val(netif) Phy/ WirelessPhy           ;# network interface
set val(mac)  Mac/ 802_11                 ;# MAC type
set val(ifq)  Queue/ DropTail/PriQueue    ;#interface queue
setval(ll)    LL                          ;#link layer type
set val(ant)  Antenna/ OmniAntenna        ;#antenna model
set val(ifqlen) 50                        ;#max packet inifq
set val(nn)   20                           ;#number of nodes
set val(rp)   AODV                        ;#routing  protocol
set val(x)    500                         ;# X dimension
set val(y)    400                         ;# Y dimension
set val(stop) 110                         ;# simulation end
set ns        [new Simulator]
set tracefd   [open wireless.tr w]
set namtrace [open wireless.namw]
```

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

setup topography object

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

Create nnmobilenodes [$val(nn)] and attach them to the channel.

configure the nodes

$ns node-config -adhocRouting $val(rp)

-llTvpe $val(1l) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-ant Type $val(ant)\

-prop Tpe $val(prop) .

-phyType $val(netif) \

-channelType $val(chan) \

-topoInstance $topo \

-agent Trace OFF \

-routerTrace OFF \

-macTrace ON \

-movement Trace ON \

for{set i 0} {$i<$val(nn)} (incri } {

set node_($i) [$ns node)]

}

Provide initial location of nodes

$node*(0) set X* 5.0

$node*(0) set Y* 5.0

$node*(0) set Z* 0.0

$node*(1) set X* 490.0

$node*(1) set Y* 285.0

$node*(1)set Z*0.0

$node(2)setX_ 150.0

$node*(2) set Y* 240.0

$node*(2) set Z* 0.0

Generation of movements

$ns at 10.0 “$node_(0) setdest 250.0 250.03.0”

$ns at 15.0 “$node_(1)setdest 45.0 285.0 5.0”

$ns at 110.0 “$node_(0)setdest480.0 300.0 5.0”

Set a TCP connection between node*(0) and node*(1)

set udp [new Agent/ UDP]

set sink [new Agent/ Null]

$ns attach-agent Snode_(0) $udp

$ns attach-agent Snode_(1) $sink

Sns connect $udp $sink

set cbr [new Application/ Traffic/ CBR]

$cbr attach-agent Sudp

$cbr set interval_ 1

$cbr set maxpkts_ 100

$ns at 10.0 “$cbr start”

Define node initial position in nam

for (set i 0) (Si <Sval(nn)) (incri) {

30 defines the node size for nam

$ns initial*node*pos $node_(Si) 30

}

Telling nodes then the simulation ends

for{set i 0) {$i<$val(nn)} {incri} {

$ns at $val(stop) “$node_(Si) reset”;

}

ending nam and the simulation

$ns at $val(stop) “Snsnam-end-wireless $val(stop)”

$ns at $val (stop) ”stop”

$ns at 110.01 “puts \“end simulation\ “;$ns halt”

Proc stop (} {

global ns tracefdnarrtiace

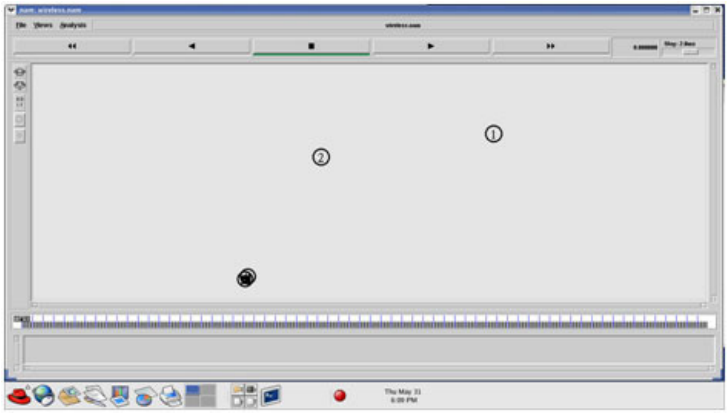
Sns flush-trace

close $tracefd

close $namtrace

Sns run

Output 5.5:



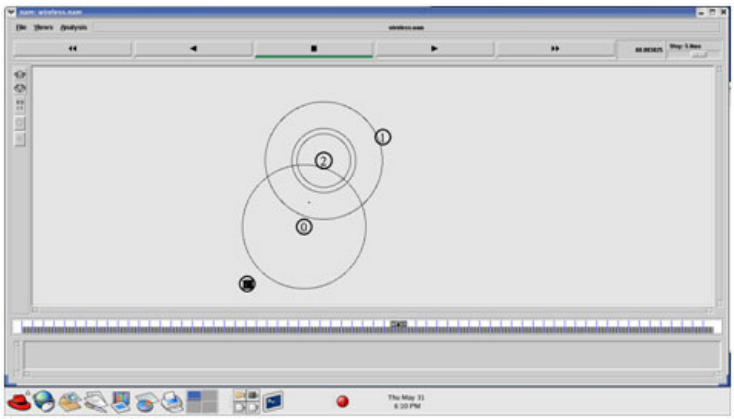


Figure 5.8: Data transmission

Program 5.6: WSN Program – 802.11

```
#Define options
set val(chan)  Channel/ Wireless/Channel ;# channel type
set val(prop)  Propagation/TwoRayGround  ;#; radio-propagation
set val(netif) Phy/ WirelessPhy          ;# network interface
set val(mac)   Mac/ 802_11               ;# MAC type
set val(ifq)   Queue/ DropTail/PriQueue  ;#interface queue
setval(ll)     LL                        ;#link layer type
set val(ant)   Antenna/ OmniAntenna      ;#antenna model
setval(ifqlen) 100                       ;#max packet inifq
set val(nn)    20                        ;#number of nodes
set val(rp)    AODV                      ;#routing  protocol
set val(x)     50                        ;# X dimension
set val(y)     50                        ;# Y dimension
set val(stop)  500                       ;# simulation end
set val(energymodel) EnergyModel         ;#EnergyModel
set val(initialenergy) 100               ;# value
set ns       [new Simulator]
set tracefd     [open sim_802_1 1.trw
set namtrace [open sim_802_11.namw]
$ns use-new trace
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace  $val(x) $val(y)
```

set up topography object

set topo [new Topography]

$topoload_flatgrid $val(x) $val(y)

create-god $val(nn)

configure the nodes

$ns node-config -adhocRoutingSval(rp)

-llTvpeSval(1l) \

-macType$val(mac)\

-ifqType $val(ifq) \

-ifqLen $val(ifqlen)\

-antType $val(ant) \

-propType $val(prop)\

-phy Type $val(netif)\

-channel [new $val(chan)]\

-topo Instance $topo \

-agentTrace OFF \

-routerTrace OFF \

-macTrace ON \

-movement Trace OFF \

-energyModel $val(energymodel) * \

-initialEnergySval(initial energy) \

-rxPower 35.28e-3 \

-txPower 31 32e-3 \

-idlePower 712e-6 \

-sleep Power 1 44e-9

for {set i 0) {$i<Sval(nn)} {incri} {

set mnode_(Si) [$ns node]

}

for{seti 1} {$i<Sval(nn)} {incri} {

$mnode*($i) set X* [expr {$val(x)* rand)]

$ninode*($i) set Y* [expr {$val(y)* rand()]

$mnode*($i) set Z* 0

}

Position of Sink

$mnode*(0) setX* [expr {$val(x)2}]

$mnode*(0) setY* [expr ($val(y)2)]

$mnode*(0) set Z* 0.0

$mnode_(0) label “Sink”

for {set i0} {$i<$val(nn)} {incri } {

Snsinitial*node*pos $mnode_($i) 10

}

Setup a UDP connection

for {seti 1} {$i<Sval(nn)} {incri } {

set udp($i) [new Agent/UDP]

$ns attach-agent Smnode ($i)$udp($i)

}

set sink [new Agent/ Null]

$ns attach-agent $mnode_(0) $sink

for{seti 1} {$i<$val(nn)} {incri) {

$ns connect Sudp(Si) $ sink

}

Setup a CBR over UDP connection

for {set i 1} {$i<$val(nn)} {incri } {

set cbr($i) [new Application/ Traffic/ CBR]

$cbr($i) attach-agent Sudp($i)

$cbr($i) set type_ CBR

$cbr($i) set packet*size* 100

$cbr($i) set maxpkts_ 100

$cbr($i) set rate_0.1Mb

$cbr($i) set interval_ 1

$cbr($i) set random_ false

for {seti 1} {$i<Svall(nn)} {incri } {

Sns at [expr ($i+ 5) “$cbr($i) start”

}

for {set i 1} {$i<$val(nn)} {incri } {

Sns at [expr $val(stop)- $i] “Scbr($i) stop”

}

Telling nodes then the simulation ends

for{set i 0} {$i<$val(nn)} {incri} {

$ns at $val (stop) ‘$mnode_ ($i) reset;”

}

ending nam and the simulation

$ns at $val(stop) “$in nam-end-wireless $val(stop)”

$ns at $val(stop) “stop”

Sns at [expr $val(stop)+ 001] “puts \”end simulation\ “ ;Snshalt”

proc stop {} {

global ns tracefdnamtrace

$in flush-trace

close $tracefd

close $namtrace

}

Sns run

Output 5.6:

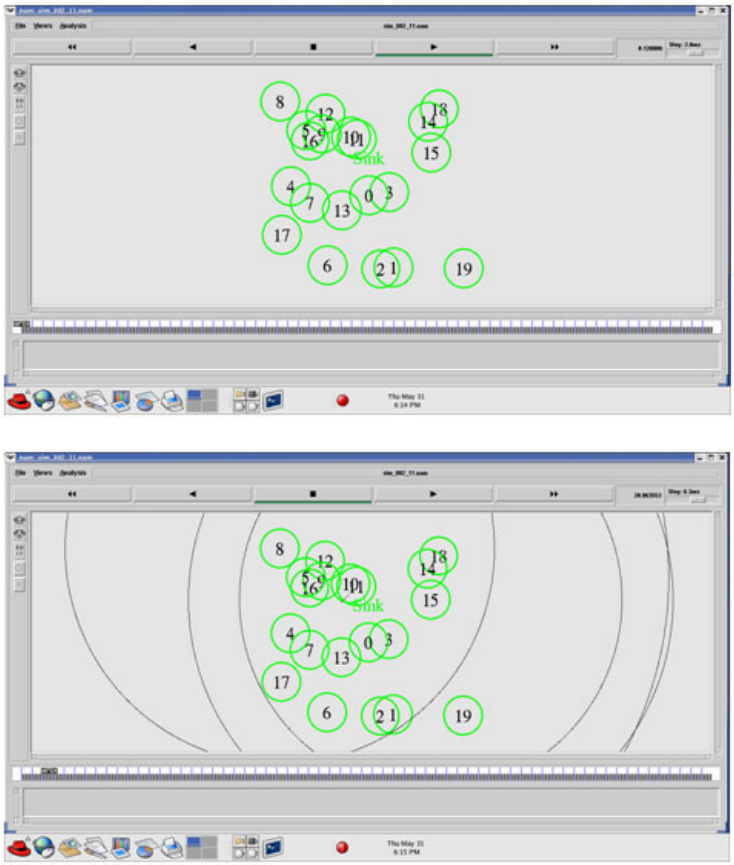


Figure 5.9: Data transmission

Program 5.7: WSN Program – 802.15.4

Define options

```
set val(chan)  Channel/Wireless/Channel ;# channel type
set val(prop)  Propagation/TwoRayGround ;#; radio-propagation
set val(netif) Phy/ WirelessPhy         ;# network interface
set val(mac)   Mac/ 802_11              ;# MAC type
set val(ifq)   Queue/ DropTail/PriQueue ;#interface queue
setval(ll)     LL                       ;#link layer type
set val(ant)   Antenna/ OmniAntenna     ;#antenna model
setval(ifqlen) 100                      ;#max packet inifq
set val(nn)    100                      ;#number of nodes
set val(rp)    AODV                     ;#routing  protocol
set val(x)     50                       ;# X dimension
set val(y)     50                       ;# Y dimension
set val(stop)  500                      ;# simulation end
set val(energymodel) EnergyModel        ;#EnergyModel
set val(initialenergy) 100              ;# value
set ns [new Simulator]
set tracefd [open sim_802_15_4.trw
set namtrace [open sim_802_15_4 nam.w]
$ns use -newtrace
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

$set topo [new Topography]

$topoload_flatgrid$val(x) $val(y)

create-god $val(nn)

configure the nodes

$ns node-config –adhocRouting $val(rp)

-llType $va)(ll) \

-mac Type $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val (ifqlen)\

- antTvpe $val(ar) \

-propType $val(prop)\

- phy Type $val(netif)\

-channel [new $val(chan)]

-topoInstance $topo\

-agentTraceOFF \

-routerTraceOFF \

-macTrace ON \

-movementTraceOFF\

-energvModel $val(energymodel) \

-initialEnergy $val(initialnergy) \

-rxPower 35e-3

-txPower 31 e-3

-idle Power 31 e-3

-sleepPower 1 5e-9

for{set i 0} {$i<$val(nn)} {incri} {

set mnode_ ($i) [$ns node]

}

for{set i 1} {$i<$val(nn)}{incri} {

$mnode*($i) set X* [expr ($val(x) rand()}]

$ninode*($i) set Y* [expr ($val(y) *rand()}]

$mnode*($i) set Z* 0

}

Position of Sink

$mnode*(0) set X* [expr {$val(x) /2}]

$mnode*(0) set Y* (expr {$va1(y)/2}]

$mnode*(0) set Z* 0.0

$mnode_(0)label “Sink”

for{set I 0} {$i<$val(nn)}{incri} {

$ns initial*node*pos $mnode_($i) 10

}

SISetup a UDP connection

for {set i 1) {$i<$val(nn)} {incri} {

set udp($i) [new Agent/ UDP]

$ns attach-agent $mnode_($i) Sudp($i)

}

set sink [new Agent/ Null]

$ns attach-agent $mnode_(0) $sink

for{set i 1} {$i<$val(nn)} {incri} {

$ns connect $udp($i) $sink

}

Setup a CBR over UDP connection

```
for {seti 1} {$i<$val(nn)} {incri } {

set cbr($i) [new Application /Traffic/ C BR]

$cbr($i) attach-agent Su4($i)

$cbr($i) set type_ CBR

$cbr($i) set packetsize 100

$cbr($i) set maxpkts_ 100

$cbr($i) set rate 0.1 Mb

$cbr($i) set interval 1

$cbr($i) set random_ false

for {set i 1} {$i<$val(nn)) {incri} {

Snsat [expr {Si+ 5}] “$cbr($i) start”

}

for {set i 1} {$i<$val(nn)} {incri} {

Sns at [expr $val(stop) - $i] “$cbr($i) stop”

}

:#Telling nodes when the simulation ends

for {set i 0} {$i<$val(nn)} {incri} {

$ns at $val(stop) “$mnode_($i) reset;”
```

ending nam and the simulation

```
$ns at $val(stop) “$ns nam-end-wireless $val(stop)”

$ns at $val(stop) “stop”

$ns at [expr $val(stop) +0.011] puts\ “end simulation\”;$ns halt”

proc stop{} {

global ns tracefdnantrace

$ns flush-trace

close $tracefd

close $namtrace

)

Sns run
```

Output 5.7:

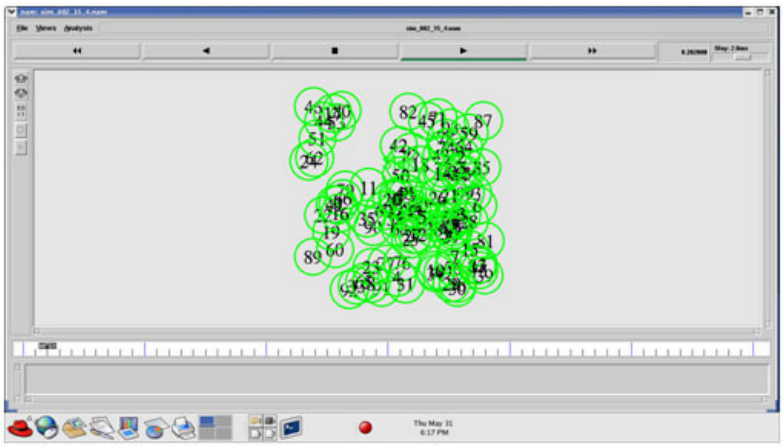


Figure 5.10: WSN Program – 802.15.4

○ Protocol Works

Procedure to construct Malicious Node in TCL Script and C++.

Modification in AODV PROTOCOL

Location: ns-allinone-2.33/ns2.33/aodv/aodv.cc

Location: ns-allinone-2.33/ns2.33/aodv/aodv.h

aodv.h file changes

Declare a Boolean variable malicious as shown below in the protected scope in the class AODV

bool malicious;

aodv.cc file changes

Initialize the malicious variable with a value " false ". Declare it inside the constructor as shown below:

AODV::AODV (nsaddr*t id):Agent(PT*AODV)…

{

…….

malicious = false;

}

Add the following statement to the aodv.cc file in the "if (argc==2)" statement.

```
if (strcmp(argv[1], "malicious") == 0) {
    desyn = true;
    return TCL_OK;
}
```

Implement the behavior of the malicious node by setting the following code in the rt_resolve(Packet *p) function. The malicious s node will simply drop the packet as shown below.

```
if(malicious ==true)
{
drop(p,DROP*RTR*ROUTE_LOOP);
}
```

Once done, recompile ns2 as given below:

Open Terminal -> Go to ~ns-2.33/ directory and type the following command to compile

$] cd /ns-allinone-2.33/ns-2.33/

$] make

Once the compilation is done, check the malicious behavior by using the Tcl Script by setting any four node as malicious node. The command to set the malicious node is

$ns at 2.0 "[$n0 set ragent_] malicious "

$ns at 2.0 "[$n8 set ragent_] malicious "

$ns at 2.0 "[$n23 set ragent_] malicious "

$ns at 2.0 "[$n19 set ragent_] malicious "

To generate random mobility in NS2

Procedure

Open the new terminal

cd ns-allinone-2.34

cd ns-2.34

cdindep-utils

pwd

ls

cdcmu-scen-gen

ls

cdsetdest

ls

./setdest

./setdest –v 2 –n 10 –s 1 –m 10 –M 50 –t 30 –P 1 –p 1 –x 500 –y 500

./setdest –v 2 –n 10 –s 1 –m 10 –M 50 –t 30 –P 1 –p 1 –x 500 –y 500 >usersetdest.tcl

geditusersetdest.tcl

To generate random agent and application creation in NS2

Procedure

cd ns-allinone-2.34

cd ns-2.34

cdindep-utils

cdcmu-scen-gen

ls

nscbrgen.tcl

nscbrgen.tcl –type cbr –nn 10 –seed 1 –mc 5 –rate 5.0

nscbrgen.tcl –type cbr –nn 10 –seed 1 –mc 5 –rate 5.0 > cbr-10.tcl

gedit cbr-10.tcl

- Sample programs

Program 5.8: To create one node wireless in ns2

Define options

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) 1 ;# number of nodes

set val(rp) DSR ;# routing protocol

set val(x) 750 ;# X dimension of topography

set val(y) 550 ;# Y dimension of topography

set val(stop) 10.0 ;# time of simulation end

------Event scheduler object creation-------

set ns [new Simulator]

creating trace file and nam file

set tracefd [open wireless1.tr w]

set namtrace [open wireless1.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set up topography object

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

configure the nodes

```
$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) \
            -channelType $val(chan) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace OFF \
            -movementTrace ON
```

Creating node objects…

```
for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i< $val(nn) } {incri } {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }
```

Provide initial location of mobile nodes

```
$node(0) set X 50.0
$node(0) set Y 50.0
$node(0) set Z 0.0
```

Define node initial position in nam

```
for {set i 0} {$i< $val(nn)} { incri } {
$ns initialnodepos $node_($i) 30
}
```

Telling nodes when the simulation ends

```
for {set i 0} {$i< $val(nn) } { incri } {
  $ns at $val(stop) "$node_($i) reset";
}
```

Ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 10.01 "puts \"end simulation\"; $ns halt"

stop procedure:

proc stop {} {

    global ns tracefdnamtrace

    $ns flush-trace

    close $tracefd

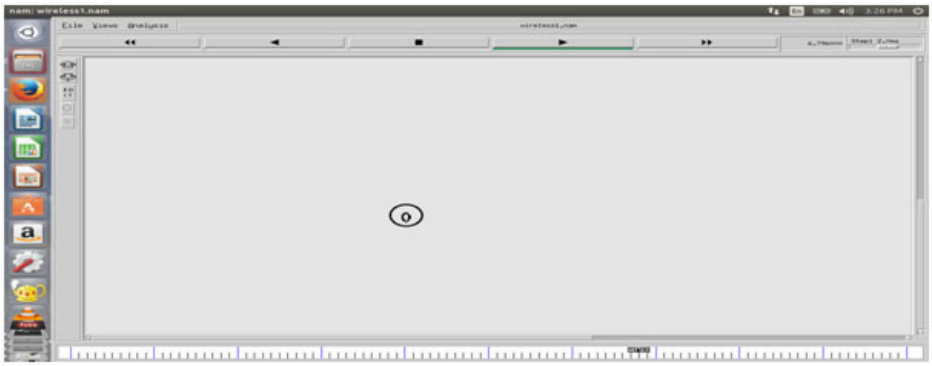
    close $namtrace

exec nam wireless1.nam &

}

$ns run

Output 5.8:



Program 5.9: Create two nodes in mobile wireless

Define options

```
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) 2;                           # number of nodes
set val(rp) DSR;                         # routing protocol
set val(x) 750;                          # X dimension of topography
set val(y) 550;                          # Y dimension of topography
set val(stop) 10.0;                      # time of simulation end
#-------Event scheduler object creation--------#
set ns [new Simulator]
```

Creating trace file and nam file

```
set tracefd [open wireless1.tr w]

set namtrace [open wireless1.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

configure the nodes

```
    $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) \
```

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF \

-movementTrace ON

Creating node objects

for {set i 0} {$i< $val(nn) } { incri } {

set node_($i) [$ns node]

}

for {set i 0} {$i< $val(nn) } {incri } {

$node_($i) color black

$ns at 0.0 "$node_($i) color black"

}

Provide initial location of mobile nodes

$node*(0) set X* 50.0

$node*(0) set Y* 50.0

$node*(0) set Z* 0.0

$node*(0) set X* 100.0

$node*(0) set Y* 100.0

$node*(0) set Z* 0.0

define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

Ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

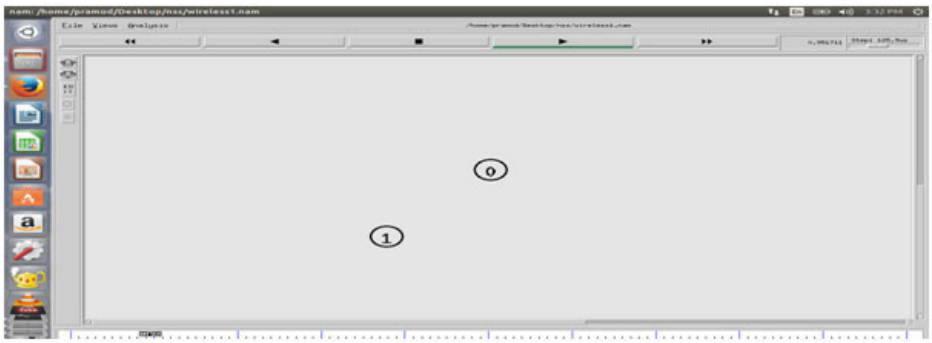
$ns at $val(stop) "stop"

$ns at 10.01 "puts \"end simulation\"; $ns halt"

stop procedure:

```
proc stop {} {
    global ns tracefdnamtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless1.nam &
}
$ns run
```

Output 5.9:



Program 5.10: Transfer data transfer between three wireless nodes

Define options

```
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 nodes
set val(rp) DSR;                         # routing protocol
set val(x) 750;                          # X dimension of topography
set val(y) 550;                            # Y dimension of topography
set val(stop) 10.0;                        # time of simulation end
```

```
#-------Event scheduler object creation--------#
set ns              [new Simulator]
#creating trace file and nam file
set tracefd   [open wireless1.tr w]
set windowVsTime2 [open win.tr w]
set namtrace   [open wireless1.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo          [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)
```

configure the nodes

```
        $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) \
                        -channelType $val(chan) \
```

```
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON
        for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
}
```

Provide initial location of mobile nodes

```
$node(0) set X 5.0
$node(0) set Y 5.0
$node(0) set Z 0.0
$node(1) set X 490.0
$node(1) set Y 285.0
$node(1) set Z 0.0
$node(2) set X 150.0
$node(2) set Y 240.0
$node(2) set Z 0.0
```

Generation of movements

```
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 19.0 "$node_(2) setdest 480.0 300.0 5.0"
```

Set a TCP connection between node*(0) and node*(1)

```
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
set tcp [new Agent/TCP/Newreno]
```

```
$tcp set class_ 2

set sink [new Agent/TCPSink]

$ns attach-agent $node_(1) $tcp

$ns attach-agent $node_(2) $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"
```

Define node initial position in nam

```
for {set i 0} {$i< $val(nn)} { incri } {
```

30 defines the node size for nam

```
$ns initial*node*pos $node_($i) 30

}
```

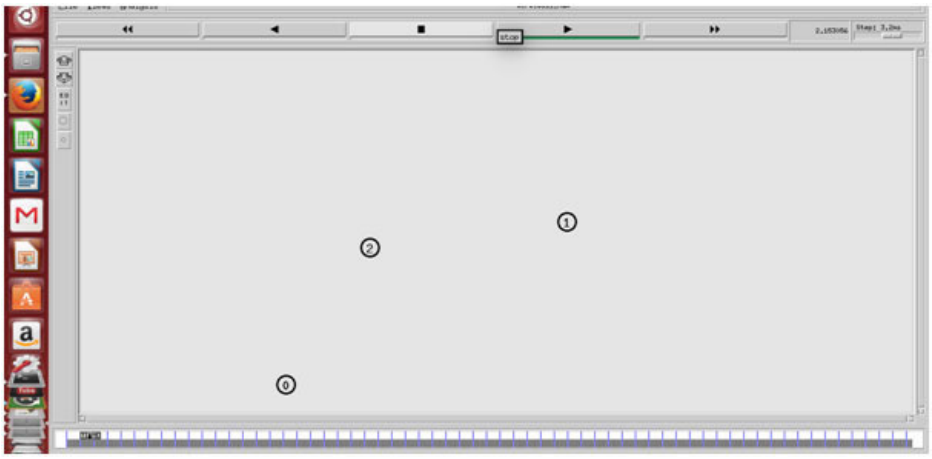
Telling nodes when the simulation ends

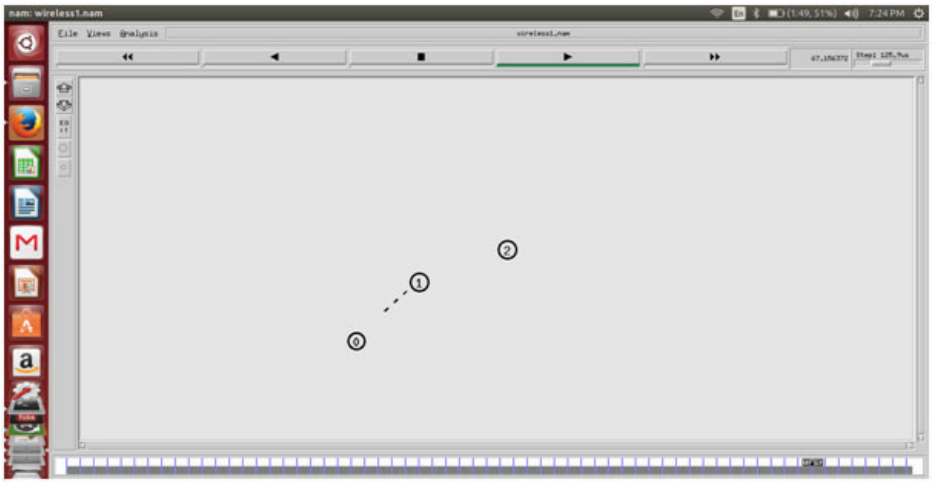
```
for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}
```

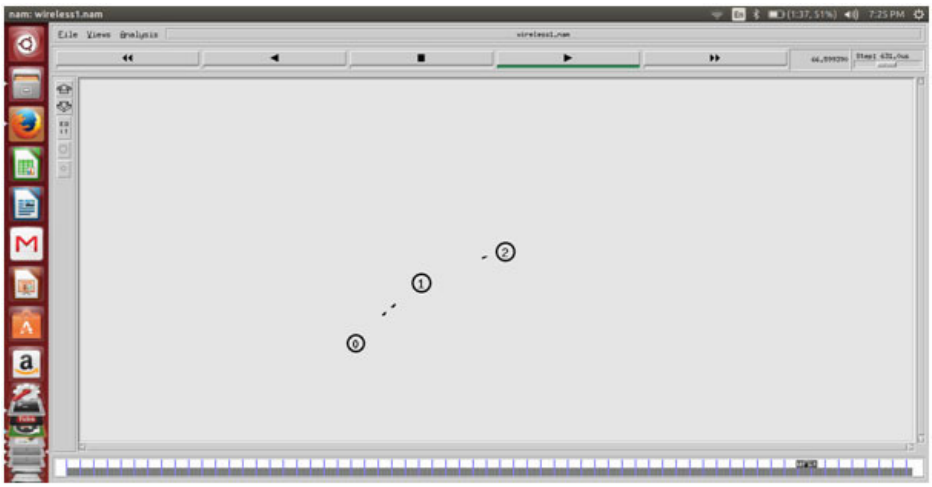
ending nam and the simulation

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

    global ns tracefdnamtrace

    $ns flush-trace

    close $tracefd

exec nam wireless1.nam &

}

$ns run
```

Output 5.10:







○ Wireless networks

Program 5.11: Wireless Network Construction using TCL script

Program Description

Basic wireless construction with number of nodes contained is three. The procedure to create nam file and trace file is given in this program. Topology is created by giving the position to the nodes and is specified by X, Y, and Z coordinates. Here, initial size of each and everynode are built by using initial*node*pos . The routing protocol, which is used in this program, is Adhoc On-demand Vector (AODV) Routing Protocol and simulation end time is 10ms.

File Name: program 5.11.tcl

- **Channel Type:** Wireless Channel
- **Propagation:** Two Ray Ground Model
- **X dimension:** 500
- **Y dimension:** 400
- **# Define options**

```
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 nodes
set val(rp)     AODV;                        # routing protocol
set val(x)      500;                         # X dimension of
                                               topography
set val(y)       400;                        # Y dimension of
topographyset val(stop) 10.0;                # time of simulation end
```

#-------Event scheduler object creation--------#

```
set ns [new Simulator]
```

Creating trace file and nam file

```
set tracefd [open wireless1.tr w]

set namtrace [open wireless1.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

configure the nodes

```
        $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) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
```

Creating node objects…

```
for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
    }
    for {set i 0} {$i< $val(nn) } {incri } {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }
```

Provide initial location of mobile nodes

```
$node(0) set X 50.0
$node(0) set Y 50.0
$node(0) set Z 0.0
$node(1) set X 200.0
$node(1) set Y 250.0
$node(1) set Z 0.0
$node(2) set X 300.0
$node(2) set Y 300.0
$node(2) set Z 0.0
```

Define node initial position in nam

```
for {set i 0} {$i< $val(nn)} { incri } {
$ns initialnodepos $node_($i) 30
```

}

Telling nodes when the simulation ends

```
for {set i 0} {$i< $val(nn) } { incri } {
    $ns at $val(stop) "$node_($i) reset";
}
```

Ending nam and the simulation

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\"; $ns halt"
```

stop procedure:

```
proc stop {} {
    global ns tracefdnamtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless1.nam &
}
$ns run
```

snapshot of the program output

Procedure to run the program in the terminal window: $ns program5.11.tcl

Output 5.11:

Program 5.12: Code for the construction of wireless nodes with fixed colors

Program Description

Number of nodes in the network is eight which are created and configured as mobile wireless nodes. Procedure for the creation of nam file and

trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y, and Z coordinates and the Z coordinates are always remains zero. Routing protocol is AODV and the stop time of the simulation is 10ms. Here, all the nodes are created in cyan color.

Channel Type: Wireless Channel

Propagation: Two Ray Ground Model

QueueType: DropTail

AntennaType: Omni Directional Antenna

Number of nodes: 8

Routing protocol: AODV

X dimension: 500

Y dimension: 400

Stop time: 10ms

Color: cyan color

File Name: program5.12.tcl

Define options

```
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 nodes
set val(rp)     AODV;                       # routing protocol
set val(x)      500;                        # X dimension of topography
set val(y)       400;                       # Y dimension of topographyset
val(stop)       10.0;                       # time of simulation end
#-------Event scheduler object creation--------#
set ns              [new Simulator]
```

Creating trace file and nam file.

```
set tracefd         [open wireless2.tr w]
set namtrace        [open wireless2.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

configure the nodes

```
        $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) \
                        -channelType $val(chan) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                    -movementTrace ON
```

Creating node objects..

```
for {set i 0} {$i< $val(nn) } { incri } {
                set node_($i) [$ns node]
        }
        for {set i 0} {$i< $val(nn) } {incri } {
                $node_($i) color cyan
                $ns at 0.0 "$node_($i) color cyan"
}
```

Provide initial location of mobilenodes

```
$node(0) set X 5.0

$node(0) set Y 30.0

$node(0) set Z 0.0

$node(1) set X 50.0
```

$node*(1) set X* 25.0

$node*(1) set Y* 0.0

$node*(2) set X* 200.0

$node*(2) set Y* 90.0

$node*(2) set Z* 0.0

$node*(3) set X* 350.0

$node*(3) set Y* 160.0

$node*(3) set Z* 0.0

$node*(4) set X* 100.0

$node*(4) set Y* 250.0

$node*(4) set Z* 0.0

$node*(5) set X* 300.0

$node*(5) set Y* 100.0

$node*(5) set Z* 0.0

$node*(6) set X* 400.0

$node*(6) set Y* 350.0

$node*(6) set Z* 0.0

$node*(7) set X* 3s50.0

$node*(7) set Y* 470.0

$node*(7) set Z* 0.0

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

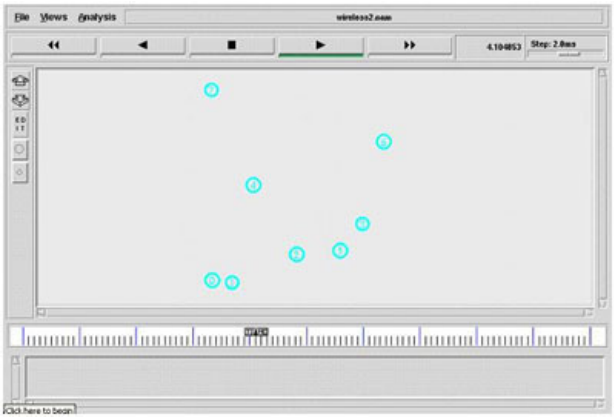
$ns at $val(stop) "stop"

```
$ns at 10.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
    global ns tracefdnamtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
exec nam wireless2.nam &
}
$ns run
```

How to run the program

Procedure to run the program in the terminal window: $ns program5.12.tcl

Output 5.12:



Program 5.13: Dynamic node creation program using AODV protocol TCL script

Program Description

Number of nodes present in the network is not static in this program. Number of nodes constructed is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y, and Z coordinates and the Z coordinates remains zero. Routing protocol is AODV and the stop time of the simulation is 10ms. Here, all the nodes are created in yellow color.

File Name: program 5.13.tcl

```
if {$argc != 1} {
        error "\nCommand: ns wireless1.tcl <no.of.mobile-
nodes>\n\n "
}
## Setting the wireless channels
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)      [lindex $argv 0]          ;# number of mobilenodes
set val(rp)      AODV                      ;# routing protocol
set val(x)       600                       ;# X dimension of topography
set val(y)       600                       ;# Y dimension of topography
set val(stop)    10                        ;# time of simulation end
#-------Event scheduler object creation--------#
set ns           [new Simulator]
```

creating the trace file and nam file

```
set tracefd       [open wireless1.tr w]
set namtrace      [open wireless1.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

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

configure the nodes

```
    $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) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
```

Creating node objects..

```
for {set i 0} {$i< $val(nn) } { incri } {
set node_($i) [$ns node]
}
for {set i 0} {$i< $val(nn) } {incri } {
$node_($i) color yellow
$ns at 0.0 "$node_($i) color yellow"
}
```

Provide initial location of mobilenodes

```
$node(0) set X 27.0
$node(0) set Y 260.0
$node(0) set Z 0.0
$node(1) set X 137.0
$node(1) set Y 348.0
$node(1) set Z 0.0
$node(2) set X 294.0
$node(2) set Y 235.0
$node(2) set Z 0.0
$node(3) set X 414.0
$node(3) set Y 342.0
$node(3) set Z 0.0
$node(4) set X 562.0
$node(4) set Y 267.0
```

$node(4) set Z 0.0

$node(5) set X 279.0

$node(5) set Y 447.0

$node(5) set Z 0.0

$node(6) set X -128.0

$node(6) set Y 260.0

$node(6) set Z 0.0

$node(7) set X 727.0

$node(7) set Y 269.0

$node(7) set Z 0.0

$node(8) set X 130.0

$node(8) set Y 126.0

$node(8) set Z 0.0

$node(9) set X 318.0

$node(9) set Y 45.0

$node(9) set Z 0.0

$node(10) set X 505.0

$node(10) set Y 446.0

$node(10) set Z 0.0

$node(11) set X 421.0

$node(11) set Y 158.0

$node(11) set Z 0.0

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initialnodepos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

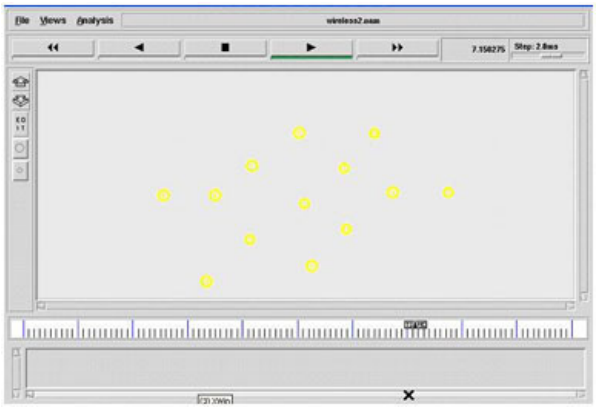
}

ending nam and the simulation

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 10.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec nam wireless1.nam &

}

$ns run
```

Procedure to run the program in the terminal window: $ns program5.13.tcl

Output 5.13:



Program 5.14: Dynamic node creation program and its initial location using AODV protocol TCL script

Program Description

Number of nodes in the network is not static in this program and the nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is specified by using the X, Y, and Z coordinates and the Z coordinates remains zero. Here, initial size of each and every node is created by using the command (initial*node*pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here, all the nodes are created in yellow color.

File Name: program 5.14.tcl

X dimension: 600

Y dimension: 600

Stop time: 10 ms

Color: Yellow color

Initial Node Position: 30

```
if {$argc != 1} {

    error "\nCommand: ns wireless3.tcl <no.of.mobile-nodes>\n\n "

}
```

Setting the wireless channels

```
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)     [lindex $argv 0]            ;# number of mobilenodes
set val(rp)     AODV                        ;# routing protocol
set val(x)      600                         ;# X dimension of topography
set val(y)      600                         ;# Y dimension of topography
set val(stop)   10                          ;# time of simulation end
```

```
#-------Event scheduler object creation--------#
set ns [new Simulator]
```

creating the trace file and nam file

```
set tracefd [open wireless3.tr w]

set namtrace [open wireless3.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo        [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

configure the nodes

```
    $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) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON
```

Creating node objects..

```
        for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
        }
        for {set i 0} {$i< $val(nn) } {incri } {
        $node_($i) color gold
        $ns at 0.0 "$node_($i) color gold"
    }
```

Provide initial location of mobilenodes..

```
        for {set i 0} {$i< $val(nn) } { incri } {
                set xx [expr rand()*600]
                set yy [expr rand()*600]
                $node($i) set X $xx
                $node($i) set Y $yy
          }
```

Define node initial position in nam

```
for {set i 0} {$i< $val(nn)} { incri } {
```

30 defines the node size for nam

```
$ns initialnodepos $node_($i) 30
}
```

Telling nodes when the simulation ends

```
for {set i 0} {$i< $val(nn) } { incri } {
$ns at $val(stop) "$node_($i) reset";
}
```

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 10.01 "puts \"end simulation\" ; $ns halt"

stop procedure…

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

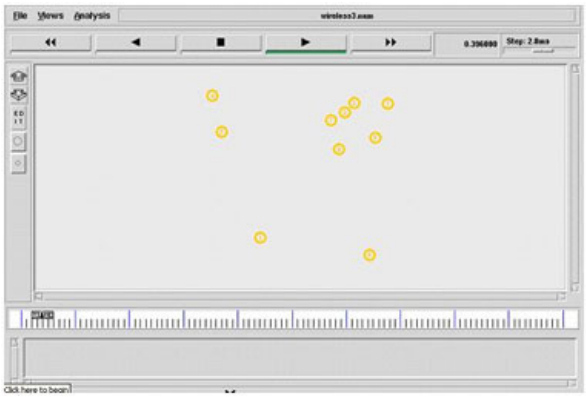
exec nam wireless3.nam &

}

$ns run

Procedure to run the program in the terminal window: $ns program 5.14.tcl

Output 5.14:



Program 5.15: Dynamic color creation of the program and the initial location of nodes are created by using AODV routing protocol by using TCL script

Program Description

Number of nodes in the network is static in this program. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is not static. X and Y coordinates are randomly selected, and the Z coordinates are always remains zero. Here, initial size of each and every node is created by using the command (initial*node*pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here, the nodes color will get modified dynamically according to the time period

File Name: program 5.15.tcl

Number of nodes: 4

X dimension: 750

Y dimension: 550

Stop time: 3.0ms

Color: Yellow color

Initial Node Position: 30

Setting the wireless channels

```
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)  5                          ;# max packet in ifq
set val(nn)      [lindex $argv 0]           ;# number of mobilenodes
set val(rp)      DSR                        ;# routing protocol
set val(x)       750                        ;# X dimension of topography
set val(y)       550                        ;# Y dimension of topography
set val(stop)    10.0                       ;# time of simulation end
```

------Event scheduler object creation-------

set ns [new Simulator]

Create a trace file and namfile..

set tracefd [open wireless1.tr w]

set namtrace [open wireless1.nam w]

Trace the nam and trace details from the main simulation..

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set up topography object..

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

Color Descriptions..

$ns color 1 dodgerblue

$ns color 2 blue

$ns color 3 cyan

$ns color 4 green

$ns color 5 yellow

$ns color 6 black

$ns color 7 magenta

$ns color 8 gold

$ns color 9 red

Array for dynamic color settings…

set colorname(0) blue

set colorname(1) cyan

set colorname(2) green

set colorname(3) red

set colorname(4) gold

set colorname(5) magenta

Setting The Distance Variables..

For model 'TwoRayGround'

set dist(5m) 7.69113e-06

set dist(9m) 2.37381e-06

set dist(10m) 1.92278e-06

set dist(11m) 1.58908e-06

set dist(12m) 1.33527e-06

set dist(13m) 1.13774e-06

set dist(25m) 3.07645e-07

set dist(30m) 2.13643e-07

set dist(35m) 1.56962e-07

set dist(40m) 1.56962e-10

set dist(45m) 1.56962e-11

set dist(50m) 1.20174e-13

#Phy/WirelessPhy set CSThresh_ $dist(50m)

#Phy/WirelessPhy set RXThresh_ $dist(50m)

Setting node config event with set of inputs..

$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) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
```

Creating node objects…

```
for {set i 0} {$i< $val(nn) } { incri } {
set node_($i) [$ns node]
}
for {set i 0} {$i< $val(nn) } {incri } {
$node_($i) color blue
$ns at 0.0 "$node_($i) color blue"
}
```

Provide initial location of mobilenodes…

```
for {set i 0} {$i< $val(nn) } { incri } {
    set xx [expr rand()*600]
    set yy [expr rand()*500]
    $node($i) set X $xx
    $node($i) set Y $yy
    $node($i) set Z 0.0
}
```

Define node initial position in nam…

```
for {set i 0} {$i< $val(nn)} { incri } {
# 30 defines the node size for nam..
$ns initialnodepos $node_($i) 30
}
```

Dynamic color procedure..

```
$ns at 0.0 "dynamic-color"

proc dynamic-color {} {

        global ns val node_ colorname

        set time 0.3

        set now [$ns now]

        set Rand [expr round(rand()*5)]

        for {set i 0} {$i< $val(nn) } {incri } {

        $node_($i) color $colorname($Rand)

        $ns at $now "$node_($i) color $colorname($Rand)"

        }

        $ns at [expr $now+$time] "dynamic-color"

}
```

stop procedure..

```
$ns at $val(stop) "stop"

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

puts "running nam…"

exec nam wireless1.nam &

exit 0

}

$ns runs
```

Procedure to run the program in the terminal window: $ns program 5.15.tcl

Output 5.15:

Program 5.16: Node mobility construction program using DSR routing protocol TCL script

Program Description

Number of nodes in the network is static. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates is zero. Movement for each and every node is built with static speed and specified the receiver address which is randomly generated and also the mobility will get change according to the time period. Here, initial size of each and every node is created by using the command (initial*node*pos). Routing protocol is DSR and the stop time of the simulation is 10ms.

File Name: program5.16.tcl

X dimension: 750

Y dimension: 550

Stop time: 3.0 ms

Color: Yellow color

Initial Node Position: 30

```
if {$argc != 1} {
      error "\nCommand: ns program5.9.tcl <no.of.mobile-
nodes>\n\n "
}
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) 5                          ;# max packet in ifq
set val(nn)    [lindex $argv 0]            ;# number of mobilenodes
set val(rp)    DSR                         ;# routing protocol
set val(x)     750                         ;# X dimension of topography
set val(y)     550                         ;# Y dimension of topography
set val(stop)  10.0                        ;# time of simulation end
```

------Event scheduler object creation-------

set ns [new Simulator]

Create a trace file and namfile..

set tracefd [open wireless2.tr w]

set namtrace [open wireless2.nam w]

Trace the nam and trace details from the main simulation..

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set up topography object..

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

Color Descriptions..

$ns color 1 dodgerblue

$ns color 2 blue

$ns color 3 cyan

$ns color 4 green

$ns color 5 yellow

$ns color 6 black

$ns color 7 magenta

$ns color 8 gold

$ns color 9 red

Setting The Distance Variables..

For model 'Two Ray Ground'

set dist(5m) 7.69113e-06

set dist(9m) 2.37381e-06

set dist(10m) 1.92278e-06

set dist(11m) 1.58908e-06

set dist(12m) 1.33527e-06

set dist(13m) 1.13774e-06

set dist(25m) 3.07645e-07

set dist(30m) 2.13643e-07

set dist(35m) 1.56962e-07

set dist(40m) 1.56962e-10

set dist(45m) 1.56962e-11

set dist(50m) 1.20174e-13

#Phy/WirelessPhy set CSThresh_ $dist(50m)

```
#Phy/WirelessPhy set RXThresh_ $dist(50m)
```

Setting node config event with set of inputs..

```
$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) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON
```

Creating node objects..

```
        for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
        }
        for {set i 0} {$i< 4 } {incri } {
        $node_($i) color yellow
        $ns at 0.0 "$node_($i) color yellow"
        }
        for {set i 4} {$i< 10 } {incri } {
        $node_($i) color red
        $ns at 3.0 "$node_($i) color red"
        }
        for {set i 10} {$i< 15 } {incri } {
        $node_($i) color blue
        $ns at 5.0 "$node_($i) color blue"
}
```

Provide initial location of mobilenodes..

$node*(0) set X* 27.0

$node*(0) set Y* 260.0

$node*(0) set Z* 0.0

$node*(1) set X* 137.0

$node*(1) set Y* 348.0

$node*(1) set Z* 0.0

$node*(2) set X* 294.0

$node*(2) set Y* 235.0

$node*(2) set Z* 0.0

$node*(3) set X* 414.0

$node*(3) set Y* 342.0

$node*(3) set Z* 0.0

$node*(4) set X* 562.0

$node*(4) set Y* 267.0

$node*(4) set Z* 0.0

$node*(5) set X* 279.0

$node*(5) set Y* 447.0

$node*(5) set Z* 0.0

$node*(6) set X* -128.0

$node*(6) set Y* 260.0

$node*(6) set Z* 0.0

$node*(7) set X* 727.0

$node*(7) set Y* 269.0

$node*(7) set Z* 0.0

$node*(8) set X* 130.0

$node*(8) set Y* 126.0

$node*(8) set Z* 0.0

$node*(9) set X* 318.0

$node*(9) set Y* 45.0

$node*(9) set Z* 0.0

$node*(10) set X* 505.0

$node*(10) set Y* 446.0

$node*(10) set Z* 0.0

$node*(11) set X* 421.0

$node*(11) set Y* 158.0

```
$node(11) set Z 0.0

$node(12) set X 72.0

$node(12) set Y 397.0

$node(12) set Z 0.0

if {$val(nn) >12} {

for {set i 13} {$i< $val(nn) } { incri } {

        set xx [expr rand()*600]

        set yy [expr rand()*500]

        $node($i) set X $xx

        $node($i) set Y $yy

        $node($i) set Z 0.0

}

}
```

Define node initial position in nam..

```
for {set i 0} {$i< $val(nn)} { incri } {

# 30 defines the node size for nam..

$ns initialnodepos $node_($i) 30

}
```

Stop procedure..

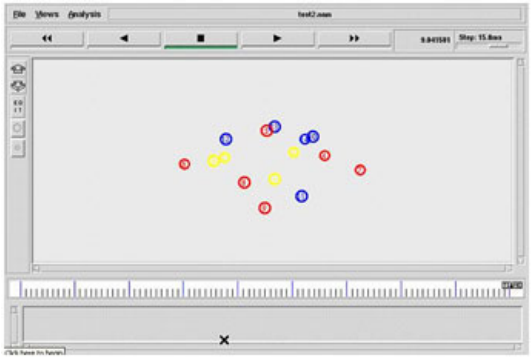
```
$ns at 0.0 "destination"

proc destination {} {

        global ns val node_

        set time 1.0

        set now [$ns now]

        for {set i 0} {$i<$val(nn)} {incri} {

        set xx [expr rand()*600]

        set yy [expr rand()*500]

        $ns at $now "$node_($i) setdest $xx $yy 20.0"

        }

        $ns at [expr $now+$time] "destination"

}

$ns at $val(stop) "stop"
```

stop procedure:

```
proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

puts "running nam…"

exec nam wireless2.nam &

exit 0

}

$ns run
```

Procedure to run the program in the terminal window: $ns program 5.16.tcl

Output 5.16:



Program 5.17: Creation of TCP (Transmission Control Protocol) communication between the nodes using AODV routing protocol TCL script

Program Description

Number of nodes in the network is three and are static. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates remains zero. Movement for each and every node is built with static speed and specified receiver address which is randomly generated and also the mobility will get change. Here, initial size of each and every node is created by using the command (initial*node*pos). Routing protocol is AODV and the stop time of the simulation is 150ms. Three nodes are created which are node 0, node 1, and node 2. Send TCP agent is created and attached to node 0, destination TCP sink agent is created and attached to node 1. Then, the TCP agent and the TCP sink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now, the communication is initiated.

File Name: program 5.17.tcl

X dimension: 500

Y dimension: 400

Stop time: 150 ms

Define setting option

```
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)     AODV                       ;# routing protocol
set val(x)      500                        ;# X dimension of topography
set val(y)      400                        ;# Y dimension of topography
set val(stop)   150                        ;# time of simulation end
```

------Event scheduler object creation-------

```
set ns              [new Simulator]
```

creating trace file and nam file

```
set tracefd         [open wireless1.tr w]
set windowVsTime2 [open win.tr w]
set namtrace        [open wireless1.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

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

configure the nodes

```
    $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) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON
        for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
    }
```

Provide initial location of mobilenodes

$node*(0) set X* 5.0

$node*(0) set Y* 5.0

$node*(0) set Z* 0.0

$node*(1) set X* 490.0

$node*(1) set Y* 285.0

$node*(1) set Z* 0.0

$node*(2) set X* 150.0

$node*(2) set Y* 240.0

$node*(2) set Z* 0.0

Generation of movements

$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"

$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"

$ns at 19.0 "$node_(2) setdest 480.0 300.0 5.0"

Set a TCP connection between node*(0) and node*(1)

set tcp [new Agent/TCP/Newreno]

$tcp set class_ 2

set sink [new Agent/TCPSink]

$ns attach-agent $node_(0) $tcp

$ns attach-agent $node_(1) $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

set tcp [new Agent/TCP/Newreno]

$tcp set class_ 2

set sink [new Agent/TCPSink]

$ns attach-agent $node_(1) $tcp

$ns attach-agent $node_(2) $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

Printing the window size

proc plotWindow {tcpSource file} {

global ns

set time 0.01

set now [$ns now]

set cwnd [$tcpSource set cwnd_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file" }

$ns at 10.0 "plotWindow $tcp $windowVsTime2"

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

```
$ns at $val(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec namsimwrls.nam&

}

$ns run
```

Procedure to run the program in the terminal window: $ns program 5.17.tcl

Output 5.17:

Program 5.18: Creation of TCP (Transmission Control Protocol) communication between the nodes by using DSR routing protocol TCL script

Program Description

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. Value of X and Y coordinates are given in the program and the value of Z coordinates are zero. Movement for each and every node is built with static speed and specified receiver address which is randomly generated and also the mobility will get change according to the time period. Here, initial size of each and every node is created by the use of the command (initial*node*pos). Routing protocol is DSR and the stop time of the simulation is 150ms. Three nodes are created which are node 0, node 1, and node 2. Send TCP agent is created and attached to node 0, destination TCP sink agent is created and attached to node 1. Then, the TCP agent and the TCP sink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now, the communication is initiated.

File Name: program5.18.tcl

X dimension: 500

Y dimension: 400

Stop time: 150 ms

Define setting option

```
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)    DSR                        ;# routing protocol
set val(x)     500                        ;# X dimension of topography
set val(y)     400                        ;# Y dimension of topography
set val(stop)  150                        ;# time of simulation end
```

------Event scheduler object creation-------

```
set ns              [new Simulator]
```

Creating trace file and nam file

```
set tracefd         [open dsr.tr w]
set windowVsTime2 [open win.tr w]
set namtrace        [open dsr.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

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

configure the nodes

```
    $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) \
        -channelType $val(chan) \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace OFF \
        -movementTrace ON
    for {set i 0} {$i< $val(nn) } { incri } {
    set node_($i) [$ns node]
    }
```

Provide initial location of mobilenodes

```
$node(0) set X 5.0
$node(0) set Y 5.0
$node(0) set Z 0.0
$node(1) set X 490.0
$node(1) set Y 285.0
$node(1) set Z 0.0
$node(2) set X 150.0
$node(2) set Y 240.0
$node(2) set Z 0.0
```

Generation of movements

```
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 300.0 5.0"
```

Set a TCP connection between node*(0) and node*(1)

```
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
```

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

Printing the window size

proc plotWindow {tcpSource file} {

global ns

set time 0.01

set now [$ns now]

set cwnd [$tcpSource set cwnd_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file" }

$ns at 10.1 "plotWindow $tcp $windowVsTime2"

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec namdsr.nam&

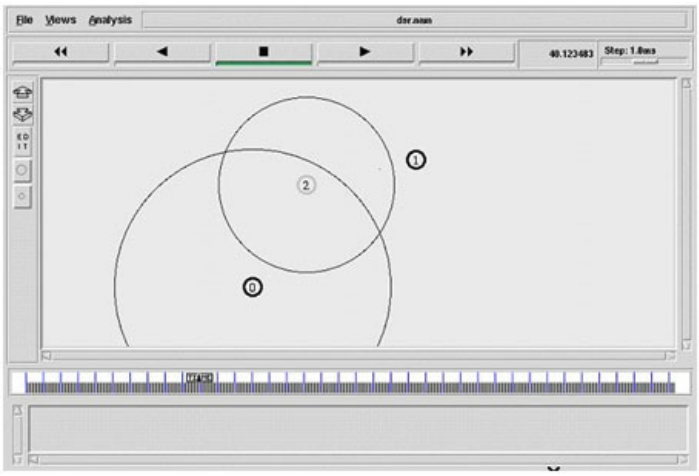
exit 0

}

$ns run

Procedure to run the program in the terminal window: $ns program 5.18.tcl

Output 5.18:



Program 5.19: Creation of UDP (User Datagram Protocol) communication between nodes with CBR traffic by using AODV routing protocol TCL script

Program Description

Number of nodes in the network is static and is declared as 22 in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the value of Z coordinates remains zero. Movement for each and every node is built with static speed and specified receiver address which is randomly generated and also the mobility will get change according to the time period. Here, initial size of each and every node is created by the use of the command (initial*node*pos). Routing protocol is AODV and the stop time of the simulation is 150ms. Send UDP agent is created and attached to sender node, destination UDP Null agent is created and attached to destination node. Then, the UDP agent and the UDP Null agent are connected. In the next level, CBR application is created and attached to the sender UDP agent. Now, the communication is initiated.

File Name: program5.19.tcl

Number or nodes: 22

X dimension: 1800

Y dimension: 840

Stop time: 150 ms

Define setting option

```
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)     22                         ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      1800                       ;# X dimension of topography
set val(y)      840                        ;# Y dimension of topography
set val(stop)   150                        ;# time of simulation end
```

Setting The Simulator Objects

```
set ns_ [new Simulator]
```

create the nam and trace file:

```
set tracefd [open aodv.tr w]

$ns_ trace-all $tracefd

set namtrace [open aodv.nam w]

$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

set chan1 [new $val(chan)]
```

Setting The Distance Variables

```
# For model 'TwoRayGround'

set dist(5m) 7.69113e-06

set dist(9m) 2.37381e-06

set dist(10m) 1.92278e-06

set dist(11m) 1.58908e-06

set dist(12m) 1.33527e-06

set dist(13m) 1.13774e-06

set dist(14m) 9.81011e-07
```

```
set dist(15m) 8.54570e-07

set dist(16m) 7.51087e-07

set dist(20m) 4.80696e-07

set dist(25m) 3.07645e-07

set dist(30m) 2.13643e-07

set dist(35m) 1.56962e-07

set dist(40m) 1.56962e-10

set dist(45m) 1.56962e-11

set dist(50m) 1.20174e-13

Phy/WirelessPhy set CSThresh_ $dist(50m)

Phy/WirelessPhy set RXThresh_ $dist(50m)
```

Defining Node Configuration

```
$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) \

  -topoInstance $topo \

  -agentTrace ON \

  -routerTrace ON \

  -macTrace ON \

  -movementTrace ON \

  -channel $chan1
```

Creating The WIRELESS NODES

```
set Server1 [$ns_ node]

set Server2 [$ns_ node]

set n2 [$ns_ node]

set n3 [$ns_ node]

set n4 [$ns_ node]

set n5 [$ns_ node]
```

```
set n6 [$ns_ node]

set n7 [$ns_ node]

set n8 [$ns_ node]

set n9 [$ns_ node]

set n10 [$ns_ node]

set n11 [$ns_ node]

set n12 [$ns_ node]

set n13 [$ns_ node]

set n14 [$ns_ node]

set n15 [$ns_ node]

set n16 [$ns_ node]

set n17 [$ns_ node]

set n18 [$ns_ node]

set n19 [$ns_ node]

set n20 [$ns_ node]

set n21 [$ns_ node]

set n22 [$ns_ node]

set opt(seed) 0.1

set a [ns-random $opt(seed)]

set i 0

while {$i< 5} {

incri

}
```

Setting The Initial Positions of Nodes

```
$Server1 set X_ 513.0

$Server1 set Y_ 517.0

$Server1 set Z_ 0.0

$Server2 set X_ 1445.0

$Server2 set Y_ 474.0

$Server2 set Z_ 0.0

$n2 set X_ 36.0

$n2 set Y_ 529.0

$n2 set Z_ 0.0

$n3 set X_ 143.0
```

```
$n3 set Y_ 666.0
$n3 set Z_ 0.0
$n4 set X_ 201.0
$n4 set Y_ 552.0
$n4 set Z_ 0.0
$n5 set X_ 147.0
$n5 set Y_ 403.0
$n5 set Z_ 0.0
$n6 set X_ 230.0
$n6 set Y_ 291.0
$n6 set Z_ 0.0
$n7 set X_ 295.0
$n7 set Y_ 419.0
$n7 set Z_ 0.0
$n8 set X_ 363.0
$n8 set Y_ 335.0
$n8 set Z_ 0.0
$n9 set X_ 334.0
$n9 set Y_ 647.0
$n9 set Z_ 0.0
$n10 set X_ 304.0
$n10 set Y_ 777.0
$n10 set Z_ 0.0
$n11 set X_ 412.0
$n11 set Y_ 194.0
$n11 set Z_ 0.0
$n12 set X_ 519.0
$n12 set Y_ 361.0
$n12 set Z_ 0.0
$n13 set X_ 569.0
$n13 set Y_ 167.0
$n13 set Z_ 0.0
$n14 set X_ 349.0
$n14 set Y_ 546.0
$n14 set Z_ 0.0
```

```
$n15 set X_ 466.0
$n15 set Y_ 668.0
$n15 set Z_ 0.0
$n16 set X_ 489.0
$n16 set Y_ 794.0
$n16 set Z_ 0.0
$n17 set X_ 606.0
$n17 set Y_ 711.0
$n17 set Z_ 0.0
$n18 set X_ 630.0
$n18 set Y_ 626.0
$n18 set Z_ 0.0
$n19 set X_ 666.0
$n19 set Y_ 347.0
$n19 set Z_ 0.0
$n20 set X_ 741.0
$n20 set Y_ 152.0
$n20 set Z_ 0.0
$n21 set X_ 882.0
$n21 set Y_ 264.0
$n21 set Z_ 0.0
$n22 set X_ 761.0
$n22 set Y_ 441.0
$n22 set Z_ 0.0
```

Giving Mobility to Nodes

```
$ns_ at 0.75 "$n2 setdest 379.0 349.0 20.0"
$ns_ at 0.75 "$n3 setdest 556.0 302.0 20.0"
$ns_ at 0.20 "$n4 setdest 309.0 211.0 20.0"
$ns_ at 1.25 "$n5 setdest 179.0 333.0 20.0"
$ns_ at 0.75 "$n6 setdest 139.0 63.0 20.0"
$ns_ at 0.75 "$n7 setdest 320.0 27.0 20.0"
$ns_ at 1.50 "$n8 setdest 505.0 124.0 20.0"
$ns_ at 1.25 "$n9 setdest 274.0 487.0 20.0"
$ns_ at 1.25 "$n10 setdest 494.0 475.0 20.0"
```

```
$ns_ at 1.25 "$n11 setdest 899.0 757.0 25.0"
$ns_ at 0.50 "$n12 setdest 598.0 728.0 25.0"
$ns_ at 0.25 "$n13 setdest 551.0 624.0 25.0"
$ns_ at 1.25 "$n14 setdest 397.0 647.0 25.0"
$ns_ at 1.25 "$n15 setdest 748.0 688.0 25.0"
$ns_ at 1.25 "$n16 setdest 842.0 623.0 25.0"
$ns_ at 1.25 "$n17 setdest 678.0 548.0 25.0"
$ns_ at 0.75 "$n18 setdest 741.0 809.0 20.0"
$ns_ at 0.75 "$n19 setdest 437.0 799.0 20.0"
$ns_ at 0.20 "$n20 setdest 159.0 722.0 20.0"
$ns_ at 1.25 "$n21 setdest 700.0 350.0 20.0"
$ns_ at 0.75 "$n22 setdest 839.0 444.0 20.0"
```

Setting The Node Size

```
$ns_ initial_node_pos $Server1 75
$ns_ initial_node_pos $Server2 75
$ns_ initial_node_pos $n2 40
$ns_ initial_node_pos $n3 40
$ns_ initial_node_pos $n4 40
$ns_ initial_node_pos $n5 40
$ns_ initial_node_pos $n6 40
$ns_ initial_node_pos $n7 40
$ns_ initial_node_pos $n8 40
$ns_ initial_node_pos $n9 40
$ns_ initial_node_pos $n10 40
$ns_ initial_node_pos $n11 40
$ns_ initial_node_pos $n12 40
$ns_ initial_node_pos $n13 40
$ns_ initial_node_pos $n14 40
$ns_ initial_node_pos $n15 40
$ns_ initial_node_pos $n16 40
$ns_ initial_node_pos $n17 40
$ns_ initial_node_pos $n18 40
$ns_ initial_node_pos $n19 40
$ns_ initial_node_pos $n20 40
```

$ns_ initial*node*pos $n21 40

$ns_ initial*node*pos $n22 40

Setting The Labels For Nodes

$ns_ at 0.0 "$Server1 label Server1"

$ns_ at 0.0 "$Server2 label Server2"

#Setting Color For Server

$Server1 color maroon

$ns_ at 0.0 "$Server1 color maroon"

$Server2 color maroon

$ns_ at 0.0 "$Server2 color maroon"

SETTING ANIMATION RATE

$ns_ at 0.0 "$ns_ set-animation-rate 15.0ms"

COLORING THE NODES

$n9 color blue

$ns_ at 4.71 "$n9 color blue"

$n5 color blue

$ns_ at 7.0 "$n5 color blue"

$n2 color blue

$ns_ at 7.29 "$n2 color blue"

$n16 color blue

$ns_ at 7.59 "$n16 color blue"

$n9 color maroon

$ns_ at 7.44 "$n9 color maroon"

$ns_ at 7.43 "$n9 label TTLover"

$ns_ at 7.55 "$n9 label \"\""

$n12 color blue

$ns_ at 7.85 "$n12 color blue"

Establishing Communication

set udp0 [$ns_ create-connection UDP $Server1 LossMonitor $n18 0]

$udp0 set fid_ 1

set cbr0 [$udp0 attach-app Traffic/CBR]

$cbr0 set packetSize_ 1000

$cbr0 set interval_ .07

```
$ns_ at 0.0 "$cbr0 start"

$ns_ at 4.0 "$cbr0 stop"

set udp1 [$ns_ create-connection UDP $Server1 LossMonitor $n22 0]

$udp1 set fid_ 1

set cbr1 [$udp1 attach-app Traffic/CBR]

$cbr1 set packetSize_ 1000

$cbr1 set interval_ .07

$ns_ at 0.1 "$cbr1 start"

$ns_ at 4.1 "$cbr1 stop"

set udp2 [$ns_ create-connection UDP $n21 LossMonitor $n20 0]

$udp2 set fid_ 1

set cbr2 [$udp2 attach-app Traffic/CBR]

$cbr2 set packetSize_ 1000

$cbr2 set interval_ .07

$ns_ at 2.4 "$cbr2 start"

$ns_ at 4.1 "$cbr2 stop"

set udp3 [$ns_ create-connection UDP $Server1 LossMonitor $n15 0]

$udp3 set fid_ 1

set cbr3 [$udp3 attach-app Traffic/CBR]

$cbr3 set packetSize_ 1000

$cbr3 set interval_ 5

$ns_ at 4.0 "$cbr3 start"

$ns_ at 4.1 "$cbr3 stop"

set udp4 [$ns_ create-connection UDP $Server1 LossMonitor $n14 0]

$udp4 set fid_ 1

set cbr4 [$udp4 attach-app Traffic/CBR]

$cbr4 set packetSize_ 1000

$cbr4 set interval_ 5

$ns_ at 4.0 "$cbr4 start"

$ns_ at 4.1 "$cbr4 stop"

set udp5 [$ns_ create-connection UDP $n15 LossMonitor $n16 0]

$udp5 set fid_ 1

set cbr5 [$udp5 attach-app Traffic/CBR]

$cbr5 set packetSize_ 1000
```

```
$cbr5 set interval_ 5

$ns_ at 4.0 "$cbr5 start"

$ns_ at 4.1 "$cbr5 stop"

set udp6 [$ns_ create-connection UDP $n15 LossMonitor $n17 0]

$udp6 set fid_ 1

set cbr6 [$udp6 attach-app Traffic/CBR]

$cbr6 set packetSize_ 1000

$cbr6 set interval_ 5

$ns_ at 4.0 "$cbr6 start"

$ns_ at 4.1 "$cbr6 stop"

set udp7 [$ns_ create-connection UDP $n14 LossMonitor $n4 0]

$udp7 set fid_ 1

set cbr7 [$udp7 attach-app Traffic/CBR]

$cbr7 set packetSize_ 1000

$cbr7 set interval_ 5

$ns_ at 4.0 "$cbr7 start"

$ns_ at 4.1 "$cbr7 stop"

set udp8 [$ns_ create-connection UDP $n14 LossMonitor $n9 0]

$udp8 set fid_ 1

set cbr8 [$udp8 attach-app Traffic/CBR]

$cbr8 set packetSize_ 1000

$cbr8 set interval_ 5

$ns_ at 4.0 "$cbr8 start"

$ns_ at 4.1 "$cbr8 stop"

set udp9 [$ns_ create-connection UDP $n4 LossMonitor $n3 0]

$udp9 set fid_ 1

set cbr9 [$udp9 attach-app Traffic/CBR]

$cbr9 set packetSize_ 1000

$cbr9 set interval_ 5

$ns_ at 4.0 "$cbr9 start"

$ns_ at 4.1 "$cbr9 stop"

set udp10 [$ns_ create-connection UDP $n4 LossMonitor $n2 0]

$udp10 set fid_ 1

set cbr10 [$udp10 attach-app Traffic/CBR]

$cbr10 set packetSize_ 1000
```

```
$cbr10 set interval_ 5

$ns_ at 4.0 "$cbr10 start"

$ns_ at 4.1 "$cbr10 stop"

set udp11 [$ns_ create-connection UDP $n9 LossMonitor $n16 0]

$udp11 set fid_ 1

set cbr11 [$udp11 attach-app Traffic/CBR]

$cbr11 set packetSize_ 1000

$cbr11 set interval_ 5

$ns_ at 4.0 "$cbr11 start"

$ns_ at 4.1 "$cbr11 stop"

set udp12 [$ns_ create-connection UDP $n9 LossMonitor $n10 0]

$udp12 set fid_ 1

set cbr12 [$udp12 attach-app Traffic/CBR]

$cbr12 set packetSize_ 1000

$cbr12 set interval_ 5

$ns_ at 4.0 "$cbr12 start"

$ns_ at 4.1 "$cbr12 stop"
```

ANNOTATIONS DETAILS

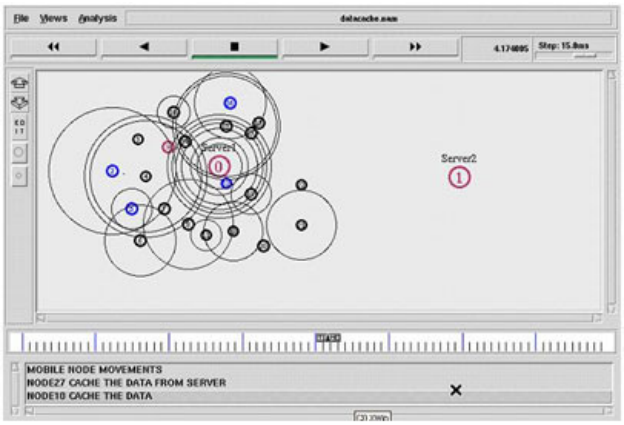
```
$ns_ at 0.0 "$ns_ trace-annotate \"MOBILE NODE MOVEMENTS\""

$ns_ at 4.1 "$ns_ trace-annotate \"NODE27 CACHE THE DATA FRO SERVER\""

#$ns_ at 4.59 "$ns_ trace-annotate \"PACKET LOSS AT NODE27\""

$ns_ at 4.71 "$ns_ trace-annotate \"NODE10 CACHE THE DATA\""
```

PROCEDURE TO STOP

```
proc stop {} {
        global ns_ tracefd
        $ns_ flush-trace
        close $tracefd
        exec namdatacache.nam&
        exit 0
    }

puts "Starting Simulation……."

$ns_ at 25.0 "stop"

$ns_ run
```

Procedure to run the program in the terminal window: $ns program 5.19.tcl

Output 5.19:



Program 5.20: Creation of TCP (Transmission Control Protocol) communication between the nodes by using DSDV routing protocol TCL script

Program Description

Number of nodes in the network is static and is declared as three in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates is zero. Movement for each and every node is built with static speed and specified receiver address which is randomly generated and also the mobility will get change according to the time period.

Here, initial size of each node is created by using the command (initial*node*pos). Routing protocol is DSDV and the stop time of the simulation is 150ms. Three nodes are created which are node 0, node 1, and node 2. Send TCP agent is created and attached to node 0, destination TCP sink agent is created and attached to node 1. Then, the TCP agent and the TCP sink agent are connected. In the next level, FTP application is created and attached to the sender TCP agent. Now, the communication is initiated.

File Name: program 5.20.tcl

Channel Type: Wireless Channel

Propagation: Two Ray Ground Model

Queue Type: DropTail

Antenna Type: Omni Directional Antenna

Number of nodes: 3

Routing protocol: DSDV

X dimension: 500

Y dimension: 400

Stop time: 150ms

AGENT: TCP

Application: FTP

Setting The wireless Channels..

```
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)     750                       ;# X dimension of topography
set val(y)     550                       ;# Y dimension of topography
set val(stop)  150                       ;# time of simulation end
```

Creating trace file and nam file

```
set tracefd       [open dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace      [open dsdv.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

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

configure the nodes

```
    $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) \
                    -channelType $val(chan) \
```

```
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON
        for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
        }
```

Provide initial location of mobilenodes

```
$node(0) set X 5.0
$node(0) set Y 5.0
$node(0) set Z 0.0
$node(1) set X 490.0
$node(1) set Y 285.0
$node(1) set Z 0.0
$node(2) set X 150.0
$node(2) set Y 240.0
$node(2) set Z 0.0
```

Generation of movements

```
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 110.0 "$node_(0) setdest 480.0 300.0 5.0"
```

Set a TCP connection between node*(0) and node*(1)

```
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
```

Printing the window size

proc plotWindow {tcpSource file} {

global ns

set time 0.01

set now [$ns now]

set cwnd [$tcpSource set cwnd_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file" }

$ns at 10.1 "plotWindow $tcp $windowVsTime2"

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 150.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec namdsdv.nam&

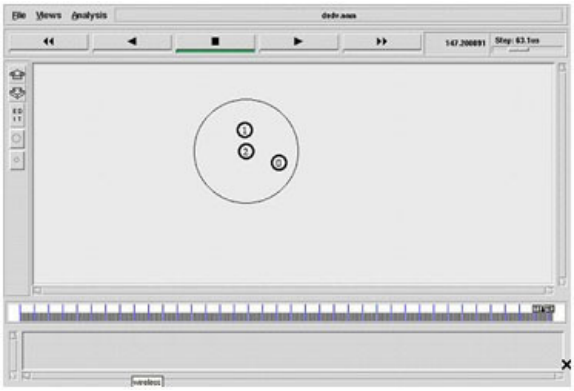
exit 0

}

$ns run

Procedure to run the program in the terminal window: $ns program5.20.tcl

Output 5.20:



Program 5.21: Mobile node energy model construction TCL script

Program Description

Number of nodes in the network is static and is declared as six in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are zero.

Movement for every node is built with static speed and specified receiver address is randomly generated and also the mobility will get change according to the time period. Here, initial size of each node is created by using the command (initial*node*pos). Routing protocol is DSR and the stop time of the simulation is 18ms.

File Name: program 5.21.tcl

Channel: Wireless Channel

Propagation: Two Ray Ground Propagation

Queue Type: Drop Tail

Antenna: Omni Directional Antenna

Initial Energy: 20 J

Transmission Power: 0.9 J

Receiver Power: 0.8 J

Idle Power: 0.0 J

Sense Power: 0.0175 J

Routing Protocol: DSR

Simulation Time: 18ms

Number of nodes: 6 nodes

X dimension: 750

Y dimension: 550

Initial Node Position: 30

Setting The wireless Channels..

```
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) 5                          ;# max packet in ifq
set val(nn)     6                          ;# number of mobilenodes
set val(rp)     DSR                        ;# routing protocol
set val(x)      750                        ;# X dimension of topography
set val(y)      550                        ;# Y dimension of topography
set val(stop)   18.0                       ;# time of simulation end
```

Create a simulator object(nothing but, a scheduler's object)..

```
set ns [new Simulator]
```

Create a trace file and namfile..

```
set tracefd [open wireless1.tr w]

set namtrace [open wireless1.nam w]
```

Trace the nam and trace details from the main simulation..

```
$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object..

```
set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

Color Descriptions..

```
$ns color 1 dodgerblue

$ns color 2 blue

$ns color 3 cyan

$ns color 4 green

$ns color 5 yellow

$ns color 6 black
```

$ns color 7 magenta

$ns color 8 gold

$ns color 9 red

Setting The Distance Variables..

For model 'TwoRayGround'

set dist(5m) 7.69113e-06

set dist(9m) 2.37381e-06

set dist(10m) 1.92278e-06

set dist(11m) 1.58908e-06

set dist(12m) 1.33527e-06

set dist(13m) 1.13774e-06

set dist(14m) 9.81011e-07

set dist(15m) 8.54570e-07

set dist(16m) 7.51087e-07

set dist(20m) 4.80696e-07

set dist(25m) 3.07645e-07

set dist(30m) 2.13643e-07

set dist(35m) 1.56962e-07

set dist(40m) 1.56962e-10

set dist(45m) 1.56962e-11

set dist(50m) 1.20174e-13

#Phy/WirelessPhy set CSThresh_ $dist(50m)

#Phy/WirelessPhy set RXThresh_ $dist(50m)

Setting node config event with set of inputs..

puts "Node Configuration Started here…\n \

-channel $val(chan) \n \

-adhocRouting $val(rp) \n \

-llType $val(ll) \n \

-macType $val(mac) \n \

-ifqType $val(ifq) \n \

-ifqLen $val(ifqlen) \n \

-antType $val(ant) \n \

-propType $val(prop) \n \

```
                -phyType $val(netif) \n"
    $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) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON
```

Energy model

```
        $ns node-config -energyModelEnergyModel \
                    -initialEnergy 20 \
                    -txPower 0.9 \
                    -rxPower 0.8 \
                    -idlePower 0.0 \
                    -sensePower 0.0175
```

Creating node objects..

```
        for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
        }
for {set i 0} {$i< $val(nn)} {incri} {
        $node_($i) color darkgreen
        $ns at 0.0 "$node_($i) color darkgreen"
    }
```

Provide initial location of mobilenodes..

```
        if {$val(nn) >0} {
        for {set i 1} {$i< $val(nn) } { incri } {
```

```
            set xx [expr rand()*600]
            set yy [expr rand()*500];
            $node($i) set X $xx
            $node($i) set Y $yy
        }
    }
```

set god distance..

```
        $god_ set-dist 0 1 2
        $god_ set-dist 0 2 2
        $god_ set-dist 0 3 2
        $god_ set-dist 0 4 1
        $god_ set-dist 0 5 2
        $god_ set-dist 1 2 3
        $god_ set-dist 1 3 3
```

Define node initial position in nam..

```
        for {set i 0} {$i< $val(nn)} { incri } {
        # 30 defines the node size for nam..
        $ns initialnodepos $node_($i) 30
        }
```

Telling nodes when the simulation ends..

```
for {set i 0} {$i< $val(nn) } { incri } {
$ns at $val(stop) "$node_($i) reset";
}
```

Ending nam and the simulation..

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 16.01 "puts \"end simulation\" " ;# $ns halt
```

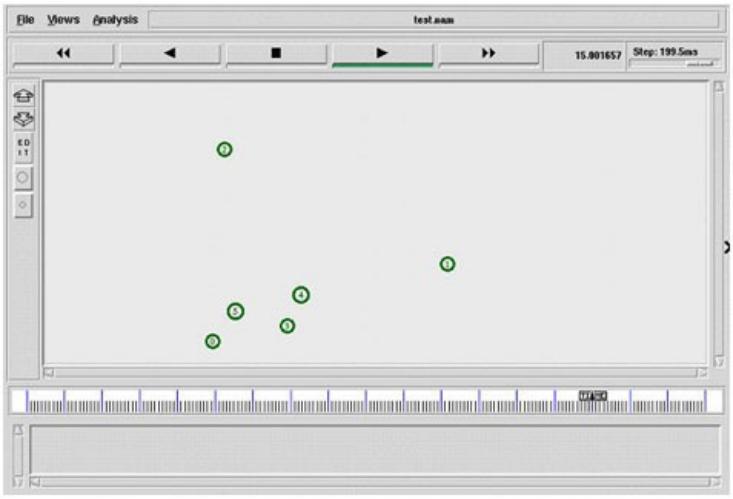
Stop procedure..

```
proc stop {} {
global ns tracefdnamtrace
$ns flush-trace
close $tracefd
```

```
close $namtrace

exec nam wireless1.nam &

exit 0

}

$ns run
```

Procedure to run the program in the terminal window: $ns program 5.21.tcl

Output 5.21:



Program 5.22: Creation of nodes at random destination at a particular time interval by using AODV routing protocol TCL script

Program Description

Number of nodes in the network is not static and is declared as six in the network. Nodes are configured in the mobile wireless node format. Number of nodes construction is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program.Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are zero. Movement for each and every node is built with static speed and specify receiver address which is randomly generated and also the destination location will get change according to the time period.

File Name: program 5.22.tcl

Define options

```
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) 5                             ;# number of mobilenodes
set val(rp) AODV                          ;# routing protocol
set val(x) 500                            ;# X dimension of topography
set val(y) 400                            ;# Y dimension of topography
set val(stop) 10                          ;# time of simulation end
set ns [new Simulator]
```

Creating nam and trace file:

```
set tracefd [open wireless1.tr w]
set namtrace [open wireless1.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

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

configure the nodes

```
$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) \
                -channelType $val(chan) \
```

```
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
```

Creating node objects..

```
for {set i 0} {$i< $val(nn) } { incri } {
    set node_($i) [$ns node]
    }
    for {set i 0} {$i< $val(nn) } {incri } {
    $node_($i) color black
    $ns at 0.0 "$node_($i) color black"
    }
```

Provide initial location of mobilenodes..

```
    for {set i 0} {$i< $val(nn) } { incri } {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $node(*$i) set X* $xx
        $node(*$i) set Y* $yy
    }
```

Define node initial position in nam

```
for {set i 0} {$i< $val(nn)} { incri } {
```

30 defines the node size for nam

```
$ns initial*node*pos $node_($i) 30
}
```

Telling nodes when the simulation ends

```
for {set i 0} {$i< $val(nn) } { incri } {
$ns at $val(stop) "$node_($i) reset";
}
```

Destination procedure..

```
$ns at 0.0 "destination"
```

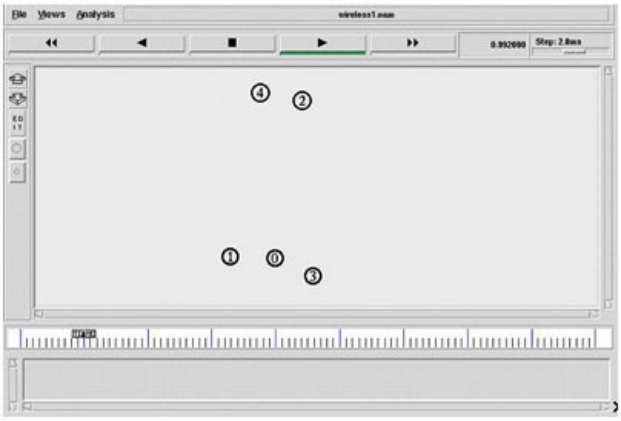
```
proc destination {} {
        global ns val node_
        set time 1.0
        set now [$ns now]
        for {set i 0} {$i<$val(nn)} {incri} {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $ns at $now "$node_($i) setdest $xx $yy 10.0"
        }
        $ns at [expr $now+$time] "destination"
}
$ns at $val(stop) "stop"
```

Stop procedure

```
proc stop {} {
global ns tracefdnamtrace
$ns flush-trace
close $tracefd
close $namtrace
exec nam wireless1.nam &
}
$ns run
```

Procedure to run the program in the terminal window: $ns program 5.22.tcl

Output 5.22:



Program 5.23: Creation of nodes destination and random coloring by using AODV routing protocol TCL script

Program Description

Number of nodes in the network is not static and is declared as eight in the network. Nodes are configured in the mobile wireless node format.

Number of nodes constructed is given during the run time of the program. The user should give the number of nodes in the terminal window during the execution of the program.Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. Values of X and Y coordinates are given in the program and the Z coordinates are zero. Movement for each and every node is built with static speed and specified receiver address which is randomly generated and also the destination location will get change according to the time period. Here, initial size of every node is created by the use of the command (initial*node*pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here, every group of the nodes is constructed with different type of colors.

File Name: program5.23.tcl

Define setting options

```
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)     8                          ;# number of mobilenodes
set val(rp)     AODV                       ;# routing protocol
set val(x)      500                        ;# X dimension of topography
set val(y)      400                        ;# Y dimension of topography
set val(stop)   10                         ;# time of simulation end
set ns          [new Simulator]
```

Creating nam and trace file:

```
set tracefd     [open wireless2.tr w]
set namtrace    [open wireless2.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

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

configure the nodes

```
    $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) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
```

Creating node objects..

```
for {set i 0} {$i< 3 } { incri } {
        set node_($i) [$ns node]
        }
        for {set i 0} {$i< 3 } {incri } {
        $node_($i) color blue
        $ns at 0.0 "$node_($i) color blue"
        }
for {set i 3} {$i< 6 } { incri } {
        set node_($i) [$ns node]
        }
        for {set i 3} {$i< 5 } {incri } {
        $node_($i) color cyan
        $ns at 0.0 "$node_($i) color cyan"
        }
for {set i 5} {$i< 8 } { incri } {
        set node_($i) [$ns node]
        }
        for {set i 5} {$i< 8 } {incri } {
        $node_($i) color red
        $ns at 0.0 "$node_($i) color red"
    }
```

Provide initial location of mobilenodes..

```
for {set i 0} {$i< $val(nn) } { incri } {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $node($i) set X $xx
        $node($i) set Y $yy
  }
```

Define node initial position in nam

```
for {set i 0} {$i< $val(nn)} { incri } {
```

30 defines the node size for nam

```
$ns initialnodepos $node_($i) 30
}
```

Telling nodes when the simulation ends

```
for {set i 0} {$i< $val(nn) } { incri } {
$ns at $val(stop) "$node_($i) reset";
}
```

dynamic destination setting procedure..

```
$ns at 0.0 "destination"
proc destination {} {
        global ns val node_
        set time 1.0
        set now [$ns now]
        for {set i 0} {$i<$val(nn)} {incri} {
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $ns at $now "$node_($i) setdest $xx $yy 10.0"
        }
        $ns at [expr $now+$time] "destination"
}
```

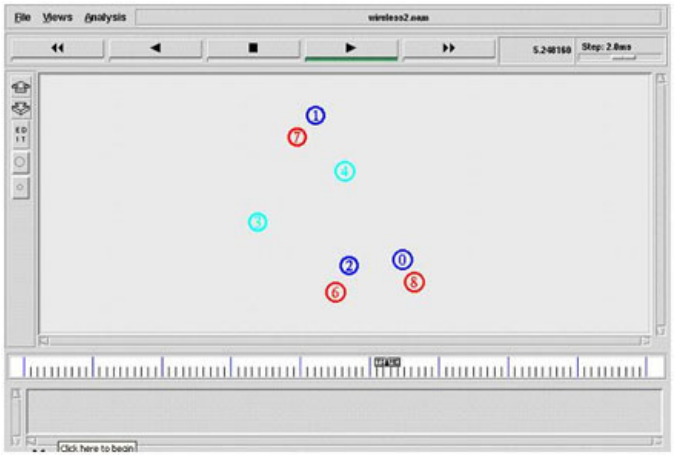
stop procedure..

```
$ns at $val(stop) "stop"
```

```
proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec nam wireless2.nam &

}

$ns run
```

Procedure to run the program in the terminal window: $ns program 5.23.tcl

Output 5.23:



Program 5.24: Creation of nodes with the initial and destination position in random manner by using AODV routing protocol TCL script

Program Description

Number of nodes in the network is not static and declared nodes are eight in the network. Nodes are configured in the mobile wireless node format. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are zero.

Movement for each node is built with static speed and the specified receiver address is randomly generated and also the destination location will get change according to the time period. Here, initial size of every node is created by using the command (initial*node*pos). Routing protocol is AODV and the stop time of the simulation is 10ms. Here, each and every group of the nodes is constructed with different type of colors.

File Name: program5.24.tcl

Channel: Wireless Channel

Propagation: Two Ray Ground Propagation

Queue Type: Drop Tail

Antenna: Omni Directional Antenna

Routing Protocol: AODV

Simulation Time: 18ms

Number of nodes: 8 nodes

X dimension: 500

Y dimension: 400

Initial Node Position: 30

Define options

```
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)   8                            ;# number of mobilenodes
set val(rp)   AODV                         ;# routing protocol
set val(x)    500                          ;# X dimension of topography
set val(y)    400                          ;# Y dimension of topography
set val(stop) 10                           ;# time of simulation end
```

Creating simulation:

```
set ns                [new Simulator]
```

Creating nam and trace file:

```
set tracefd       [open wireless3.tr w]

set namtrace      [open wireless3.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo          [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

configure the nodes

```
    $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) \
		-channelType $val(chan) \
		-topoInstance $topo \
		-agentTrace ON \
		-routerTrace ON \
		-macTrace OFF \
		-movementTrace ON
```

Creating node objects..

```
for {set i 0} {$i< 3 } { incri } {
		set node_($i) [$ns node]
		}
		for {set i 0} {$i< 3 } {incri } {
		$node_($i) color blue
		$ns at 0.0 "$node_($i) color blue"
		}
for {set i 3} {$i< 6 } { incri } {
		set node_($i) [$ns node]
		}
		for {set i 3} {$i< 6 } {incri } {
		$node_($i) color cyan
		$ns at 1.0 "$node_($i) color cyan"
		}
for {set i 6} {$i< 8 } { incri } {
		set node_($i) [$ns node]
		}
		for {set i 5} {$i< 8 } {incri } {
		$node_($i) color red
		$ns at 2.0 "$node_($i) color red"
    }
```

Provide initial location of mobilenodes..

```
		for {set i 0} {$i< $val(nn) } { incri } {
```

```
        set xx [expr rand()*500]
        set yy [expr rand()*400]
        $node($i) set X $xx
        $node($i) set Y $yy
    }
```

Define node initial position in nam

```
for {set i 0} {$i< $val(nn)} { incri } {
```

30 defines the node size for nam

```
$ns initialnodepos $node_($i) 30
}
```

Telling nodes when the simulation ends

```
for {set i 0} {$i< $val(nn) } { incri } {
$ns at $val(stop) "$node_($i) reset";
}
```

dynamic destination setting procedure..

```
$ns at 0.0 "destination"
proc destination {} {
    global ns val node_
    set time 1.0
    set now [$ns now]
    for {set i 0} {$i<$val(nn)} {incri} {
    set xx [expr rand()*500]
    set yy [expr rand()*400]
    $ns at $now "$node_($i) setdest $xx $yy 10.0"
    }
    $ns at [expr $now+$time] "destination"
}
```

stop procedure..

```
$ns at $val(stop) "stop"
proc stop {} {
global ns tracefdnamtrace
```

$ns flush-trace

close $tracefd

close $namtrace

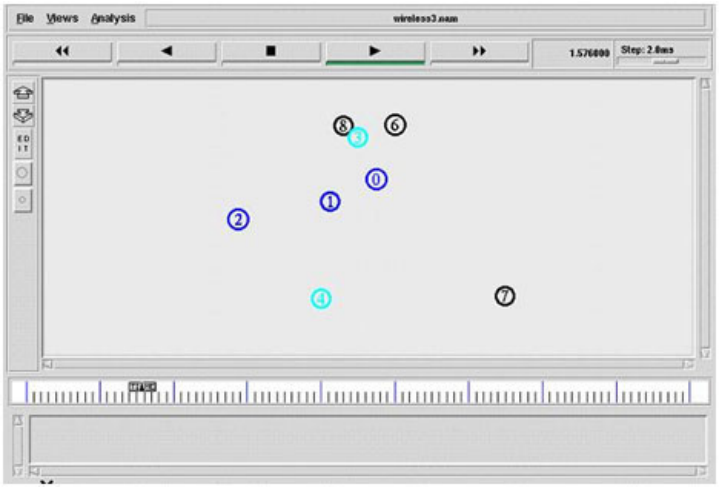
exec nam wireless3.nam &

}

$ns run

Procedure to run the program in the terminal window: $ns program 5.24.tcl

Output 5.24:



Program 5.25: Creation of graphs with X dimension and Y Dimensionrandomly by using AODV routing protocol and in TCL script

Program Description

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are zero. Graph is randomly generated by using the X and Y dimensions and is programmed to generate the trace file. Here, the trace file acts as an input file to plot the graph in the form of trace file. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name: program 5.25.tcl

Channel: Wireless Channel

Propagation: Two Ray Ground Propagation

Queue Type: Drop Tail

Antenna: Omni Directional Antenna

Routing Protocol: AODV

Simulation Time: 10ms

Initial Node Position: 30

Define options

```
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 nodes
set val(rp)    AODV;                       # routing protocol
set val(x)     500;                        # X dimension of topography
set val(y)     400;                        # Y dimension of topography
set val(stop) 10.0;                        # time of simulation end
```

Creating simulation

```
set ns [new Simulator]
```

Creating nam and trace file

```
set tracefd       [open Graph1.tr w]

set namtrace      [open Graph1.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

configure the nodes

```
    $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) \
```

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF \

-movementTrace ON

Creating node objects..

for {set i 0} {$i< $val(nn) } { incri } {

set node_($i) [$ns node]

}

for {set i 0} {$i< $val(nn) } {incri } {

$node_($i) color black

$ns at 0.0 "$node_($i) color black"

}

Provide initial location of mobilenodes

$node*(0) set X* 50.0

$node*(0) set Y* 50.0

$node*(0) set Z* 0.0

$node*(1) set X* 200.0

$node*(1) set Y* 250.0

$node*(1) set Z* 0.0

$node*(2) set X* 300.0

$node*(2) set Y* 300.0

$node*(2) set Z* 0.0

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 10.01 "puts \"end simulation\" ; $ns halt"

Graph procedure..

$ns at 1.0 "Graph"

set g [open graph.tr w]

proc Graph {} {

global ns g

set time 1.0

set now [$ns now]

puts $g "[expr rand()*8] [expr rand()*6]"

$ns at [expr $now+$time] "Graph"

}

Stop proceture

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec xgraph -M -bb -geometry 700X800 graph.tr &

exec nam Graph1.nam &

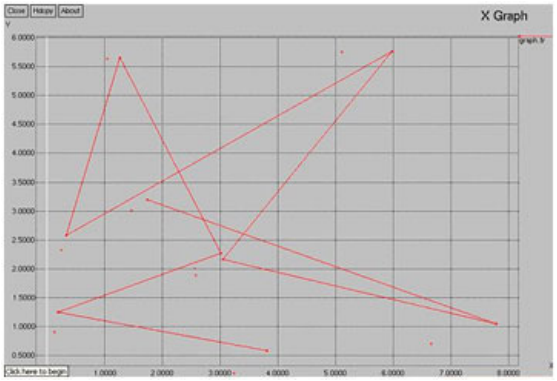
exit 0

}

$ns run

Procedure to run the program in the terminal window: $ns program 5.25.tcl

Output 5.25:



Program 5.26: Creation of graphs with two parameters as inputs by using AODV routing protocol TCL script

Program Description

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given and is followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the Z coordinates are zero. Graph is randomly generated by using the X and Y dimensions and is programmed to generate the trace file accordingly. The trace file acts as an input file to plot the graph in the format of trace file. Here, a single plotted graph consists of two trace file values. Different colors are given to each trace file during plotting. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name: program5.26.tcl

Define options

```
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 nodes
set val(rp)     AODV;                       # routing protocol
set val(x)      500;                        # X dimension of topography
set val(y)      400;                        # Y dimension of topography
set val(stop)   10.0;                       # time of simulation end
```

Creating simulation

```
set ns [new Simulator]
```

Creating nam and trace file

```
set tracefd     [open Graph2.tr w]
set namtrace    [open Graph2.nam w]
$ns trace-all $tracefd
```

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set up topography object

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

configure the nodes

```
$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) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace ON
```

Creating node objects..

```
for {set i 0} {$i< $val(nn) } { incri } {
        set node_($i) [$ns node]
        }
        for {set i 0} {$i< $val(nn) } {incri } {
        $node_($i) color black
        $ns at 0.0 "$node_($i) color black"
    }
```

Provide initial location of mobilenodes

$node*(0) set X* 50.0

$node*(0) set Y* 50.0

$node*(0) set Z* 0.0

$node*(1) set X* 200.0

$node*(1) set Y* 250.0

$node*(1) set Z* 0.0

$node*(2) set X* 300.0

$node*(2) set Y* 300.0

$node*(2) set Z* 0.0

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 10.01 "puts \"end simulation\" ; $ns halt"

Graph procedure..

procedure..

$ns at 1.0 "Graph"

set g [open graph.tr w]

set g1 [open graph1.tr w]

proc Graph {} {

global ns g g1

set time 1.0

set now [$ns now]

puts $g "[expr rand()*8] [expr rand()*6]"

puts $g1 "[expr rand()*8] [expr rand()*6]"

$ns at [expr $now+$time] "Graph"

}

stop procedure:

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec xgraph -P -bb -geometry 700X800 graph.tr graph1.tr &

exec nam Graph2.nam &

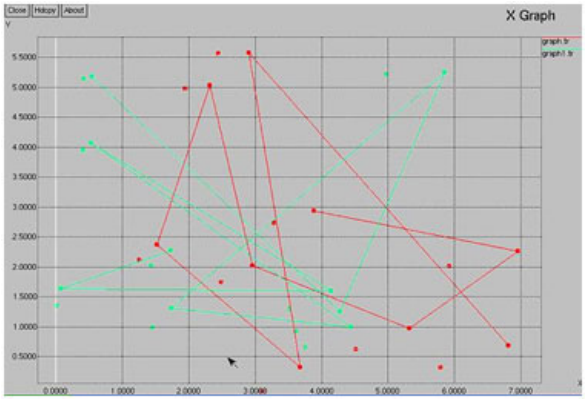
exit 0

}

$ns run

Procedure to run the program in the terminal window: $ns program 5.26.tcl

Output 5.26:



Program 5.27: Creation of graphs with more than two parameter files as inputs by using AODV routing protocol TCL script

Program Description

Number of nodes in the network is static and is declared as three in the network. Procedure for the creation of nam file and trace file is given, followed by the topology creation. Localization of the network is static. X and Y coordinates values are given in the program and the value of Z coordinates are zero. Graph is randomly generated using the X and Y dimensions and is programmed to generate the trace file accordingly. The trace file acts as input file to plot the graph in the format of trace file. Here, a single plotted graph consists of more than two trace file values. Different colors are given to each trace file during plotting. Routing protocol is AODV and the stop time of the simulation is 10ms.

File Name: program 5.27.tcl

Define options

```
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 nodes
set val(rp)     AODV;                         # routing protocol
set val(x)      500;                          # X dimension of topography
set val(y)      400;                          # Y dimension of topography
set val(stop) 10.0;                           # time of simulation end
```

Creating simulation

```
set ns [new Simulator]
```

Creating nam and trace file

```
set tracefd       [open Graph3.tr w]

set namtrace      [open Graph3.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

set up topography object

```
set topo          [new Topography]

$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
```

configure the nodes

```
    $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) \
```

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF \

-movementTrace ON

Creating node objects..

for {set i 0} {$i< $val(nn) } { incri } {

set node_($i) [$ns node]

}

for {set i 0} {$i< $val(nn) } {incri } {

$node_($i) color black

$ns at 0.0 "$node_($i) color black"

}

Provide initial location of mobilenodes

$node*(0) set X* 50.0

$node*(0) set Y* 50.0

$node*(0) set Z* 0.0

$node*(1) set X* 200.0

$node*(1) set Y* 250.0

$node*(1) set Z* 0.0

$node*(2) set X* 300.0

$node*(2) set Y* 300.0

$node*(2) set Z* 0.0

Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

30 defines the node size for nam

$ns initial*node*pos $node_($i) 30

}

Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node_($i) reset";

}

ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 10.01 "puts \"end simulation\" ; $ns halt"

Graph procedure..

$ns at 1.0 "Graph"

set g [open graph.tr w]

set g1 [open graph1.tr w]

set g2 [open graph2.tr w]

proc Graph {} {

global ns g g1

set time 1.0

set now [$ns now]

puts $g "[expr rand()*8] [expr rand()*6]"

puts $g1 "[expr rand()*8] [expr rand()*6]"

puts $g2 "[expr rand()*8] [expr rand()*6]"

$ns at [expr $now+$time] "Graph"

}

Stop proceture

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

close $tracefd

close $namtrace

exec xgraph -M -bb -geometry 700X800 graph.tr graph1.tr graph2.tr &

exec nam Graph3.nam &

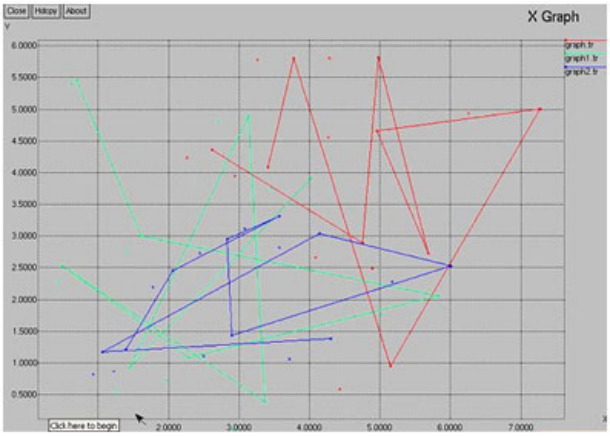
exit 0

}

$ns run

Procedure to run the program in the terminal window: $ns program 5.27.tcl

Output 5.27:


Close
Hdcpy
About
X Graph
Y
graph.tr
graph1.tr
graph2.tr
6.0000
5.5000
5.0000
4.5000
4.0000
3.5000
3.0000
2.5000
2.0000
1.5000
1.0000
0.5000
Click here to begin
2.0000
3.0000
4.0000
5.0000
6.0000
7.0000
X