



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

LAB MANUAL

Mobile Communications

Code: 16CS404

2019-20 II Semester

Faculty:

Deevi Radha Rani

Assistant Professor

Department of CSE

drd_cse@vignan.ac.in

Preface

About the Lab Course:

This course deals with TCL scripting language and a widely known NS2 event driven simulation tool for simulating wired as well as wireless network functions and protocols. It is useful in studying the dynamic nature of communication networks.

The objective of this course is to enable the students to

- ✓ Learn basic commands of TCL script
- ✓ Run basic programs written using TCL
- ✓ Simulates various protocols like TCP and UDP
- ✓ Demonstrates traffic models like CBR, VBR, FTP
- ✓ Parsing of tracefile for knowing the network traffic behavior
- ✓ Simulating DV and LS routing protocol.
- ✓ Simulating various protocols like DSR, DSDV, AODV

Index

S.No.	Content	Page No.
1	Vision, Mission, POs, PEOs and PSOs	4
2	Lab Evaluation Guidelines	6
3	List of Experiments	10
4	Course Outcomes	10
5	CO PO Mapping	10
6	Lab Plan	12
7	Experiments	12
	1. Simple TCL Scripts	12-22
	2. Installation of NS2	23-27
	3. a. Create FTP traffic over TCP using NS2 b. Create CBR traffic over UDP using NS2	28-35
	4. a. Write TCL script for creating nodes, duplex link, orientation, Label and Queue b. Write TCL script to create TCP agent, TCP sink and attach the TCO agent with TCP sink.	36-39
	5. Write TCL script to set identification Color to links.	40-41
	6. Simulate Link State Routing (LS) protocol in NS2?	42-45
	7. Simulate Distance Vector Routing (DV) protocol in NS2?	46-49
	8. Develop TCL script to make TCP communication between nodes using DSR routing protocol.	50-53
	9. Write TCL script to make communication between nodes using AODV routing protocol and CBR traffic.	54-59
	10. Develop and implement TCL script to make TCP communication between nodes using DSDV routing protocol	60-63

Vision, Mission, POs, PEOs and PSOs

Vision:

To evolve as a center of high reputation in Computer Science & Engineering and create computer software professionals trained on problem solving skills imbued with ethics to serve the ever evolving and emerging requirements of IT Industry and society at large.

Mission:

- Imparting quality education through well designed curriculum, innovative teaching and learning methodologies integrated with professional skill development activities to meet the challenges in the career.
- Nurture research and consultancy activities amongst students and faculty by providing State-of-art facilities and Industry-Institute Interaction.
- Developing capacity to learn new technologies and apply to solve social and industrial problems to become an entrepreneur.

Programme Educational Objectives (PEOs):

PEO 1: Pursue successful professional career in IT and IT-enabled industries.

PEO 2: Pursue lifelong learning in generating innovative engineering solutions using research and complex problem-solving skills.

PEO 3: Demonstrate professionalism, ethics, inter-personal skills and continuous learning to develop leadership qualities.

Programme Specific Outcomes (PSOs):

PSO 1: Application Development Skills: Design and development of web applications using various technologies such as HTML, JSP, PHP, ASP and ASP.NET to cater the needs of the society

PSO 2: Enrich Research Skills: Offer solutions which impact geo-socio-economic and environmental scenario by using Machine Learning, Artificial Intelligence and IoT.

Programme Outcomes (POs):

Program Outcomes (POs), are attributes acquired by the student at the time of graduation. The POs given in below, ensure that the POs are aligned to the Graduate Attributes (GAs) specified by National Board of Accreditation (NBA). These attributes are measured at the time of Graduation.

PO 1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental

considerations.

PO 4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO 6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Lab Evaluation Guidelines

I. Guidelines for Lab teacher :

A. Before commencement of lab class work :

1. The detailed information consisting of experiments, batch formations, experiment schedules, etc., will be displayed / informed to the students one week before the commencement of the semester so that the student comes prepared for the session.
2. Before commencement of laboratory classes the teachers and instructor handling the courses should practice the experiments.
3. Copies of the lab manual (soft copy) will be made available to the students along with the schedule. The lab manual will consist of the list of equipment, detailed procedure to conduct the experiment, format for record writing, algorithm or procedure, circuit diagrams, list of equipment's/tools needed, sample calculations, sample graphs, outcomes for each experiment and possible set of short questions to help student's gain critical understanding. Three hardcopies are to be made available during the lab sessions one is master copy and the other two copies are for student's ready reference during lab sessions.

B. During Laboratory Sessions :

1. Conduct a demo session of all the experiments in that cycle to students. Explain them about other guidelines to be followed in the lab.
2. A log book is to be maintained by the instructor in which student will sign every time when he visits the lab to do the experiment.
3. Verify the student record before allowing him to perform experiment. Previous experiment completion and preparedness for performing the experiment scheduled on that day has to be checked.
4. Evaluate the performance as per the table given below. The set of parameters may slightly differ from one laboratory to the other, and will be announced before the commencement of the lab session. These parameters are to be assessed for each laboratory session. The lab teacher has to use red pen to give evaluation results on record.

Continuous Evaluation of Laboratory

S.No	Component	Marks
1	Preparedness	2M
2	Coding	4M
3	Testing	2M
4	Viva	2M
Total		10M

5. These marks are to be marked in record, component wise for each experiment and total marks are to be entered into section wise lab sheet. This assessment is to be carried out for each practical session and the average mark of all the sessions is to be considered and finally scaled down to 30 marks. An internal laboratory examination will be conducted for another 20 marks.

Absent students / not performed students:

Students are allowed to do the experiments after class hrs.

Internal Laboratory Examination:

The internal laboratory examination shall be conducted around the middle of the semester. The examination is to be conducted, by a team of two examiners, one who conducts the laboratory sessions and the other appointed by the Deputy HoD. The scheme of evaluation is given below.

Component	Marks		
	Internal Examiner (Lab Teacher)	External Laboratory Examiner (Appointed by Deputy HoD)	Total
Objective & Procedure write up including outcomes	5	5	10
Coding	10	10	20
Testing	5	5	10
Viva voce	0	10	10
Total Marks	20	30	50

The teacher shall make a comparative statement of these lab marks with the student's previous aggregate percentage of marks. When the difference is more than 20%, the remarks of the teacher are to be endorsed with full justification and displayed at notice boards. A committee appointed by the Dean Evaluation will scrutinize all such comparisons and take necessary action.

C. End – semester Laboratory Evaluation :

End semester examination for each practical course is conducted jointly by both internal and external examiners. The examiners are appointed by Dean, Evaluation from the panel of examiners suggested by the respective Heads of the Department. To maintain the objectivity and seriousness of the students towards the lab curriculum and lab examinations, a panel of large number of examiners, four times to the actual requirement shall be suggested by the HoD, at least one month in advance and submit the details to the Dean Evaluation.

The Dean Evaluation will select the examiners on a random basis. After examination, at least 10% of the scripts are to be audited by a three member committee appointed by the Dean- Evaluation. For every class, the mean of the lab marks and theory marks are to be compared and if there is a deviation of more than 20% the reasons are to be analyzed and documented. The scheme of evaluation may vary depending on the nature of laboratory, which shall be shared with the student by the laboratory in-charge and also stamped on the answer scripts. The general scheme of evaluation is given in Table below.

End Semester Evaluation Pattern of Labs

Component	Marks		
	Internal Examiner	External Laboratory Examiner	Total
Objective & Procedure write up including outcomes	5	5	10
Coding	5	5	10
Testing	10	10	20
Viva voce	0	10	10
Total Marks	20	30	50

II. Guidelines to lab instructor / programmers :

A) Before commencement of semester :

1. Along with lab teacher do all the experiments offered for students in that laboratory.
2. Familiarize yourself about purpose of the experiment, outcomes, result analysis, probable sources of errors in readings etc...
3. Collect the following information
 - a) List of experiments
 - b) Instructions related to that lab
 - c) Lab manual – 3 hard copies and one softcopy
 - d) Sample lab record – 3 hard copies
 - e) Log book / File
 - f) Format of stamp to be printed on record sheets
4. Make the stamp and stamp pad ready for experiments. Before commencement of lab session check the stamp and stamp pad ready for the use by lab teachers.

B) One or two days before the lab session :

1. Check the functioning of systems, instruments and take action if found defective.
2. Maintain cleanness in the laboratory

C) During lab session :

1. Allow students having appropriate dress code and ID cards into lab.
2. Ask the student to make entry into the log book. Verify with reference to the batch wise planning of experiments. Deviations if any inform to faculty.
3. Make rounds in the laboratory, help the students to perform the experiment and clarify their doubts.
4. While helping the students, please remember that the students should be able to do the experiment on their own.
5. After class work, clean the system and make the lab ready for next session.

D) During examination :

1. Obtain the student wise experiment schedules from the lab teacher and make the lab ready for examination.
2. Collect necessary stationary from the examination section.
3. Follow the instructions given by lab teacher during examinations.
4. After evaluation, return the examination records to examination section.

E) Records to be maintained by lab instructor :

1. Log sheet for each lab

List of Experiments

1.
 - a. Write TCL Script for checking Age range.
 - b. Write TCL Script for performing arithmetic operations using switch
 - c. Write TCL Script for checking whether given number is prime or not
 - d. Write TCL Script to find factorial of a given number
 - e. Write TCL Script to swap two numbers.
 - f. Write TCL Script to find area of a rectangle.
 - g. Write TCL Script to check whether given number is Armstrong or not.
 - h. Write TCL Script to find the sum of digits.
 - i. Write TCL Script to print Fibonacci Series.
 - j. Write TCL Script to check number is palindrome or not.
2. Installation of NS2
3.
 - a. Create FTP traffic over TCP using NS2
 - b. Create CBR traffic over UDP using NS2
4.
 - a. Write TCL script for creating nodes, duplex link, orientation, Label and Queue
 - b. Write TCL script to create TCP agent, TCP sink and attach the TCO agent with TCP sink.
5. Write TCL script to set identification Color to links.
6. Simulate Link State Routing (LS) protocol in NS2?
7. Simulate Distance Vector Routing (DV) protocol in NS2?
8. Develop TCL script to make TCP communication between nodes using DSR routing protocol.
9. Write TCL script to make communication between nodes using AODV routing protocol and CBR traffic.
10. Develop and implement TCL script to make TCP communication between nodes using DSDV routing protocol.

Course Outcomes:

CO1: Understand the usage of TCL script for NS2 simulation

CO2: Analyze routing algorithms and their performance

CO3: Simulate various network topologies with different protocols

CO4: Design of a user defined network and analyze its performance by interpreting the traffic parameters

CO-PO Mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1											
CO2		3										
CO3			3									
CO4				3								

Lab Plan

S.No.	Date (Monday)	Exp. No.	Experiment Details
1	16-12-19	1	Introduction to TCL: Operators, commands, control statements
2	23-12-19	1	a. Write TCL Script for checking Age range. b. Write TCL Script for performing arithmetic operations using switch c. Write TCL Script for checking whether given number is prime or not d. Write TCL Script to find factorial of a given number e. Write TCL Script to swap two numbers.
3	30-12-19	1	f. Write TCL Script to find area of a rectangle. g. Write TCL Script to check whether given number is Armstrong or not. h. Write TCL Script to find the sum of digits. i. Write TCL Script to print Fibonacci Series. j. Write TCL Script to check number is palindrome or not.
4	6-1-20	2	Introduction of NS2 and Installation
6	27-1-20	3	a. Create FTP traffic over TCP using NS2 b. Create CBR traffic over UDP using NS2
7	3-2-20	4	a. Write TCL script for creating nodes, duplex link, orientation, Label and Queue b. Write TCL script to create TCP agent, TCP sink and attach the TCP agent with TCP sink.
8	10-2-20	5	Write TCL script to set identification Color to links.
9	17-2-20	6	Simulate Link State Routing (LS) protocol in NS2?
10	24-2-20	***	Internal Lab Examination
11	2-3-20	7	Simulate Distance Vector Routing (DV) protocol in NS2?
12	16-3-20	8	Develop TCL script to make TCP communication between nodes using DSR routing protocol.
13	23-6-20	9	Write TCL script to make communication between nodes using AODV routing protocol and CBR traffic.
14	30-3-20	10	Develop and implement TCL script to make TCP communication between nodes using DSDV routing protocol.

Experiments

1. a. Write TCL Script for checking Age range.

Algorithm:

1. Read value for age
2. If age>=0 && age<=12 Print "You are a child."
3. If age>= 13 && age <= 19 Print "You are a teen."
4. If age>19 Print "You are an adult now."

Program:

```
puts "Enter Age"
gets stdin Age
if {$Age >= 0 && $Age <= 12} {
    puts "You are a child."
} elseif {$Age >= 13 && $Age <= 19} {
    puts "You are a teen."
} elseif {$Age > 19} {
    puts "You are an adult now."
}
```

Test Cases:

Enter Age 11
You are a child.

Enter Age 20
You are an adult now.

Enter Age 14
You are a teen.

1. b. Write TCL Script for performing arithmetic operations using switch

Algorithm:

1. Read 2 numbers
2. Read a variable ch
3. Perform the Arithmetic Operation according to input value of ch (1 for addition, 2 for subtraction, 3 for multiplication, 4 for division, 5 for modulus) using switch statement
4. Print the value
5. Check whether to continue or not

Program:

```
set i 1
while {$i==1} {
puts "Enter first number"
gets stdin a
puts "Enter second number"
gets stdin b
puts "Enter 1 for addition \n    2 for subtraction \n    3
for multiplication \n    4 for divison \n    5 for modulus\n"
gets stdin ch
switch $ch {
    1 {
        set c [expr $a + $b]
        puts "Addition of $a and $b is $c"
    }
    2 {
        set d [expr $a - $b]
        puts "\Subtraction of $a and $b is $d"
    }
    3 {
        set e [expr $a * $b]
        puts "Multiplication of $a and $b is $e"
    }
    4 {
        set f [expr $a / $b]
        puts "Divison of $a and $b is $f"
    }
    5 {
        set g [expr $a % $b]
        puts "Modulus of $a and $b is $g"
    }

    default {
        puts "Enter correct option"
    }
}
```

```
}  
puts "Do u want continue or not(Y/N) "  
gets stdin con  
if {$con=="n"} {  
    exit  
}  
}
```

Test Cases:

Enter first number: 3
Enter second number: 4
Enter 1 for addition
2 for subtraction
3 for multiplication
4 for division
5 for modulus
1
Addition of 3 and 4 is 7
Do u want continue or not(Y/N)
Y

Enter first number: 3
Enter second number: 4
Enter 1 for addition
2 for subtraction
3 for multiplication
4 for division
5 for modulus
3
Multiplication of 3 and 4 is 12
Do u want continue or not(Y/N)
N

1. c. Write TCL Script for checking whether given number is prime or not

Algorithm:

1. Initialize variables
 flag←1
 i←2
2. Read n from user.
3. Repeat the steps until $i < (n/2)$
 If remainder of $n \div i$ equals 0
 flag←0
 Go to step 6
 i←i+1
4. If flag=0
 Display n is not prime
else
 Display n is prime

Program:

```
puts "Enter Number:"
gets stdin num
set c 0
for { set i 2 } { $i < $num } { incr i } {
if [ expr $num % $i==0 ] {
incr c
}
}
if [ expr $c==0 ] {
puts " $num is prime "
} else {
puts " $num is not prime"
}
```

Test Cases:

Enter Number: 5
5 is prime

Enter Number: 10
10 is not prime

1. d. Write TCL Script to find factorial of a given number

Algorithm:

1. Initialize variables
 factorial←1
 i←1
2. Read value of n
3. Repeat the steps until i=n
 factorial←factorial*i i←i+1
4. Display factorial

Program:

```
puts "Enter a number to find factorial"
gets stdin n
set i 1
set f 1
while { $i <= $n } {
    set f [ expr $f*$i ]
    incr i
}
puts "Factorial of $n is $f"
```

Test Cases:

Enter a number to find factorial 5
Factorial of 5 is 120

Enter a number to find factorial 10
Factorial of 10 is 3628800

1. e. Write TCL Script to swap two numbers.

Algorithm:

1. Read 2 numbers a, b
2. Set t=0
3. Perform t=a
4. Perform a=b
5. Perform b=t
6. Print a b

Program:

```
set x 6
set y 2
set temp 0
puts "Value of x and y are $x $y"
puts "After swapping"
set temp $x
set x $y
set y $temp
puts "Value of x and y are $x $y"
```

Test Cases:

Value of a and b are 6 2
After swapping
Value of a and b are 2 6

Value of a and b are 122 123
After swapping
Value of a and b are 123 122

Viva Questions:

1. Define TCL
2. How to run TCL file?
3. Give the syntax for reading a value to a variable
4. Give the syntax for switch statement
5. How to evaluate an expression in TCL?

1. f. Write TCL Script to find area of a rectangle.

Algorithm:

1. Read length and breadth
2. Calculate area = length * breadth
3. Print area

Program:

```
puts "Area of a rectangle"
puts "Enter length:"
gets stdin l
puts "Enter breadth:"
gets stdin b
set c [expr $l * $b]
puts "Area of rectangle is $c"
```

Test Cases:

Enter length: 3
Enter breadth: 2
Area of rectangle is 6

Enter length: 10
Enter breadth: 2
Area of rectangle is 20

1. g. Write TCL Script to check whether given number is Armstrong or not.

Algorithm:

1. Read number
2. Set sum=0 and duplicate=number
3. Reminder=number%10
4. Sum=sum+(reminder*reminder*reminder)
5. Number=number/10
6. Repeat steps 4 to 6 until number > 0
7. if sum = duplicate
8. Display number is armstrong
9. else Display number is not Armstrong

Program:

```
puts "Enter number:"
gets stdin n
set d 0
set e 0
set f 0
set c $n
while {$c!=0} {
    set e [expr $c%10]
    set d [expr $e*$e*$e]
    set f [expr $f + $d]
    set c [expr $c/10]
}
if {$f == $n} {
    puts "Number is Armstrong"
} else {
    puts "Number is not Armstrong"
}
```

Test Cases:

Enter number: 123
Number is not Armstrong

Enter number:153
Number is Armstrong

1. h. Write TCL Script to find the sum of digits.

Algorithm:

1. Read a Number
2. Initialize Sum to zero
3. While Number is not zero
 Get Remainder by Number Mod 10
 Add Remainder to Sum
 Divide Number by 10
4. Print sum

Program:

```
puts "Enter number:"
gets stdin n
set d 0
set e 0
set c $n
while {$c!=0} {
    set e [expr $c%10]
    set d [expr $d + $e]
    set c [expr $c/10]
}
puts "Sum of the digits is $d"
```

Test Cases:

Enter number: 1234
Sum of the digits is 10

Enter number: 1111
Sum of the digits is 5

1. i. Write TCL Script to print Fibonacci Series.

Algorithm:

1. Initialize variables first_term←0 second_term←1
2. Read range
3. Display first_term and second_term
4. Repeat the steps until second_term=range
 - temp←second_term
 - second_term←second_term+first term
 - first_term←temp
 - Display second_term

Program:

```
puts "Enter the range"
gets stdin n
set i 0
set j 1
set k 0
set count 2
puts $i
puts $j
while {$count < $n} {
    set k [expr $i+$j]
    set i $j
    set j $k
    puts $j
    set count [expr $count+1]
}
puts "Fibonacci num is $k"
```

Test Cases:

Enter the range: 5
Fibonacci num is 1
Fibonacci num is 1
Fibonacci num is 2
Fibonacci num is 3
Fibonacci num is 5
Enter the range: 7
Fibonacci num is 1
Fibonacci num is 1
Fibonacci num is 2
Fibonacci num is 3
Fibonacci num is 5
Fibonacci num is 8
Fibonacci num is 13

1. j. Write TCL Script to check number is palindrome or not.

Algorithm:

1. Read n
2. Set s = 0, a=n
3. while (n>0)
4. begin
5. rem=n%10
6. s=s*10+rem
7. n=n/10
8. end
9. if(s==a) print "Number is palindrome"
10. else print "Number is not palindrome"

Program:

```
puts "Enter number:"
gets stdin n
set d 0
set f 0
set c $n
while {$c!=0} {
    set e [expr $c%10]
    set d [expr $f * 10]
    set f [expr $d + $e]
    set c [expr $c / 10]
}
if {$f == $n} {
    puts "Number is palindrome"
} else {
    puts "Number is not palindrome"
}
```

Test Cases:

Enter number: 121
Number is palindrome

Enter number: 123
Number is not palindrome

Viva Questions:

1. Define TCL
2. How to run TCL file?
3. Give the syntax for reading a value to a variable
4. Give the syntax for switch statement
5. How to evaluate an expression in TCL?

2. Installation of NS2

1 Introduction

Network simulators are tools used to simulate discrete events in a network and which helps to predict the behaviours of a computer network. Generally the simulated networks have entities like links, switches, hubs, applications, etc. Once the simulation model is complete, it is executed to analyse the performance. Administrators can then customize the simulator to suit their needs. Network simulators typically come with support for the most popular protocols and networks in use today, such as WLAN,UDP,TCP,IP, WAN, etc.

Most simulators that are available today are based on a GUI application like the NCTUNS while some others incl. NS2 are CLI based. Simulating the network involves configuring the state elements like links, switches, hubs, terminals, etc. and also the events like packet drop rate, delivery status and so on. The most important output of the simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots. Most of the simulation is performed in discrete time intervals where events that are in the queue are processed one after the other in an order.

Since simulation is a complex task, we cannot guarantee that all the simulators can provide exact or accurate results for all the different type of information. Examples of network simulators are: ns, NCTUNS, NetSim, etc.

ns2 is a name for series of discrete event network simulators like ns-1, ns-2 and ns-3. All of them are discrete-event network simulators, primarily used in research and teaching. ns2 is free software, publicly available under the GNU GPLv2 license for research, development, and use.

This post deals with the installation of "ns2" also called the "network simulator 2" in Ubuntu 14.04.

2 Download and Extract ns2

Download the all in one package for ns2 from [here](#)

The package downloaded will be named "ns-allinone-2.35.tar.gz". Copy it to the home folder. Then in a terminal use the following two commands to extract the contents of the package.:

```
cd ~/
tar -xvzf ns-allinone-2.35.tar.gz
```

All the files will be extracted into a folder called "ns-allinone-2.35".

3 Building the dependencies

Ns2 requires a few packages to be pre installed. It also requires the GCC- version 4.3 to work correctly. So install all of them by using the following command:

```
sudo apt-get install build-essential autoconf automake libxmu-dev
```

One of the dependencies mentioned is the compiler GCC-4.3, which is no longer available, and thus we have to install GCC-4.4 version. The version 4.4 is the oldest we can get. To do that, use the following command:

```
sudo apt-get install gcc-4.4
```

The image below shows the output of executing both the above commands. If you have all the dependencies pre-installed, as I did, the output will look like the image below:

Once the installation is over , we have to make a change in the "ls.h" file. Use the following steps to make the changes:

```
akshay@akshay-UBPC:~$ sudo apt-get install build-essential autoconf automake libxmu-dev
[sudo] password for akshay:
Reading package lists... Done
Building dependency tree
Reading state information... Done
autoconf is already the newest version.
automake is already the newest version.
build-essential is already the newest version.
libxmu-dev is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
akshay@akshay-UBPC:~$ sudo apt-get install gcc-4.4
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc-4.4 is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
akshay@akshay-UBPC:~$
```

Navigate to the folder "linkstate", use the following command. Here it is assumed that the ns folder extracted is in the home folder of your system.

```
cd ~/ns-allinone-2.35/ns-2.35/linkstate
```

Now open the file named "ls.h" and scroll to the 137th line. In that change the word "**error**" to "**this->error**". The image below shows the line 137 (highlighted in the image below) after making the changes to the ls.h file. To open the file use the following command:

```
gedit ls.h
```

```
root@akshay-UBPC:/home/akshay/ns-allinone-2.35/ns-2.35/linkstate# cd ns-2.35/
root@akshay-UBPC:/home/akshay/ns-allinone-2.35/ns-2.35# ls
adc          bitmap      COPYRIGHTS  gaf          Makefile    plm          satellite   validate
allinone     CHANGES.html dccp       gen          Makefile.in puma        sctp       validate.out
aodv         classifier  delaybox   HOWTO-CONTRIBUTE makefile.vc pushback    sensor-nets VERSION
aomdv        common      diffserv   imcp         mcast       src_rtg     webcache
apps         conf        diffusion   indep-utils  mdart       queue       tcp         wpan
asin         config.guess diffusion3   install-sh   mobile      rap         test-all  xcp
autoconf.h   config.h    doc        INSTALL.WIN32 mpls        README      tora
autoconf.h.in config.log  dsr        lib          nix         realaudio   tmix
autoconf-win32.h config.status dsr        LICENSES     ns.l        release.steps.txt TODO.html
BASE-VERSION config.sub  empweb     link         ns_telsh.cc routealgo   tools
baytcp       configure  emulate    linkstate    packmime    routing     trace
bin          configure.in FILES       mac          pgm         rtproto

root@akshay-UBPC:/home/akshay/ns-allinone-2.35/ns-2.35# cd linkstate/
root@akshay-UBPC:/home/akshay/ns-allinone-2.35/ns-2.35/linkstate# gedit ls.h
```


Save that file and close it.

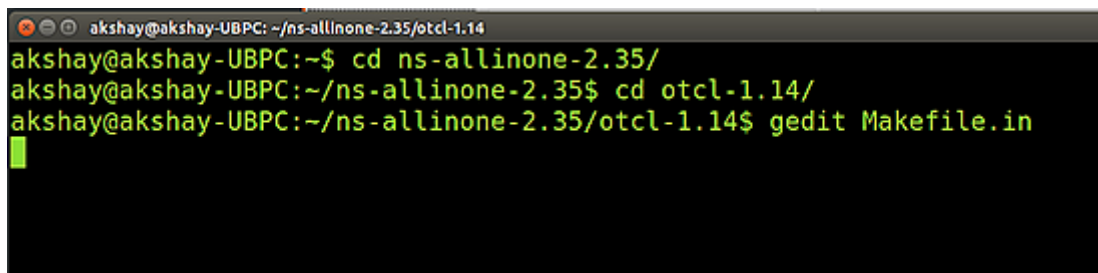
```
// this next typedef of iterator seems extraneous but is required by gcc-2.96
typedef typename map<Key, T, less<Key> >::iterator iterator;
typedef pair<iterator, bool> pair_iterator_bool;
iterator insert(const Key & key, const T & item) {
    typename baseMap::value_type v(key, item);
    pair_iterator_bool ib = baseMap::insert(v);
    return ib.second ? ib.first : baseMap::end();
}

void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
T* findPtr(Key key) {
    iterator it = baseMap::find(key);
    return (it == baseMap::end()) ? (T *)NULL : &((*it).second);
}
```

Now

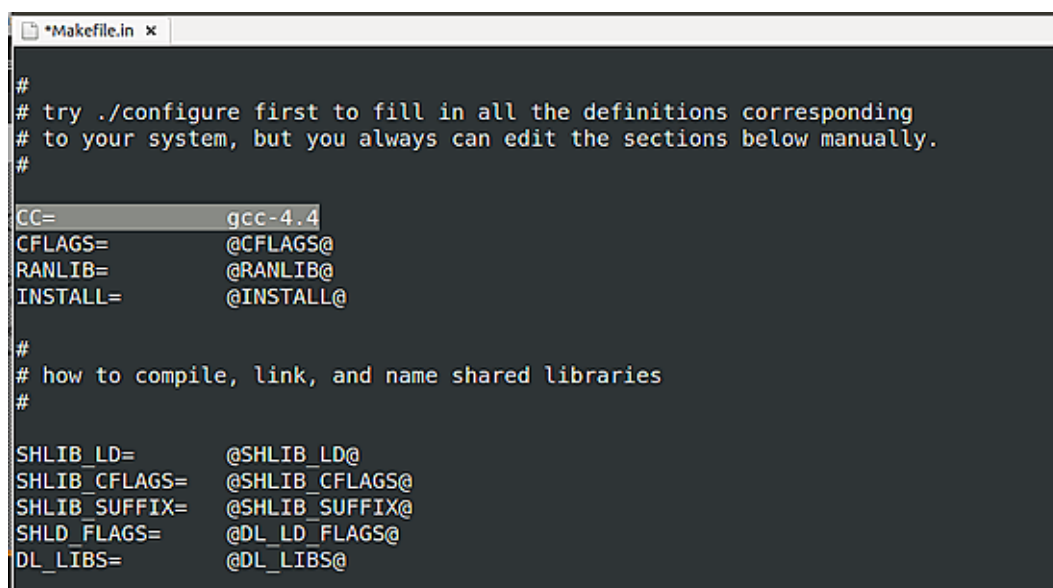
there is one more step that has to be done. We have to tell the ns which version of GCC will be used. To do so, go to your ns folder and type the following command:

Sudo gedit ns-allinone-2.34/otcl-1.13/Makefile.in



```
akshay@akshay-UBPC: ~/ns-allinone-2.35/otcl-1.14
akshay@akshay-UBPC:~/ns-allinone-2.35$ cd otcl-1.14/
akshay@akshay-UBPC:~/ns-allinone-2.35/otcl-1.14$ gedit Makefile.in
```

In the file, change Change CC= @CC@ to CC=gcc-4.4, as shown in the image below.



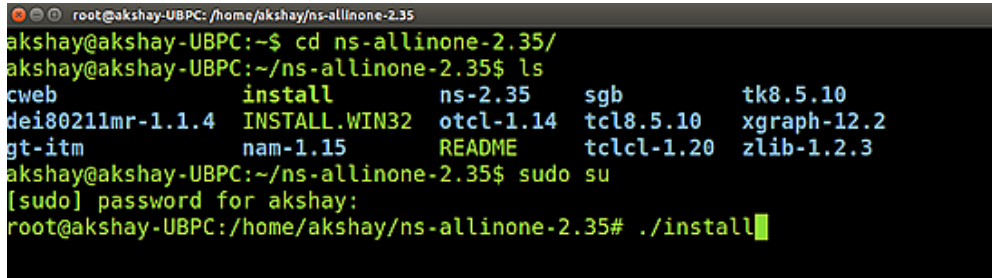
```
*Makefile.in x
#
# try ./configure first to fill in all the definitions corresponding
# to your system, but you always can edit the sections below manually.
#
CC=      gcc-4.4
CFLAGS=  @CFLAGS@
RANLIB=  @RANLIB@
INSTALL= @INSTALL@
#
# how to compile, link, and name shared libraries
#
SHLIB_LD= @SHLIB_LD@
SHLIB_CFLAGS= @SHLIB_CFLAGS@
SHLIB_SUFFIX= @SHLIB_SUFFIX@
SHLD_FLAGS= @DL_LD_FLAGS@
DL_LIBS= @DL_LIBS@
```

4 Installation

Now we are ready to install ns2. To do so we first require root privileges and then we can run the install script. Use the following two commands:

```
sudo su cd ~/ns-allinone-2.35/./install
```

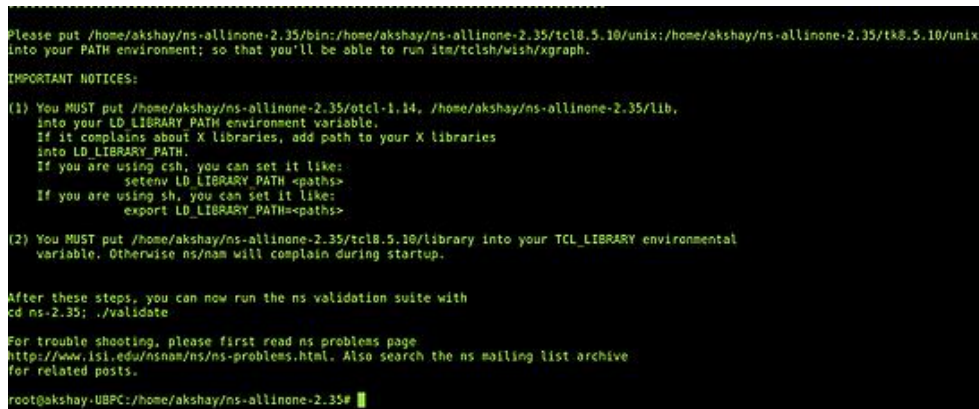
The following is a snap of these commands:



```
root@akshay-UBPC: /home/akshay/ns-allinone-2.35
akshay@akshay-UBPC:~$ cd ns-allinone-2.35/
akshay@akshay-UBPC:~/ns-allinone-2.35$ ls
cweb          install       ns-2.35      sgb          tk8.5.10
de180211mr-1.1.4  INSTALL.WIN32  otcl-1.14    tcl8.5.10    xgraph-12.2
gt-itm        nam-1.15      README      tclcl-1.20   zlib-1.2.3
akshay@akshay-UBPC:~/ns-allinone-2.35$ sudo su
[sudo] password for akshay:
root@akshay-UBPC:/home/akshay/ns-allinone-2.35# ./install
```

The image below shows how it looks upon successful execution

It took almost 6 minutes to build and install ns2 on my system. But before we run it, we need to add the build path to the environment path.



```
Please put /home/akshay/ns-allinone-2.35/bin:/home/akshay/ns-allinone-2.35/tcl8.5.10/unix:/home/akshay/ns-allinone-2.35/tk8.5.10/unix
into your PATH environment; so that you'll be able to run itm/tclsh/wish/xgraph.

IMPORTANT NOTICES:

(1) You MUST put /home/akshay/ns-allinone-2.35/otcl-1.14, /home/akshay/ns-allinone-2.35/lib,
into your LD_LIBRARY_PATH environment variable.
If it complains about X libraries, add path to your X libraries
into LD_LIBRARY_PATH.
If you are using csh, you can set it like:
    setenv LD_LIBRARY_PATH <paths>
If you are using sh, you can set it like:
    export LD_LIBRARY_PATH=<paths>

(2) You MUST put /home/akshay/ns-allinone-2.35/tcl8.5.10/library into your TCL_LIBRARY environmental
variable. Otherwise ns/nam will complain during startup.

After these steps, you can now run the ns validation suite with
cd ns-2.35; ./validate

For trouble shooting, please first read ns problems page
http://www.isi.edu/nsnam/ns/ns-problems.html. Also search the ns mailing list archive
for related posts.

root@akshay-UBPC:/home/akshay/ns-allinone-2.35#
```

5 Setting the Environment Path

The final step is to tell the system, where the files for ns2 are installed or present. To do that, we have to set the environment path using the ".bashrc" file. In that file, we need to add a few lines at the bottom. The things to be added are given below. But for the path indicated below, many of those lines have `"/home/akshay/ns-allinone-2.35/..."` , but that is where I have my extracted folder. Make sure you replace them with your path. For example, if you have installed it in a folder `"/home/abc"`, then replace `"/home/akshay/ns-allinone-2.35/otcl-1.14"` with `"/home/abc/ns-allinone-2.35/otcl-1.14"`.

Do this for all the required lines.

```
sudo gedit ~/.bashrc
```

Lines to be added:

LD_LIBRARY_PATH

OTCL_LIB=/home/akshay/ns-allinone-2.35/otcl-1.14

NS2_LIB=/home/akshay/ns-allinone-2.35/lib

X11_LIB=/usr/X11R6/lib

USR_LOCAL_LIB=/usr/local/lib

export

LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:\$OTCL_LIB:\$NS2_LIB:\$X11_LIB:\$USR_LOCAL_LIB

TCL_LIBRARY

TCL_LIB=/home/akshay/ns-allinone-2.35/tcl8.5.10/library

USR_LIB=/usr/lib

export TCL_LIBRARY=\$TCL_LIB:\$USR_LIB

PATH

XGRAPH=/home/akshay/ns-allinone-2.35/bin:/home/akshay/ns-allinone-2.35/tcl8.5.10/unix:/home/akshay/ns-allinone-2.35/tk8.5.10/unix

#the above two lines beginning from xgraph and ending with unix should come on the same line

NS=/home/akshay/ns-allinone-2.35/ns-2.35/

NAM=/home/akshay/ns-allinone-2.35/nam-1.15/

PATH=\$PATH:\$XGRAPH:\$NS:\$NAM

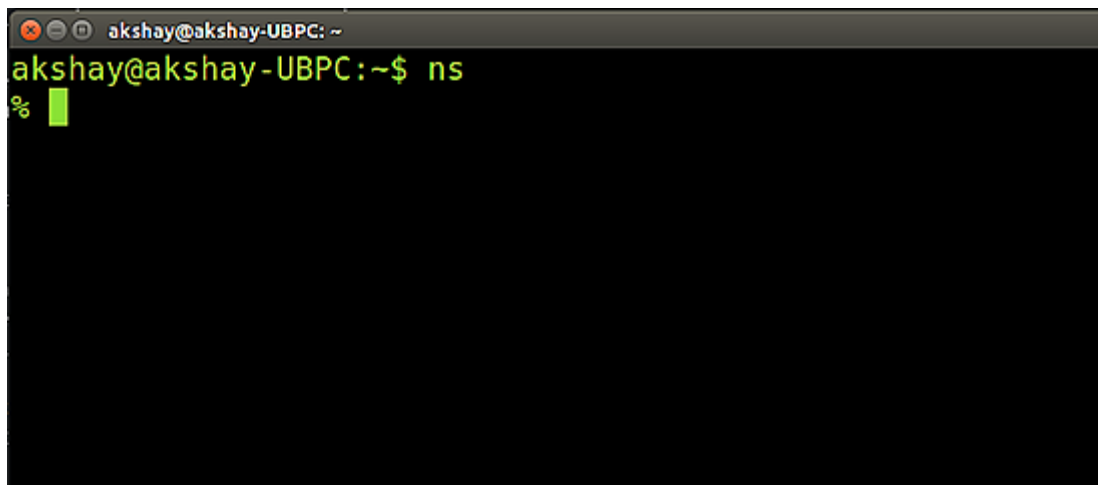
Once the changes have been made, save the file and restart the system.

6 Running ns2

Once the system has restarted, open a terminal and start ns2 by using the following command:

ns

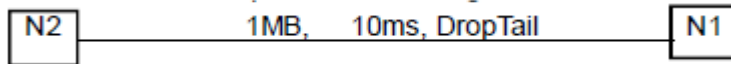
If the installation is correct then the terminal looks like the image below :



Viva Questions:

1. Abbreviate NS2
2. Give Linux command to install gcc.
3. In ls.h which line should be modified?
4. Which packages should be pre-installed?
5. Which command should be used to start ns2?

3. a. Create a TCL Script for the following Network



Create FTP traffic over TCP. Find out the throughput.

Procedure:

Open .ns file and create write as program as per the steps given below

1. Create 2 nodes
2. Open out.tr and out.name in write mode
3. Create duplex connection between 2 nodes with 1Mb 10ms DropTail
4. Create TCP agent and TCPSink
5. Attach TCP agent to node1 and TCPSink to node2
6. Create FTP traffic
7. Write finish procedure
8. Execute nam
9. Set start, stop and finish time

After executing .ns program trace file is generated in the same folder with the following format

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id
<pre>r : receive (at to_node) + : enqueue (at queue) src_addr : node.port (3.0) - : dequeue (at queue) dst_addr : node.port (0.0) d : drop (at queue)</pre>											
r	1.3556	3	2	ack	40	-----	1	3.0	0.0	15	201
+	1.3556	2	0	ack	40	-----	1	3.0	0.0	15	201
-	1.3556	2	0	ack	40	-----	1	3.0	0.0	15	201
r	1.35576	0	2	tcp	1000	-----	1	0.0	3.0	29	199
+	1.35576	2	3	tcp	1000	-----	1	0.0	3.0	29	199
d	1.35576	2	3	tcp	1000	-----	1	0.0	3.0	29	199
+	1.356	1	2	cbr	1000	-----	2	1.0	3.1	157	207
-	1.356	1	2	cbr	1000	-----	2	1.0	3.1	157	207

Take tracefile as input and write awk script to calculate the throughput of the network created.

Program:

```
set ns [new Simulator]
$ns rtproto DV
set node1 [$ns node]
set node2 [$ns node]
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
$ns duplex-link $node1 $node2 1Mb 10ms DropTail
set tcp2 [new Agent/TCP]
$ns attach-agent $node1 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $node2 $sink2
$ns connect $tcp2 $sink2
set traffic_ftp2 [new Application/FTP]
$traffic_ftp2 attach-agent $tcp2
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
$ns at 1.0 "$traffic_ftp2 start"
$ns at 3.0 "$traffic_ftp2 stop"
$ns at 5.0 "finish"
$ns run
```

Script.awk

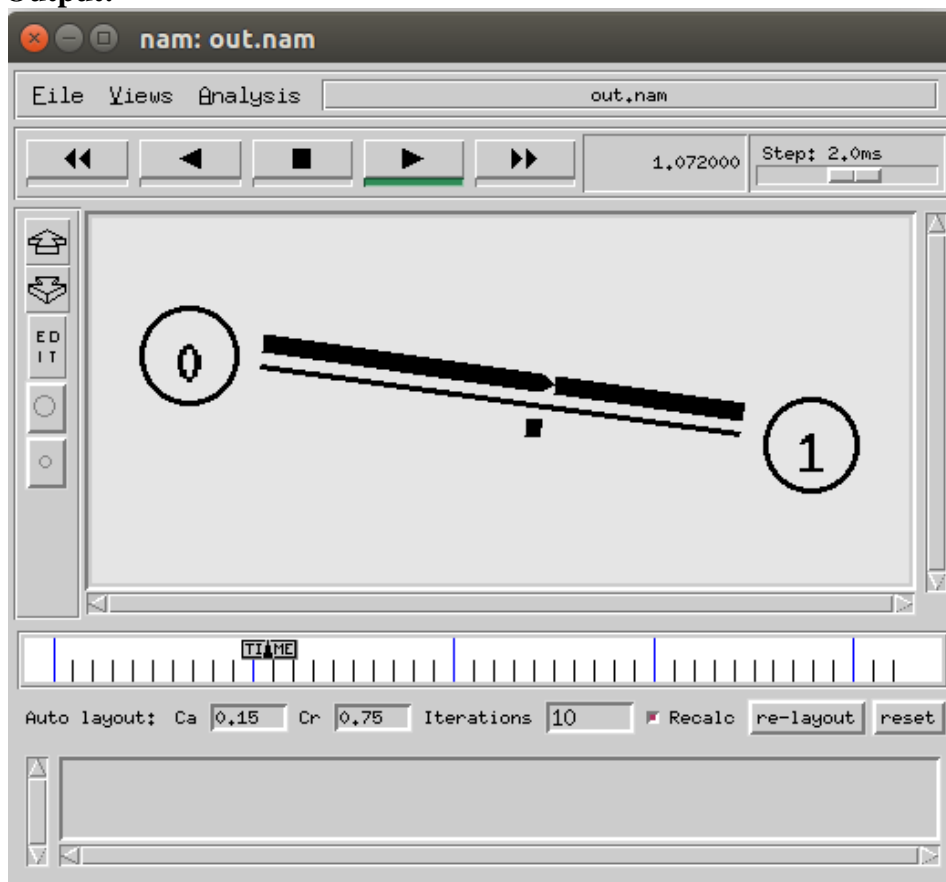
```
BEGIN{
    send=0;
    received=0;
    dropped=0;
    start=1.0;
    stop=3.0;
}
{
    if($1=="+/")
    {
        send++;
    }
    if($5=="tcp")
    {
        if($1=="r")
        {
            received++;
        }
    }
}
```

```

}}
if($1=="d")
{
dropped++;
}
}
END{
if(send=="0" && received == "0")
{
print "empty trace file\t"
}
print "no of packets received\t" received
print "throughput\t =" (received*8) / (stop-start) "bits per
second"
print "no of packets dropped\t" dropped
}

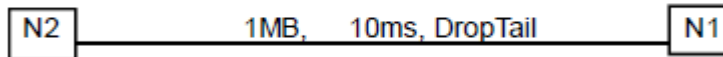
```

Output:



```
vignan-pc@vignan: ~/Desktop
vignan-pc@vignan:~$ cd Desktop/
vignan-pc@vignan:~/Desktop$ ns 12.tcl
vignan-pc@vignan:~/Desktop$ awk -f 12.awk out.tr
no of packets received 255
throughput =1020bits per second
no of packets dropped 0
vignan-pc@vignan:~/Desktop$
```


3. b. Create TCL Script for the following Network



Create CBR traffic over UDP. Find out the throughput.

Procedure:

Open .ns file and create write as program as per the steps given below

1. Create 2 nodes
2. Open out.tr and out.name in write mode
3. Create duplex connection between 2 nodes with 1Mb 10ms DropTail
4. Create TCP agent and LossMonitor
5. Attach TCP agent to node1 and LossMonitor to node2
6. Create CBR traffic
7. Write finish procedure
8. Execute nam
9. Set start, stop and finish time

After executing .ns program trace file is generated in the same folder with the following format

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id
<pre>r : receive (at to_node) + : enqueue (at queue) src_addr : node.port (3.0) - : dequeue (at queue) dst_addr : node.port (0.0) d : drop (at queue)</pre>											
r	1.3556	3	2	ack	40	-----	1	3.0	0.0	15	201
+	1.3556	2	0	ack	40	-----	1	3.0	0.0	15	201
-	1.3556	2	0	ack	40	-----	1	3.0	0.0	15	201
r	1.35576	0	2	tcp	1000	-----	1	0.0	3.0	29	199
+	1.35576	2	3	tcp	1000	-----	1	0.0	3.0	29	199
d	1.35576	2	3	tcp	1000	-----	1	0.0	3.0	29	199
+	1.356	1	2	cbr	1000	-----	2	1.0	3.1	157	207
-	1.356	1	2	cbr	1000	-----	2	1.0	3.1	157	207

Take tracefile as input and write awk script to calculate the throughput of the network created.

Program:

```
set ns [new Simulator]
$ns rtproto DV
set node1 [$ns node]
set node2 [$ns node]
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
$ns duplex-link $node1 $node2 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $node1 $udp0
set null0 [new Agent/LossMonitor]
$ns attach-agent $node2 $null0
$ns connect $udp0 $null0
set traffic_cbr0 [new Application/Traffic/CBR]
$traffic_cbr0 attach-agent $udp0
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
$ns at 1.0 "$traffic_cbr0 start"
$ns at 4.0 "$traffic_cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

script.awk

```
BEGIN{
    send=0;
    received=0;
    dropped=0;
    start=1.0;
    stop=3.0;
}
{
    if($1==" /+ /")
    {
        send++;
    }

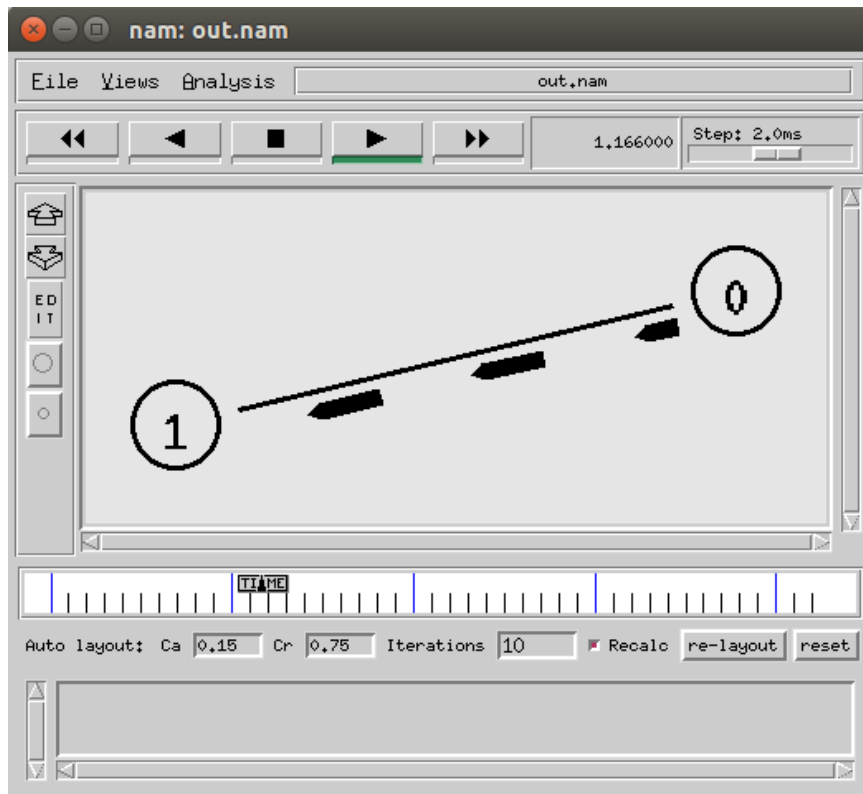
    if($5=="cbr")
    {
        if($1=="r")
        {
```

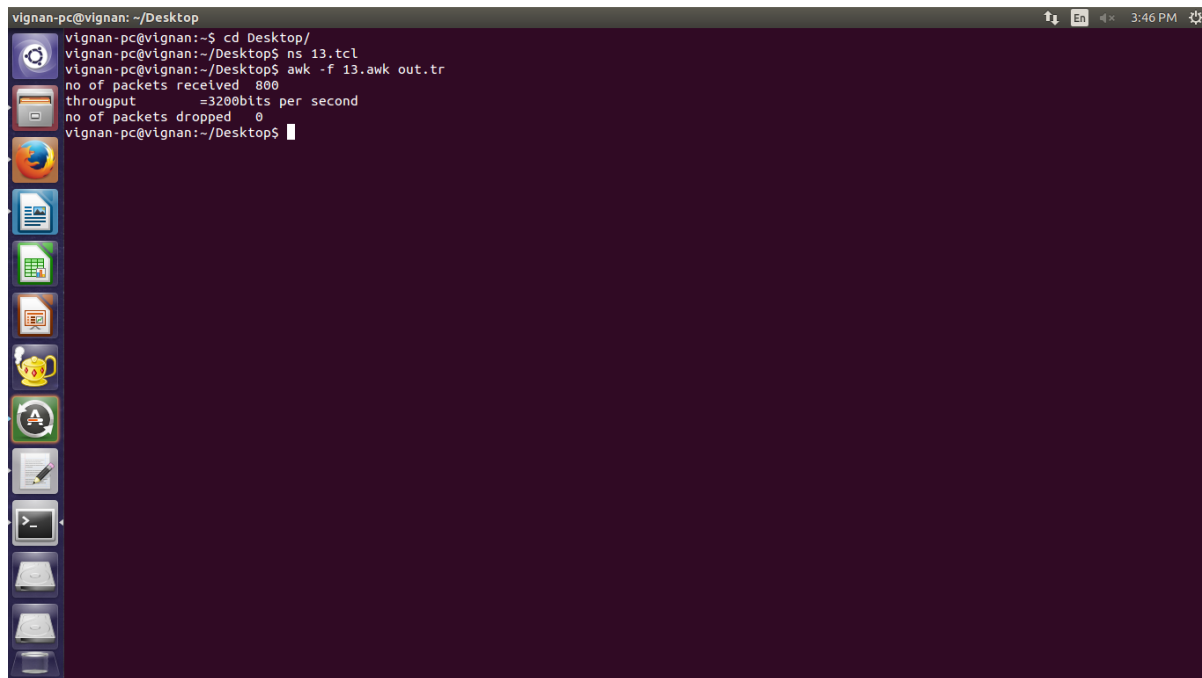
```

received++;
}}
if($1=="d")
{
dropped++;
}
}
END{
if(send=="0" && received == "0")
{
print "empty trace file\t"
}
print "no of packets received\t" received
print "throughput\t =" (received*8) /(stop-start) "bits per
second"
print "no of packets dropped\t" dropped
}

```

Output:



A terminal window titled 'vignan-pc@vignan: ~/Desktop' with a dark purple background. The window shows the execution of a network simulation. The user runs 'cd Desktop/' and then 'ns 13.tcl'. The output shows 'no of packets received 800', 'throughput =3200bits per second', and 'no of packets dropped 0'. The user then runs 'awk -f 13.awk out.tr'. The window has a standard Linux desktop environment with various application icons on the left sidebar and a top status bar showing 'En' and '3:46 PM'.

```
vignan-pc@vignan: ~/Desktop
vignan-pc@vignan:~$ cd Desktop/
vignan-pc@vignan:~/Desktop$ ns 13.tcl
vignan-pc@vignan:~/Desktop$ awk -f 13.awk out.tr
no of packets received 800
throughput =3200bits per second
no of packets dropped 0
vignan-pc@vignan:~/Desktop$
```

Viva Questions:

1. In which mode name and trace file be opened?
2. What is sink node?
3. Give the format of trace file.
4. How to find dropped packets?
5. Give the formula for throughput.

4. a. Write TCL script for creating nodes, duplex link, orientation, Label and Queue

Procedure:

1. Open out.tr and out.name file in write mode
2. Create atleast 3 nodes
3. Create a duplex link and droptail queue between 3 nodes
4. Color each node using the syntax:
Set n0 [\$ns node]
\$n0 color red
5. Label all 3 nodes using the syntax:
\$ns at 0.0 "\$n0 label Client"
6. Set orientation to the links using the syntax:
\$ns duplex-link-op \$node1 \$node2 orient right
(can use any of the orientations: left, right, up, down, left-up, left-down, right-up, right-down)
7. Set shape to the nodes using the syntax:
\$Endserver1 shape hexagon

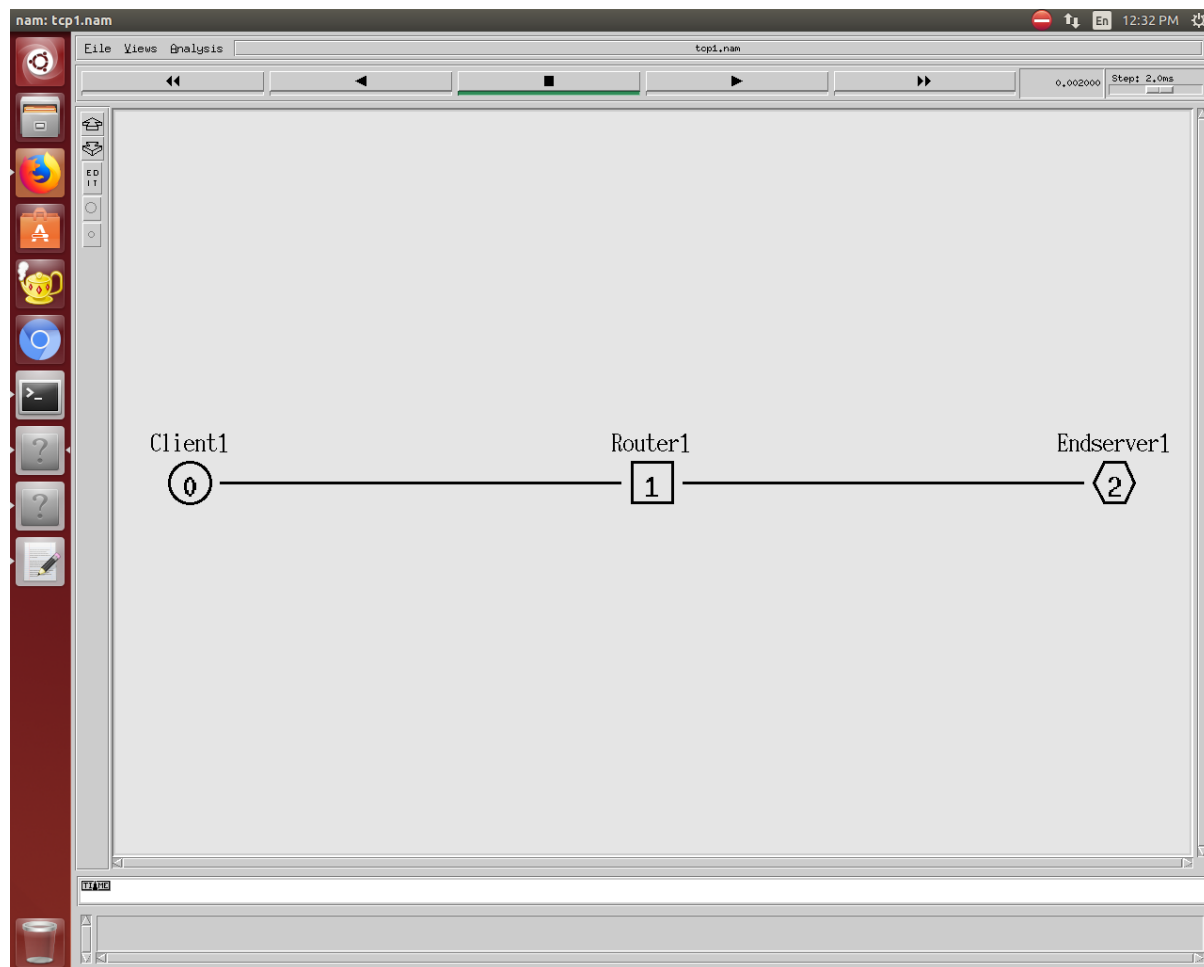
Program:

```
set ns [new Simulator]
set nf [open tcp1.nam w]
$ns namtrace-all $nf
set nt [open tcp1.tr w]
$ns trace-all $nt
$ns color 1 blue
$ns color 2 yellow
$ns color 3 red
set Client1 [$ns node]
set Router1 [$ns node]
set Endserver1 [$ns node]
$ns duplex-link $Client1 $Router1 2Mb 100ms DropTail
$ns duplex-link $Router1 $Endserver1 200Kb 100ms DropTail
#-----creating orientation-----#
$ns duplex-link-op $Client1 $Router1 orient right
$ns duplex-link-op $Router1 $Endserver1 orient right
#-----Labelling-----#
$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Endserver1 label Endserver1"
#-----Configuring nodes-----#
$Endserver1 shape hexagon
$Router1 shape square
```

```
proc finish {} {  
    global ns nf nt  
    $ns flush-trace  
    close $nf  
    close $nt  
  
    puts "running nam..."  
    exec nam tcp1.nam &  
    exit 0  
}
```

```
$ns at 6.0 "finish"  
$ns run
```

Output:



4. b. Write TCL script to create TCP agent, TCP sink and attach the TCO agent with TCP sink.

Procedure:

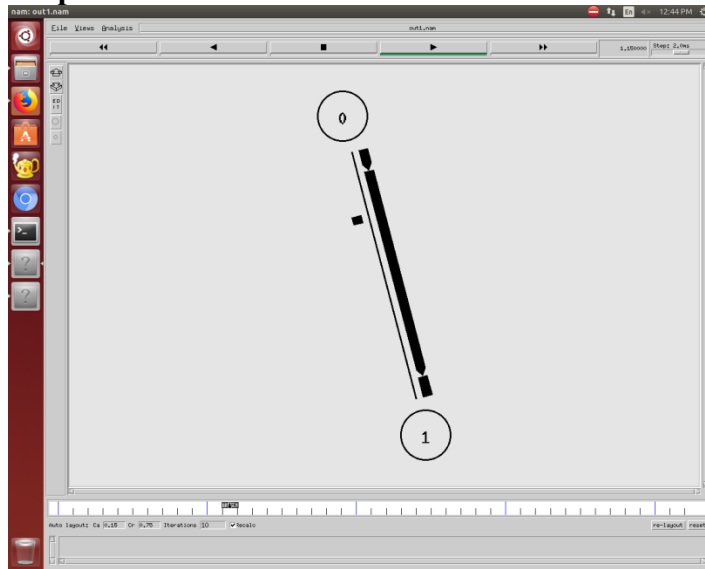
Open .ns file and create write as program as per the steps given below

1. Create 2 nodes
2. Open out.tr and out.name in write mode
3. Create duplex connection between 2 nodes with 1Mb 10ms DropTail
4. Create TCP agent and TCPSink
5. Attach TCP agent to node1 and TCPSink to node2
6. Create FTP traffic
7. Write finish procedure
8. Execute nam
9. Set start, stop and finish time

Program:

```
set ns [new Simulator]
$ns rtproto DV
set node1 [$ns node]
set node2 [$ns node]
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
$ns duplex-link $node1 $node2 1Mb 10ms DropTail
set tcp2 [new Agent/TCP]
$ns attach-agent $node1 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $node2 $sink2
$ns connect $tcp2 $sink2
set traffic_ftp2 [new Application/FTP]
$traffic_ftp2 attach-agent $tcp2
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
$ns at 1.0 "$traffic_ftp2 start"
$ns at 3.0 "$traffic_ftp2 stop"
$ns at 5.0 "finish"
$ns run
```

Output:



Viva Questions:

1. Give the formula for throughput.
2. How to create a router node?
3. Write syntax for labeling a node.
4. Write syntax for left orientation
5. How many orientations are possible with NS2?

5. Write TCL script to set identification Color to links.

Procedure:

Open .ns file and create write as program as per the steps given below

1. Create 2 nodes
2. Open out.tr and out.name in write mode
3. Create duplex connection between 2 nodes with 1Mb 10ms DropTail
4. Color the link using the following syntax:
 \$ns duplex-link-op \$Cn1 \$n2 color red
5. Write finish procedure

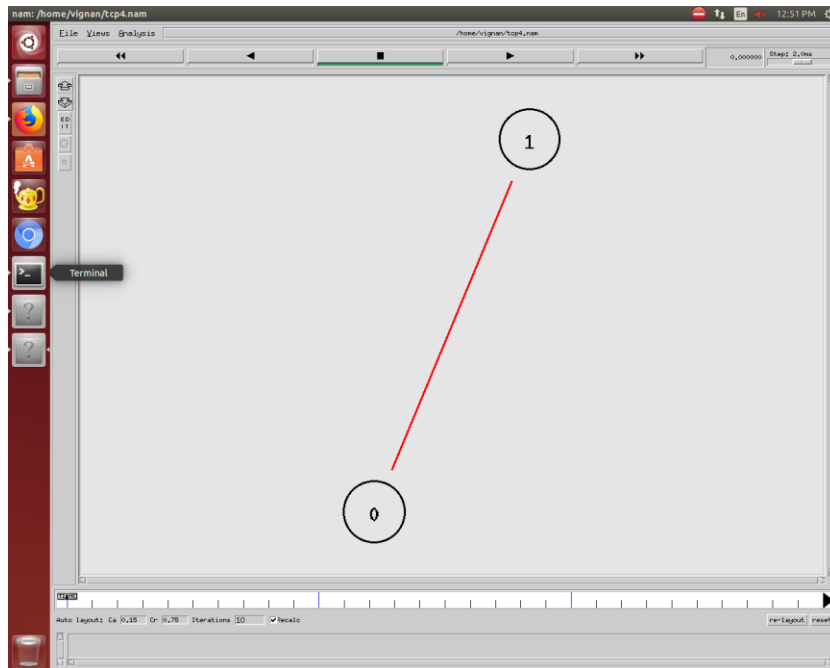
Program:

```
set ns [ new Simulator ]
set nf [open tcp4.nam w]
$ns namtrace-all $nf
set nt [open tcp4.tr w]
$ns trace-all $nt
$ns color 1 red
set C1 [$ns node]
set R1 [$ns node]
$ns duplex-link $C1 $R1 1Mb 10ms DropTail
$ns at 0.0 "$C1 label CL1"
$ns at 0.0 "$R1 label RC1"
$ns duplex-link-op $C1 $R1 color red

proc finish {} {
    global ns nf nt nfl

    $ns flush-trace
    close $nf
    puts "running nam..."
    exec nam tcp4.nam &
    exit 0
}
$ns at 20.0 "finish"
$ns run
```


Output:



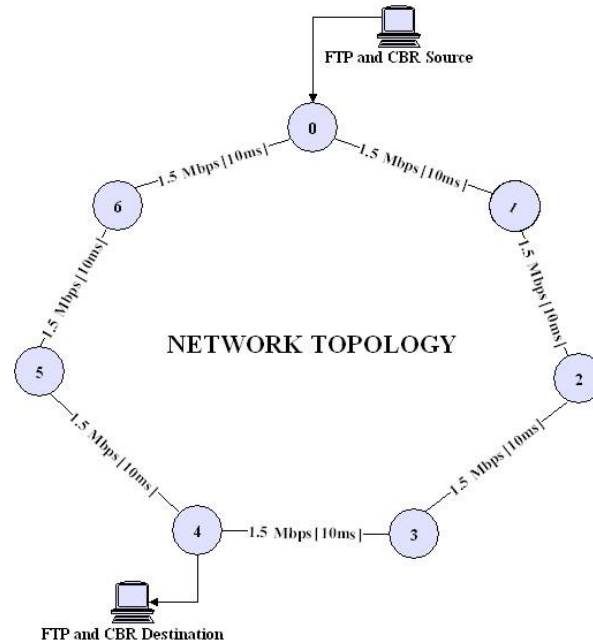
Viva Questions:

1. Write syntax for labeling a node.
2. How many orientations are possible with NS2?
3. Write syntax for coloring the link.
4. Write syntax for coloring a node.
5. What is Droptail?

6. Simulate Link State Routing (LS) protocol in NS2?

Consider the following steps for the simulation:

The network will consist of 7 nodes and we will have a ring topology. The topology is shown in the following figure.



The technical characteristics of the network are the following

- All the links will have bandwidth of 1.5 Mbps, delay of 10ms and they will have a DropTail queue
- We will have two types of traffics: FTP Traffic and CBR traffic. The CBR traffic will have rate of 1Mbps and the packet size will be 1000 bytes
 - Use **TCP** for the **FTP** Traffic
 - Use **UDP** for the **CBR** Traffic

Simulation Scenario

In this simulation we want to see how the Distance vector and Link state protocols behave in the presence of link or node failures. We will use the network model that we just described.

The Events of the scenario are:

- In time 0 the simulation will start
- The FTP and CBR will start in time 0.5 with source node0 and destination node4
- At time 1.0 the link between node4 and node5 will fail and will be restored at time 3.0
- At time 3.8 node6 will fail and will recover at time 4.5
- At time 5.0 the simulation will end

You will test the above scenario with link state routing protocol.

Some new commands that will be used in the simulation:

- **Set a Routing Protocol:** `$ns rtproto name_of_routing_protocol (LS for Link State)`
Note: For Static routes do not use this command.
- **Simulate Node Failure/Restoration:** `$ns rtmodel-at [time] [down/up] [node_id]`
Examples:
`$ns rtmodel-at 10.2 down $n16 #Node failure`
`$ns rtmodel-at 43.5 up $n16 #Node restoration`
- **Simulate Link Failure/Restoration:** `$ns rtmodel-at [time] [down/up] [node_id_from node_id_to]`

Program:

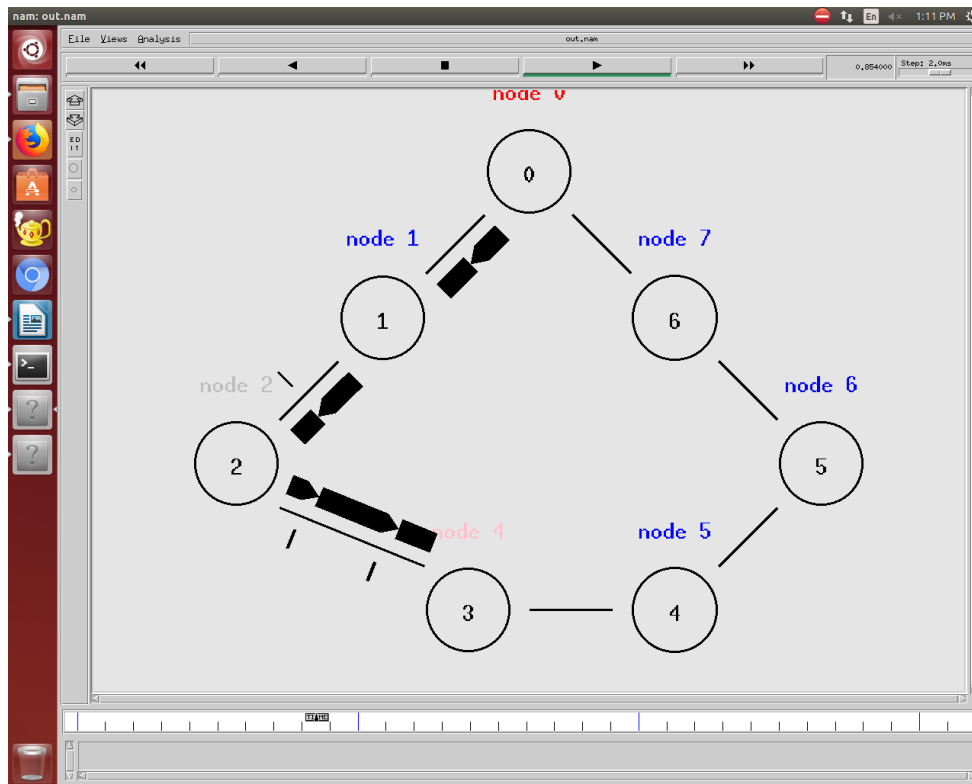
```
set ns [new Simulator]
$ns rtproto LS
set node1 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]
set node5 [$ns node]
set node6 [$ns node]
set node7 [$ns node]
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
$node1 label "node 0"
$node2 label "node 1"
$node3 label "node 2"
$node4 label "node 4"
$node5 label "node 5"
$node6 label "node 6"
$node7 label "node 7"
$node1 label-color red
$node2 label-color blue
$node3 label-color grey
$node4 label-color pink
$node5 label-color blue
$node6 label-color blue
$node7 label-color blue
$node1 id
$node2 id
$node3 id
$node4 id
$node5 id
$node6 id
$node7 id
$ns duplex-link $node1 $node2 1.5Mb 10ms DropTail
```

```

$ns duplex-link $node2 $node3 1.5Mb 10ms DropTail
$ns duplex-link $node3 $node4 1.5Mb 10ms DropTail
$ns duplex-link $node4 $node5 1.5Mb 10ms DropTail
$ns duplex-link $node5 $node6 1.5Mb 10ms DropTail
$ns duplex-link $node6 $node7 1.5Mb 10ms DropTail
$ns duplex-link $node7 $node1 1.5Mb 10ms DropTail
$ns duplex-link-op $node1 $node2 orient left-down
$ns duplex-link-op $node2 $node3 orient left-down
$ns duplex-link-op $node3 $node4 orient right-down
$ns duplex-link-op $node4 $node5 orient right
$ns duplex-link-op $node5 $node6 orient right-up
$ns duplex-link-op $node6 $node7 orient left-up
$ns duplex-link-op $node7 $node1 orient left-up
set tcp2 [new Agent/TCP]
$ns attach-agent $node1 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $node4 $sink2
$ns connect $tcp2 $sink2
set traffic_ftp2 [new Application/FTP]
$traffic_ftp2 attach-agent $tcp2
set udp0 [new Agent/UDP]
$ns attach-agent $node1 $udp0
set null0 [new Agent/LossMonitor]
$ns attach-agent $node4 $null0
$ns connect $udp0 $null0
set traffic_cbr0 [new Application/Traffic/CBR]
$traffic_cbr0 attach-agent $udp0
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
$ns at 0.5 "$traffic_ftp2 start"
$ns rtmodel-at 1.0 down $node2 $node3
$ns rtmodel-at 2.0 up $node2 $node3
$ns at 2.5 "$traffic_ftp2 start"
#$ns rtmodel-at 3.8 down $node2
#$ns rtmodel-at 4.5 up $node2
$ns at 3.0 "$traffic_ftp2 stop"
$ns at 5.0 "finish"
$ns run

```

Output:



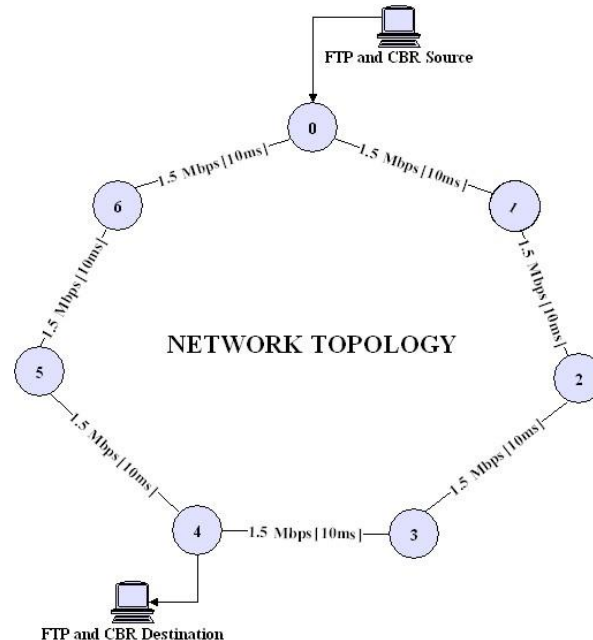
Viva Questions:

1. How to simulate node failure?
2. How to simulate node restoration?
3. Which path is selected initially for transmitting the packet and why?
4. What is LossMonitor?
5. How to set CBR traffic?

7. Simulate Distance Vector Routing (DV) protocol in NS2?

Consider the following steps for the simulation:

The network will consist of 7 nodes and we will have a ring topology. The topology is shown in the following figure.



The technical characteristics of the network are the following

- All the links will have bandwidth of 1.5 Mbps, delay of 10ms and they will have a DropTail queue
- We will have two types of traffics: FTP Traffic and CBR traffic. The CBR traffic will have rate of 1Mbps and the packet size will be 1000 bytes
 - Use **TCP** for the **FTP** Traffic
 - Use **UDP** for the **CBR** Traffic

Simulation Scenario

In this simulation we want to see how the Distance vector and Link state protocols behave in the presence of link or node failures. We will use the network model that we just described.

The Events of the scenario are:

- In time 0 the simulation will start
- The FTP and CBR will start in time 0.5 with source node0 and destination node4
- At time 1.0 the link between node4 and node5 will fail and will be restored at time 3.0
- At time 3.8 node6 will fail and will recover at time 4.5
- At time 5.0 the simulation will end

You will test the above scenario with link state routing protocol.

Some new commands that will be used in the simulation:

- **Set a Routing Protocol:** `$ns rtproto name_of_routing_protocol (DV for Distance Vector)`
Note: For Static routes do not use this command.
- **Simulate Node Failure/Restoration:** `$ns rtmodel-at [time] [down/up] [node_id]`
Examples:
`$ns rtmodel-at 10.2 down $n16 #Node failure`
`$ns rtmodel-at 43.5 up $n16 #Node restoration`
- **Simulate Link Failure/Restoration:** `$ns rtmodel-at [time] [down/up] [node_id_from node_id_to]`

Program:

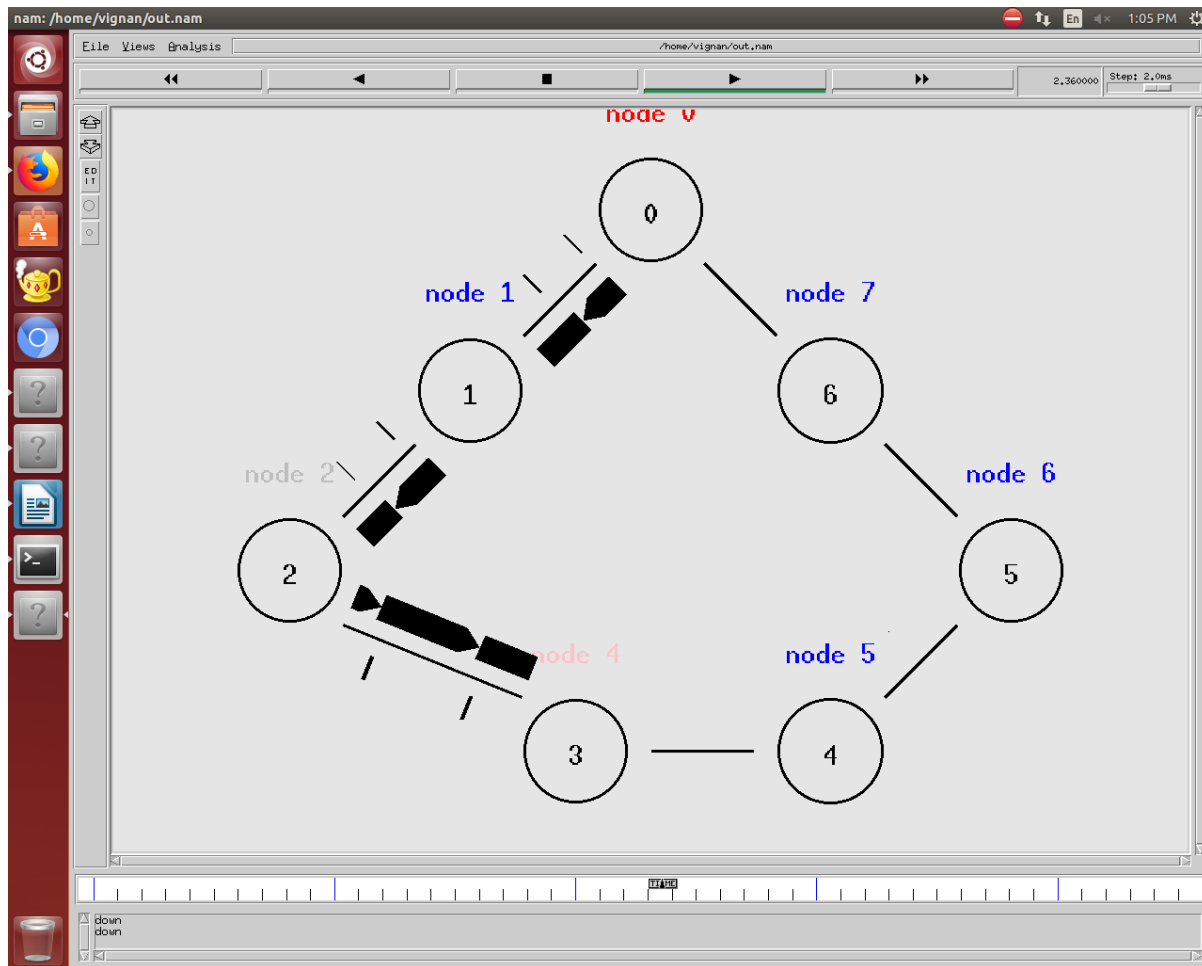
```
set ns [new Simulator]
$ns rtproto DV
set node1 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]
set node5 [$ns node]
set node6 [$ns node]
set node7 [$ns node]
set tf [open out.tr w]
$ns trace-all $tf
set nf [open out.nam w]
$ns namtrace-all $nf
$node1 label "node 0"
$node2 label "node 1"
$node3 label "node 2"
$node4 label "node 4"
$node5 label "node 5"
$node6 label "node 6"
$node7 label "node 7"
$node1 label-color red
$node2 label-color blue
$node3 label-color grey
$node4 label-color pink
$node5 label-color blue
$node6 label-color blue
$node7 label-color blue
$node1 id
$node2 id
$node3 id
$node4 id
$node5 id
$node6 id
$node7 id
$ns duplex-link $node1 $node2 1.5Mb 10ms DropTail
```

```

$ns duplex-link $node2 $node3 1.5Mb 10ms DropTail
$ns duplex-link $node3 $node4 1.5Mb 10ms DropTail
$ns duplex-link $node4 $node5 1.5Mb 10ms DropTail
$ns duplex-link $node5 $node6 1.5Mb 10ms DropTail
$ns duplex-link $node6 $node7 1.5Mb 10ms DropTail
$ns duplex-link $node7 $node1 1.5Mb 10ms DropTail
$ns duplex-link-op $node1 $node2 orient left-down
$ns duplex-link-op $node2 $node3 orient left-down
$ns duplex-link-op $node3 $node4 orient right-down
$ns duplex-link-op $node4 $node5 orient right
$ns duplex-link-op $node5 $node6 orient right-up
$ns duplex-link-op $node6 $node7 orient left-up
$ns duplex-link-op $node7 $node1 orient left-up
set tcp2 [new Agent/TCP]
$ns attach-agent $node1 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $node4 $sink2
$ns connect $tcp2 $sink2
set traffic_ftp2 [new Application/FTP]
$traffic_ftp2 attach-agent $tcp2
set udp0 [new Agent/UDP]
$ns attach-agent $node1 $udp0
set null0 [new Agent/LossMonitor]
$ns attach-agent $node4 $null0
$ns connect $udp0 $null0
set traffic_cbr0 [new Application/Traffic/CBR]
$traffic_cbr0 attach-agent $udp0
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
$ns at 0.5 "$traffic_ftp2 start"
$ns rtmodel-at 1.0 down $node2 $node3
$ns rtmodel-at 2.0 up $node2 $node3
$ns at 2.5 "$traffic_ftp2 start"
#$ns rtmodel-at 3.8 down $node2
#$ns rtmodel-at 4.5 up $node2
$ns at 3.0 "$traffic_ftp2 stop"
$ns at 5.0 "finish"
$ns run

```


Output:



Viva Questions:

1. How to simulate node failure?
2. How to simulate node restoration?
3. Which path is selected initially for transmitting the packet and why?
4. What is LossMonitor?
5. How to set CBR traffic?

8. Develop TCL script to make TCP communication between nodes using DSR routing protocol.

Description:

Number of nodes (3) is fixed in the program. Nodes are configured with specific parameters of a mobile wireless node. After creating the nam file and trace file, we set up topography object. set node_ (\$i) [\$ns node] is used to create the nodes. Initial location of the nodes is fixed. Specific X, Y coordinates are assigned to every node. Nodes are given mobility with fixed speed and fixed destination location. Here we set the initial size for the every node by using initial_node_pos. DSR routing protocol is used here. \$val(stop) specifies the end time of the simulation. TCP agent is attached to node_ (0). TCPSink agent is attached to node_(1). Both the agents are connected and FTP application is attached to TCP agent. Now communication set up for node_(0) and node_(1) is established.

Program:

```
#Creating trace file and nam file
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(nn) 3
set val(rp) DSR
set val(x) 500
set val(y) 400
set val(stop) 150
set ns [ new Simulator ]

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 < $val(nn) } { incr i } {
        set node_($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)} { incr i } {
        # 30 defines the node size for nam
        $ns initial_node_pos $node_($i) 30
    }

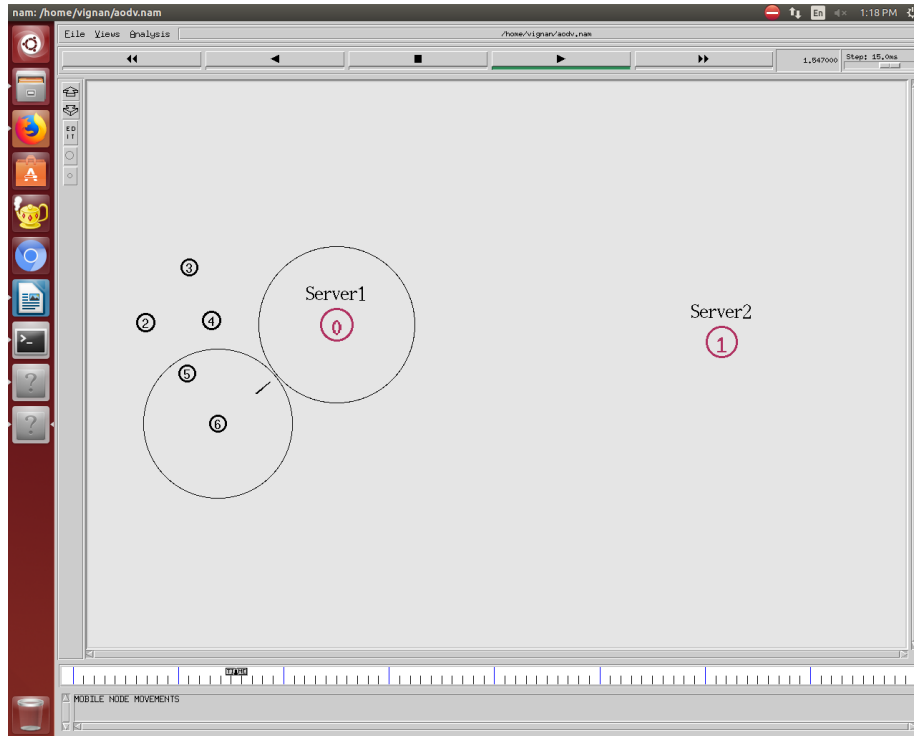
    # Telling nodes when the simulation ends
    for {set i 0} {$i < $val(nn) } { incr i } {
        $ns at $val(stop) "$node_($i) reset";
    }

    # 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 tracefd namtrace
        $ns flush-trace
        close $tracefd
        close $namtrace
    }
    exec nam dsr.nam &
    exit 0
}

$ns run

```

Output:



Viva Questions:

1. How to fix initial location?
2. How to generate movements for nodes?
3. Give Syntax for window size.
4. How to setup topography object?
5. How DSR protocol works?

9. Write TCL script to make communication between nodes using AODV routing protocol and CBR traffic.

Description:

Number of nodes (7) is fixed in the program. Nodes are configured with specific parameters of a mobile wireless node. After creating the nam file and trace file, we set up topography object. set node_ (\$i) [\$ns node] is used to create the nodes. Initial location of the nodes is fixed. Specific X, Y coordinates are assigned to every node. Nodes are given mobility with fixed speed and fixed destination location. Here we set the initial size for the every node by using initial_node_pos. AODV routing protocol is used here. \$val(stop) specifies the end time of the simulation. UDP agent is attached to sender node. LossMonitor agent is attached to receiver node. Both the agents are connected and CBR traffic is attached to UDP agent. Now communication set up for nodes are established.

Program:

```
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(nn) 22
set val(rp) AODV
set val(x) 1800
set val(y) 840
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 chan_1_ [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 $chan_1_

### 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 opt(seed) 0.1
    set a [ns-random $opt(seed)]
    set i 0
    while {$i < 5} {
        incr i
    }

### 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

## 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"

## 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

#### 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
$n4 color blue
$ns_ at 4.71 "$n4 color blue"
$n5 color blue
$ns_ at 7.0 "$n5 color blue"
$n2 color blue
$ns_ at 7.29 "$n2 color blue"
$n4 color maroon
$ns_ at 7.44 "$n4 color maroon"
$ns_ at 7.43 "$n4 label TTlover"
$ns_ at 7.55 "$n4 label \"\""
$n3 color blue
$ns_ at 7.85 "$n3 color blue"

#### Establishing Communication
set udp0 [$ns_ create-connection UDP $Server1 LossMonitor $n6 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 $n5 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 $n5 LossMonitor $n6 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 $n3 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"

```

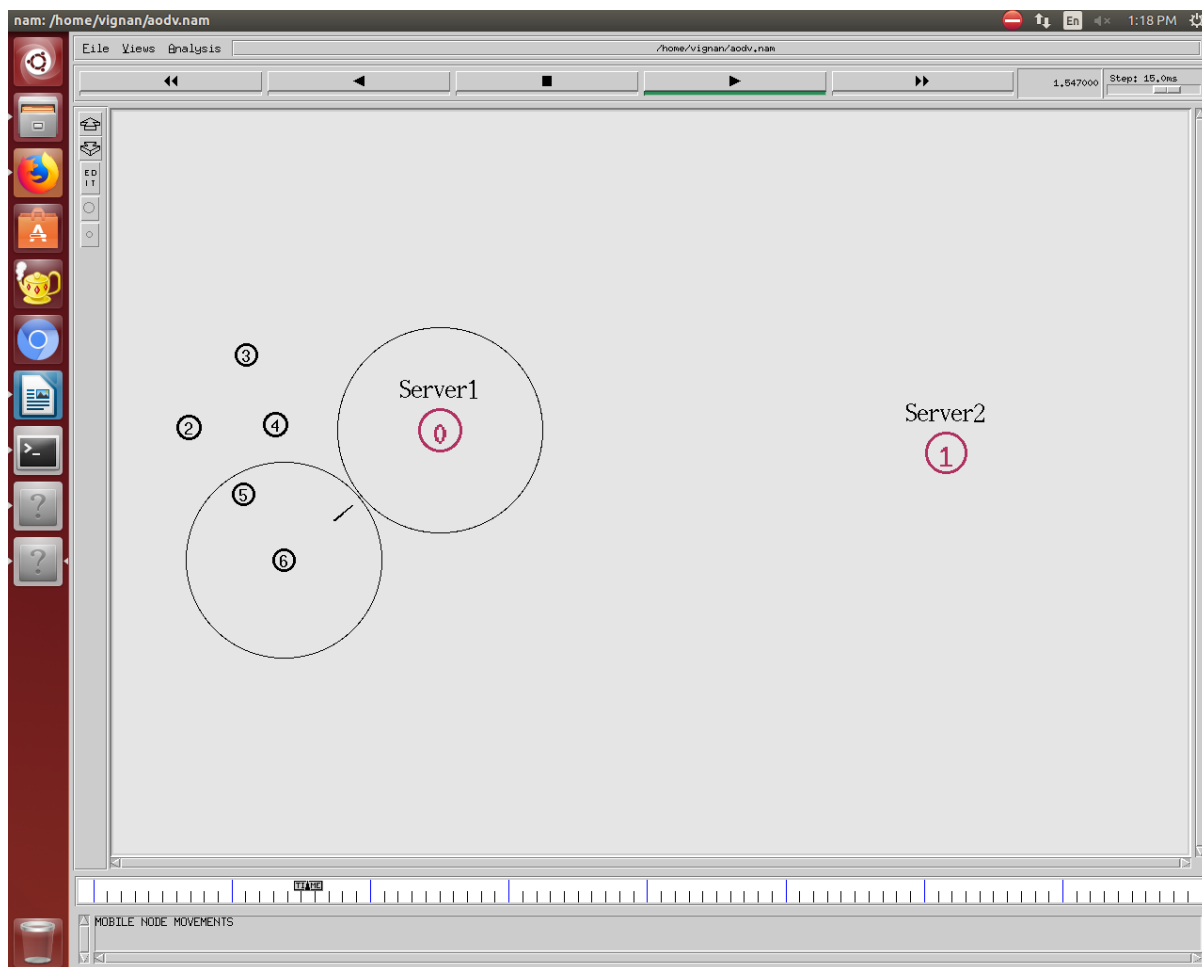
```

set udp4 [$ns_ create-connection UDP $Server1 LossMonitor $n2 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 $n2 LossMonitor $n3 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 $n4 LossMonitor $n5 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 $n3 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"
#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\""
$ns_ at 4.71 "$ns_ trace-annotate \"NODE10 CACHE THE
DATA\""

### PROCEDURE TO STOP
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
    exec nam aodv.nam &
    exit 0
}
puts "Starting Simulation....."
$ns_ at 25.0 "stop"
$ns_ run

```

Output:



Viva Questions:

1. How to fix initial location?
2. How to generate movements for nodes?
3. Give Syntax for window size.
4. How to setup topography object?
5. How AODV protocol work?

10. Develop and implement TCL script to make TCP communication between nodes using DSDV routing protocol.

Description:

Number of nodes (3) is fixed in the program. Nodes are configured with specific parameters of a mobile wireless node. After creating the nam file and trace file, we set up topography object. set node_ (\$i) [\$ns node] is used to create the nodes. Initial location of the nodes is fixed. Specific X, Y coordinates are assigned to every node. Nodes are given mobility with fixed speed and fixed destination location. Here we set the initial size for the every node by using initial_node_pos. DSDV routing protocol is used here. \$val(stop) specifies the end time of the simulation. TCP agent is attached to node_ (0). TCPSink agent is attached to node_(1). Both the agents are connected and FTP application is attached to TCP agent. Now communication set up for node_(0) and node_(1) is established.

Program:

```
#Creating trace file and nam file
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(nn) 3
set val(rp) DSDV
set val(x) 500
set val(y) 400
set val(stop) 150
set ns [ new Simulator ]

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 < $val(nn) } { incr i } {
        set node_($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)} { incr i } {
    # 30 defines the node size for nam
    $ns initial_node_pos $node_($i) 30
}

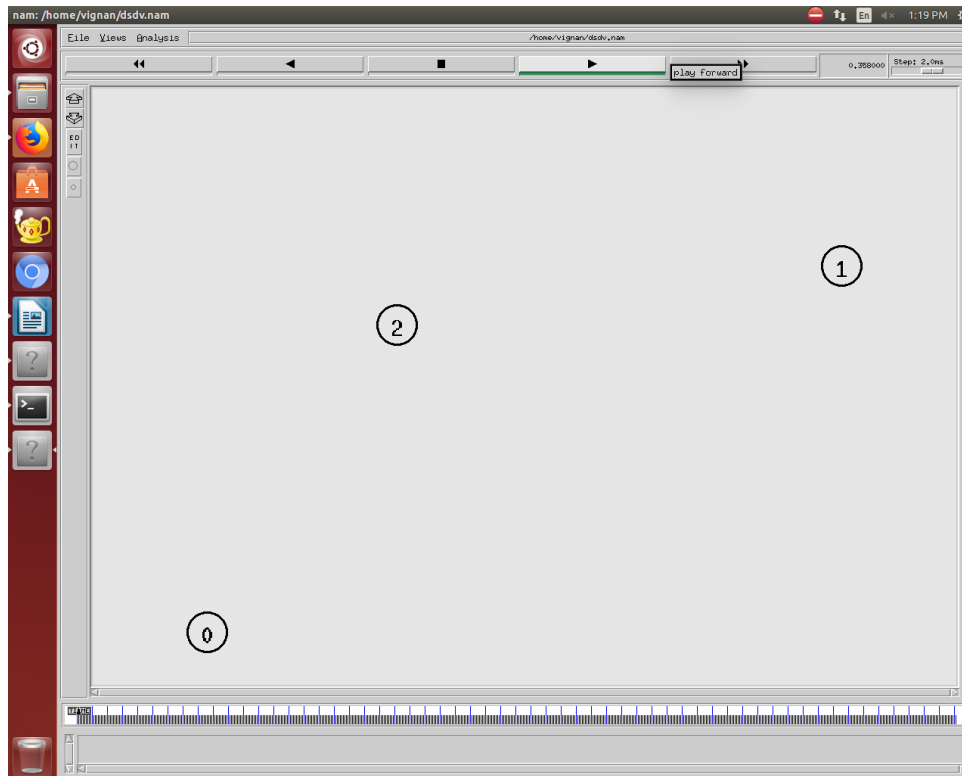
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}

# 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 tracefd namtrace
    $ns flush-trace
    close $tracefd
    close $namtrace
    exec nam dsdv.nam &
    exit 0
}

$ns run

```

Output:



Viva Questions:

1. How to fix initial location?
2. How to generate movements for nodes?
3. Give Syntax for window size.
4. How to setup topography object?
5. How DSDV protocol work?