Roll Number : **16** Seat Number: IT 4A016

Vivekanand Education Society's Institute of Technology
Hashu Advani memorial Complex
Collector's Colony, R C Marg, Chembur Mumbai 400074, Phone Number 022-61532532
www.vesit.edu

CERTIFICATE

Certified that **Ms Khushboo Chandnani** Of **D10 SE** semester **IV** has successfully completed necessary experiments in **Networking Lab** course under my supervision in VES Institute of Technology during academic Year **2017-2018**

Lab in-charge
Subject Teacher

Name of the Course	**:**	**Computer Network**
Class	**:**	**D10**
Semester	**:**	**IV**
Year	**:**	**2017 -18**

Programme Educational Objectives(PEOs)

The objectives of a programme are:

I. To provide students with a solid foundation in the core engineering concepts like mathematics, programming, data management, networking etc. This will further enable students to analyse, design and create solutions for any enterprise, national or global in multidisciplinary fields.

II. To inculcate in students a strong ethical and professional attitude which along with effective communication, managerial and teamwork skills will enable success in a broad social context.

III. To provide students with an environment programmed for academic excellence, leadership, and life-long learning needed for a successful professional career.

IV. To empower and enable students with the capabilities to develop high end business and innovation skills.

Program Outcomes:

PO1.Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2.Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3.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.

PO4.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.

PO5.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.

PO6.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.

PO7.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.

PO8.Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9.Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10.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.

PO11.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.

PO12.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.

Program specified outcomes:

PSO1. Professional Skills - The ability to develop programs for computer based systems of varying complexity and domains using standard practices.

PSO2. Successful Career – The ability to adopt skills, languages, environment and platforms for creating innovative carrier paths, being successful entrepreneurs or for pursuing higher studies.

Course Objectives:

Students will try to:

1. Study the basic taxonomy and terminology of the computer networking and enumerate the layers of OSI model and TCP/IP model.
2. Acquire knowledge of Application layer and Presentation layer paradigms and protocols.
3. Study Session layer design issues, Transport layer services, and protocols.
4. Gain core knowledge of Network layer routing protocols and IP addressing.
5. Study data link layer concepts, design issues, and protocols.
6. Read the fundamentals and basics of Physical layer, and will apply them in real time applications.

Course Outcomes:
Students will be able to:
1. Describe the functions of each layer in OSI and TCP/IP model.
2. Explain the functions of Application layer and Presentation layer paradigms and Protocols.
3. Describe the Session layer design issues and Transport layer services.
4. Classify the routing protocols and analyze how to assign the IP addresses for the given network. Describe the functions of data link layer and explain the protocols.
5. Explain the types of transmission media with real time applications.

Laboratory Evaluation Plan

Roll No: 31

Lab Outcomes:
1. Execute and evaluate network administration commands and demonstrate their use in different network scenarios
2. Demonstrate the installation and configuration of network simulator.
3. Demonstrate and measure different network scenarios and their performance behavior.
4. Analyze the contents the packet contents of different protocols.
5. Implement the socket programming for client server architecture.

Index

Sr. No	Topic	Lab Outcomes Outcomes	DOP	DOS	Marks
1	Networking Commands	LO1	16/01/2018	24/01/2018	8
2	Ns2 Installation	LO2	23/01/2018	31/01/2018	10
3	TCL Programming	LO2	30/01/2018	07/02/2018	8
4	FTP over TCP	LO3	07/02/2018	14/02/2018	10
5	CBR over UDP	LO3	21/02/2018	07/03/2018	10
6	Wireless Topology	LO3	14/03/2018	14/03/2018	10
7	Analysis of network performance for quality of service parameters	LO3	14/03/2018	21/03/2018	10
8	Socket Programming with C/Java: TCP Client, TCP Server	LO5	28/03/2018	04/04/2018	10
9	Socket Programming with C/Java: UDP Client, UDP Server	LO5	28/03/2018	04/04/2018	10
10	Analysis of Packet headers using Wireshark	LO4	04/04/2018	05/04/2018	9

10					
	Assignments				
1	Change colour of node, data flow, shape, link type)	LO3	07/02/2018	17/02/2018	10
2	Design the given Topology	LO3	28/02/2018	10/03/2018	10
3	Assignment -3:Configuring transport layer protocol	LO3	28/02/2018	10/03/2018	10
Average Grade					9.62

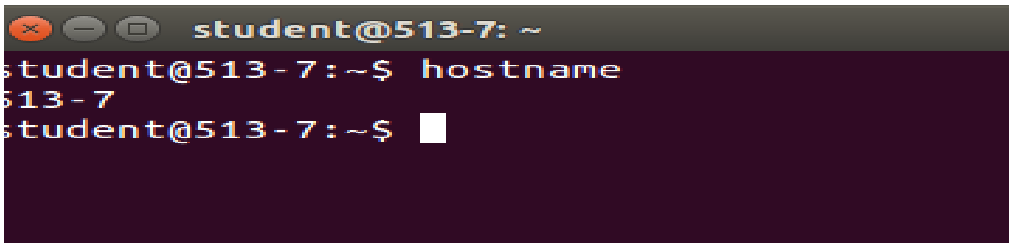
Teacher Sign: ---------

Experiment 1

Aim: Networking Commands

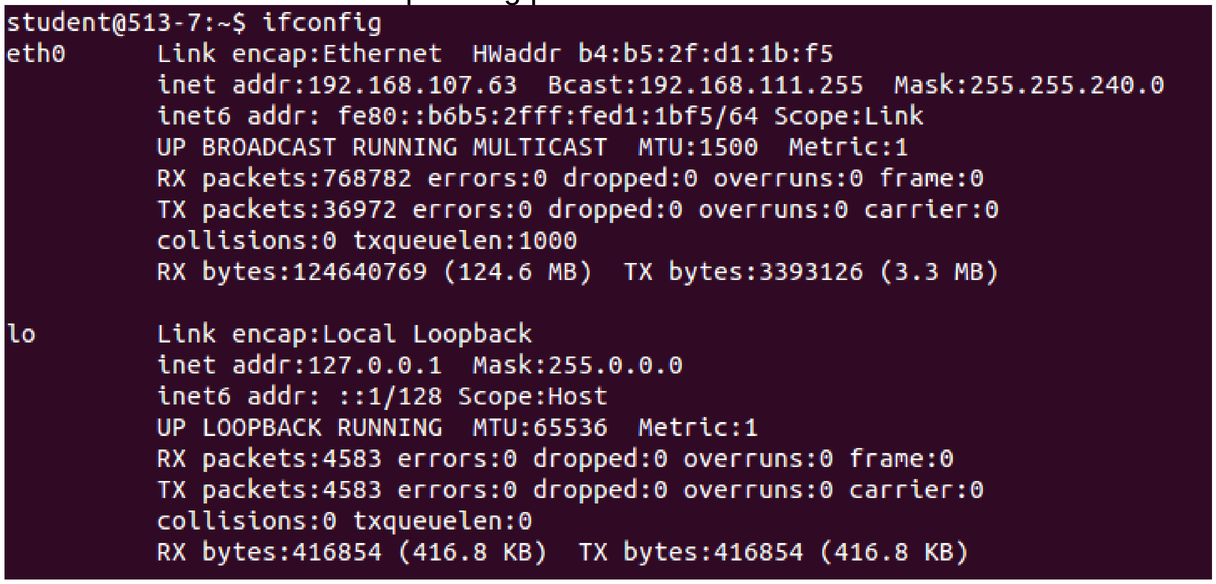
Commands:

- **Hostname**
  A hostname is the label (the name) assigned to a device (a host) on a network and is used to distinguish one device from another on a specific network or over the Internet. ... A computer's hostname may instead be referred to as a computer name, sitename, or nodename. You may also see hostname spelled as host name.
  hostname can further be of the following types :
    1. hostname -d : Displays the domain name.
    2. hostname -f : Displays the full domain name.
    3. hostname -i



```
student@513-7: ~
student@513-7:~$ hostname
513-7
student@513-7:~$
```

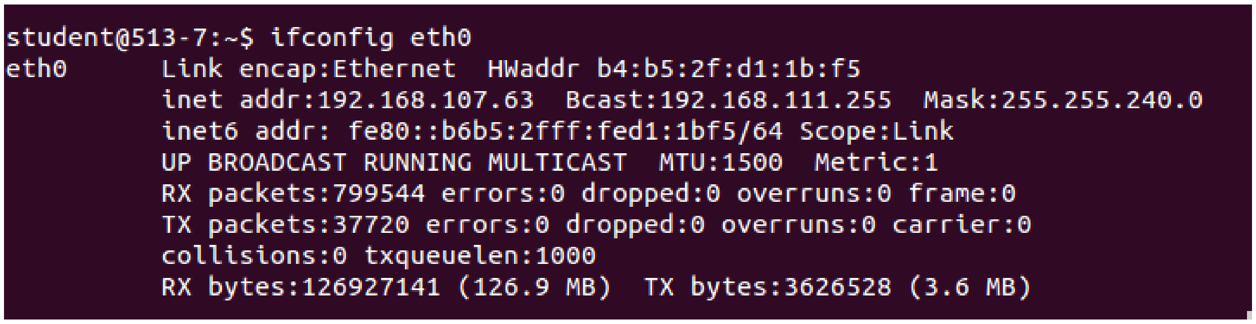
- **Ifconfig**
  You can use the **ifconfig** command to assign an address to a network interface and to configure or display the current network interface configuration information.After system startup, it can also be used to redefine an interfaces address and its other operating parameters.



```
student@513-7:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr b4:b5:2f:d1:1b:f5
          inet addr:192.168.107.63  Bcast:192.168.111.255  Mask:255.255.240.0
          inet6 addr: fe80::b6b5:2fff:fed1:1bf5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:768782 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36972 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:124640769 (124.6 MB)  TX bytes:3393126 (3.3 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4583 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4583 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:416854 (416.8 KB)  TX bytes:416854 (416.8 KB)
```

  Further ifconfig can perform the following functionalities :
  ifconfig <network name> :

- Displays the description (ip address, Mask, MAC address, Broadcast address etc) of the specified network.



```
student@513-7:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr b4:b5:2f:d1:1b:f5
          inet addr:192.168.107.63  Bcast:192.168.111.255  Mask:255.255.240.0
          inet6 addr: fe80::b6b5:2fff:fed1:1bf5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:799544 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37720 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:126927141 (126.9 MB)  TX bytes:3626528 (3.6 MB)
```

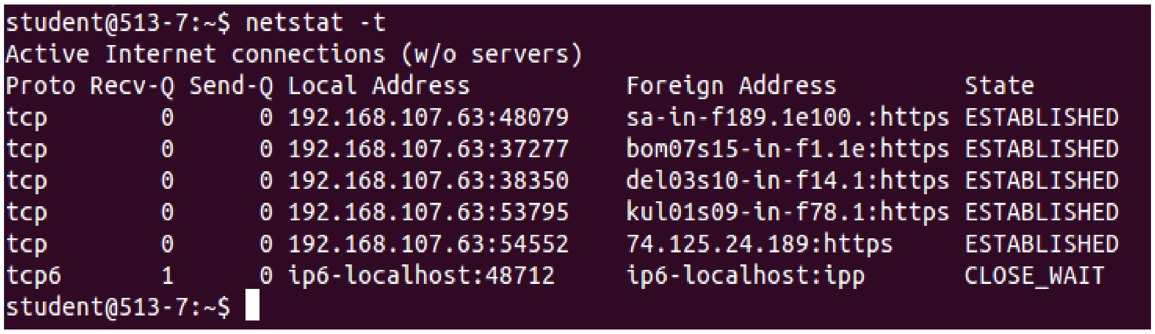
Ifconfig <network name> down :

- It ends the connection with the specified network.

Ifconfig <network name> up :

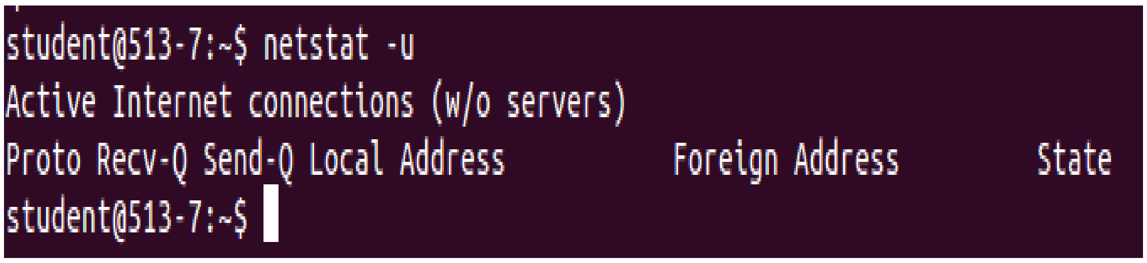
- It restarts the connection with the specified network.

- **Netstat**
  Netstat is a common command line TCP/IP networking utility available in most versions of Windows, Linux, UNIX and other operating systems. Netstat provides information and statistics about protocols in use and current TCP/IP network connections. (The name derives from the words network and statistics.)
  - Different versions of netstat are :
    1. netstat -t : used to display TCP.
    2. netstat -u : used to display UDP.
    3. netstat -a : used to display All ACTIVE connections.
  - **netstat -a**
  - **netstat -t**



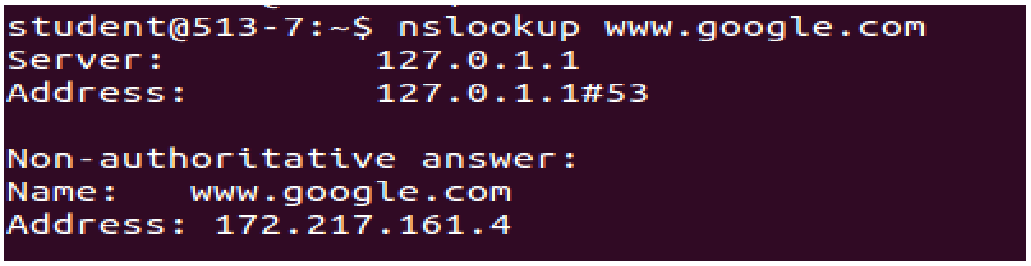
```
student@513-7:~$ netstat -t
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.107.63:48079    sa-in-f189.1e100.:https ESTABLISHED
tcp        0      0 192.168.107.63:37277    bom07s15-in-f1.1e:https ESTABLISHED
tcp        0      0 192.168.107.63:38350    del03s10-in-f14.1:https ESTABLISHED
tcp        0      0 192.168.107.63:53795    kul01s09-in-f78.1:https ESTABLISHED
tcp        0      0 192.168.107.63:54552    74.125.24.189:https     ESTABLISHED
tcp6       1      0 ip6-localhost:48712     ip6-localhost:ipp       CLOSE_WAIT
student@513-7:~$ 
```

  - **netstat -u**



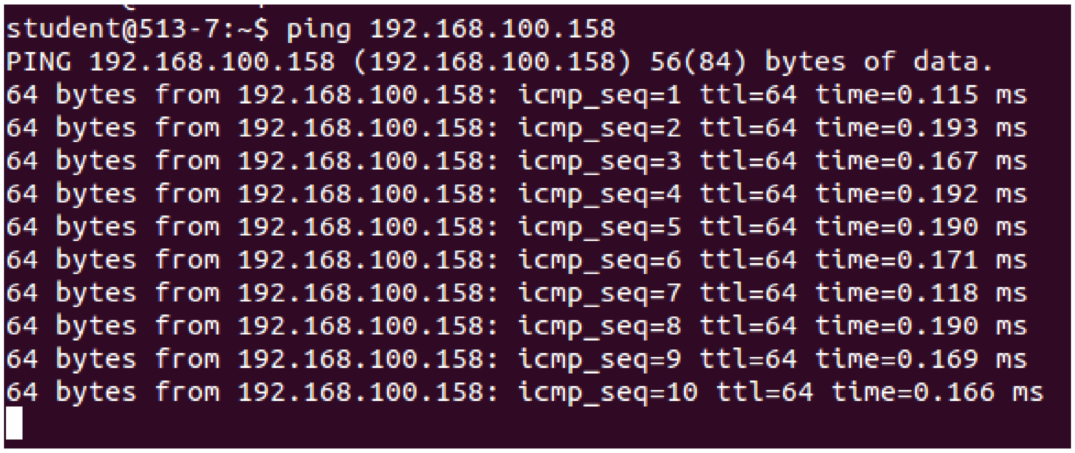
```
student@513-7:~$ netstat -u
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
student@513-7:~$
```

- **Nslookup**
  nslookup is a useful tool for troubleshooting DNS problems, such as host name resolution. When you start Nslookup,it is used to obtain IP address of the domain name and vice versa.



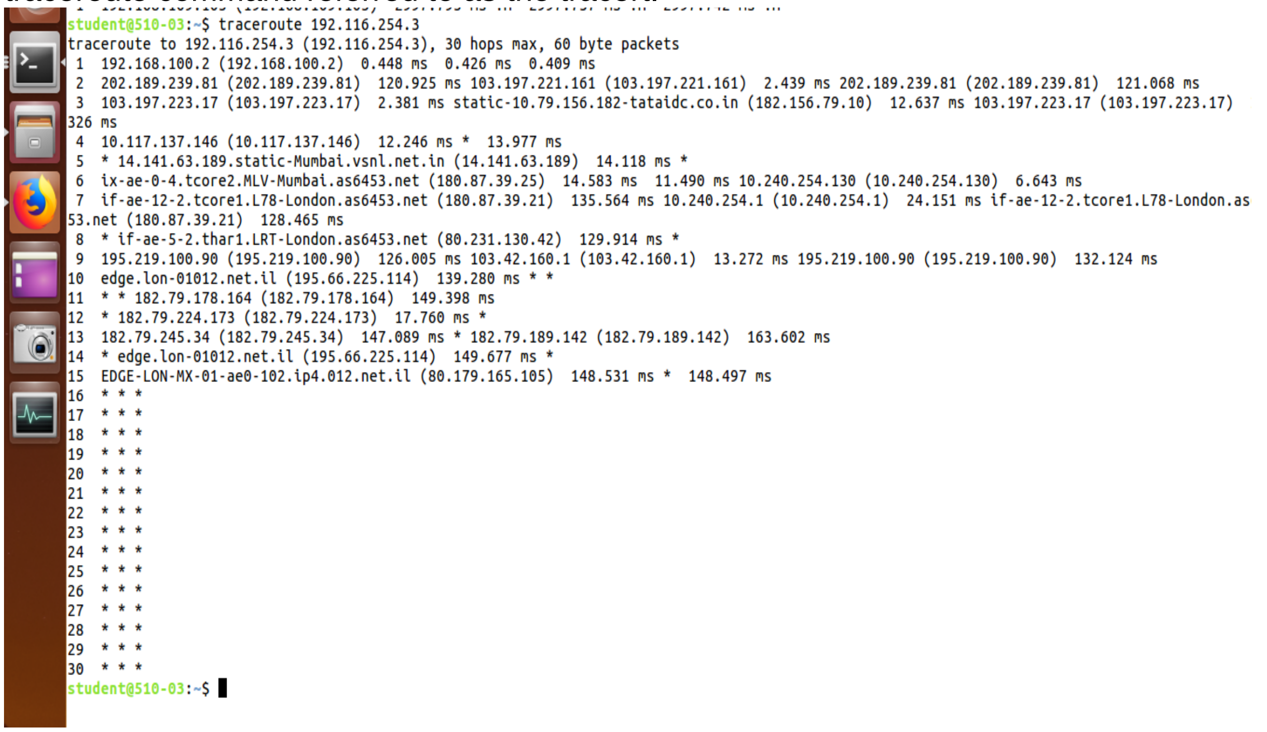
```
student@513-7:~$ nslookup www.google.com
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
Name:   www.google.com
Address: 172.217.161.4
```

- **Ping**
  The ping command is a Command Prompt command used to test the ability of the source computer to reach a specified destination computer. The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response.The ping command returns the IP address, the size and the time taken to transfer the packets.
  Type “ping” and then type in the address you'd like to ping and then hit the Enter key on your keyboard.



```
student@513-7:~$ ping 192.168.100.158
PING 192.168.100.158 (192.168.100.158) 56(84) bytes of data.
64 bytes from 192.168.100.158: icmp_seq=1 ttl=64 time=0.115 ms
64 bytes from 192.168.100.158: icmp_seq=2 ttl=64 time=0.193 ms
64 bytes from 192.168.100.158: icmp_seq=3 ttl=64 time=0.167 ms
64 bytes from 192.168.100.158: icmp_seq=4 ttl=64 time=0.192 ms
64 bytes from 192.168.100.158: icmp_seq=5 ttl=64 time=0.190 ms
64 bytes from 192.168.100.158: icmp_seq=6 ttl=64 time=0.171 ms
64 bytes from 192.168.100.158: icmp_seq=7 ttl=64 time=0.118 ms
64 bytes from 192.168.100.158: icmp_seq=8 ttl=64 time=0.190 ms
64 bytes from 192.168.100.158: icmp_seq=9 ttl=64 time=0.169 ms
64 bytes from 192.168.100.158: icmp_seq=10 ttl=64 time=0.166 ms
```

- **Traceroute**
  The traceroute command is a Command Prompt command that's used to show several details about the path that a packet takes from the computer or device you're on to whatever destination you specify. You might also sometimes see the traceroute command referred to as the tracert.



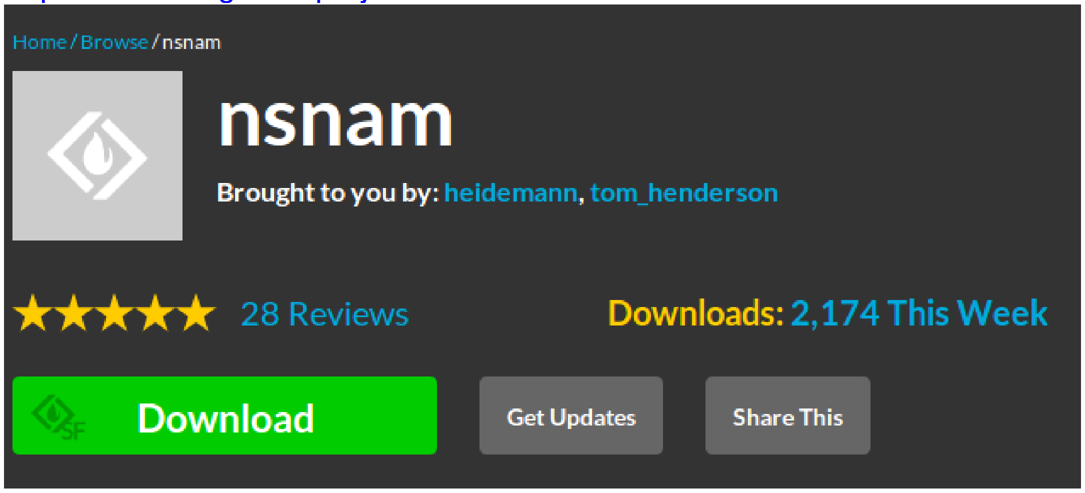
```
student@510-03:~$ traceroute 192.116.254.3
traceroute to 192.116.254.3 (192.116.254.3), 30 hops max, 60 byte packets
 1  192.168.100.2 (192.168.100.2)  0.448 ms  0.426 ms  0.409 ms
 2  202.189.239.81 (202.189.239.81)  120.925 ms 103.197.221.161 (103.197.221.161)  2.439 ms 202.189.239.81 (202.189.239.81)  121.068 ms
 3  103.197.223.17 (103.197.223.17)  2.381 ms static-10.79.156.182-tataidc.co.in (182.156.79.10)  12.637 ms 103.197.223.17 (103.197.223.17)
326 ms
 4  10.117.137.146 (10.117.137.146)  12.246 ms *  13.977 ms
 5  * 14.141.63.189.static-Mumbai.vsnl.net.in (14.141.63.189)  14.118 ms *
 6  ix-ae-0-4.tcore2.MLV-Mumbai.as6453.net (180.87.39.25)  14.583 ms  11.490 ms 10.240.254.130 (10.240.254.130)  6.643 ms
 7  if-ae-12-2.tcore1.L78-London.as6453.net (180.87.39.21)  135.564 ms 10.240.254.1 (10.240.254.1)  24.151 ms if-ae-12-2.tcore1.L78-London.as
53.net (180.87.39.21)  128.465 ms
 8  * if-ae-5-2.thar1.LRT-London.as6453.net (80.231.130.42)  129.914 ms *
 9  195.219.100.90 (195.219.100.90)  126.005 ms 103.42.160.1 (103.42.160.1)  13.272 ms 195.219.100.90 (195.219.100.90)  132.124 ms
10  edge.lon-01012.net.il (195.66.225.114)  139.280 ms * *
11  * * 182.79.178.164 (182.79.178.164)  149.398 ms
12  * 182.79.224.173 (182.79.224.173)  17.760 ms *
13  182.79.245.34 (182.79.245.34)  147.089 ms * 182.79.189.142 (182.79.189.142)  163.602 ms
14  * edge.lon-01012.net.il (195.66.225.114)  149.677 ms *
15  EDGE-LON-MX-01-ae0-102.ip4.012.net.il (80.179.165.105)  148.531 ms *  148.497 ms
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
student@510-03:~$
```

EXPERIMENT 2

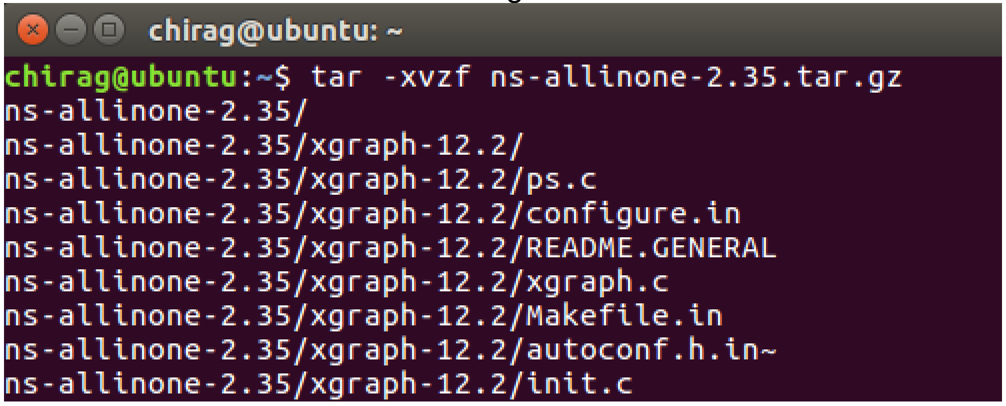
Aim: To download,extract and install Ns2 and also check its working.

Theory: NS (Network Simulator) is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Ns2’s primary usage was to analyse the performance of congestion control algorithms implemented in TCP. Even today, it remains the most widely used network simulator for TCP research. Over the period of time, it gained wide acceptance in industry, and now supports simulation of latest networking paradigms such as MANETs, VANETs, etc.
Following are the steps that have to followed in order to install, extract and test Ns2 on Linux.

- ➔ **STEP 1:** Download the allinone package for Ns2 from http://sourceforge.net/projects/nsnam/files/latest/download


Home / Browse / nsnam
nsnam
Brought to you by: heidemann, tom_henderson
28 Reviews
Downloads: 2,174 This Week
Download
Get Updates
Share This

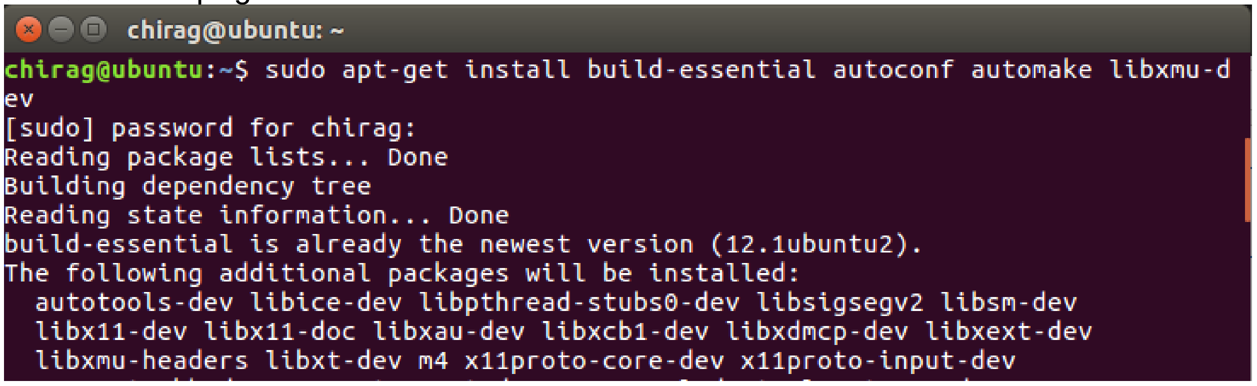
- ➔ The package downloaded will be named "ns-allinone-2.35.tar.gz". Copy it to the home folder. Then execute the following two commands in the terminal
    - ◆ cd ~/
    - ◆ tar -xvzf ns-allinone-2.35.tar.gz



```
chirag@ubuntu: ~
chirag@ubuntu:~$ tar -xvzf ns-allinone-2.35.tar.gz
ns-allinone-2.35/
ns-allinone-2.35/xgraph-12.2/
ns-allinone-2.35/xgraph-12.2/ps.c
ns-allinone-2.35/xgraph-12.2/configure.in
ns-allinone-2.35/xgraph-12.2/README.GENERAL
ns-allinone-2.35/xgraph-12.2/xgraph.c
ns-allinone-2.35/xgraph-12.2/Makefile.in
ns-allinone-2.35/xgraph-12.2/autoconf.h.in~
ns-allinone-2.35/xgraph-12.2/init.c
```

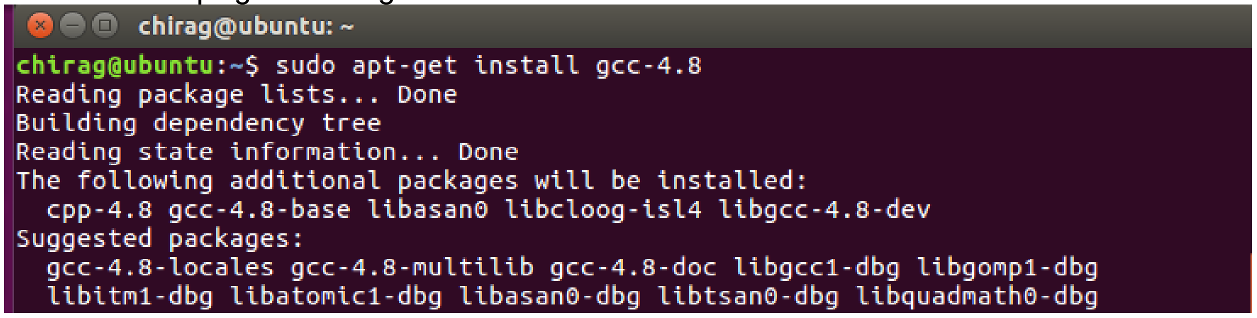
- ➔ All the files will be extracted into a folder by the name "**ns-allinone-2.35**".

- ➔ **STEP 2:** Ns2 does require some packages that have to be pre installed. Also Ns2 requires a gcc-4.8 compiler for proper execution. These can be installed by using the following commands.
  - ◆ sudo apt-get install build-essential autoconf automake libxmu-dev



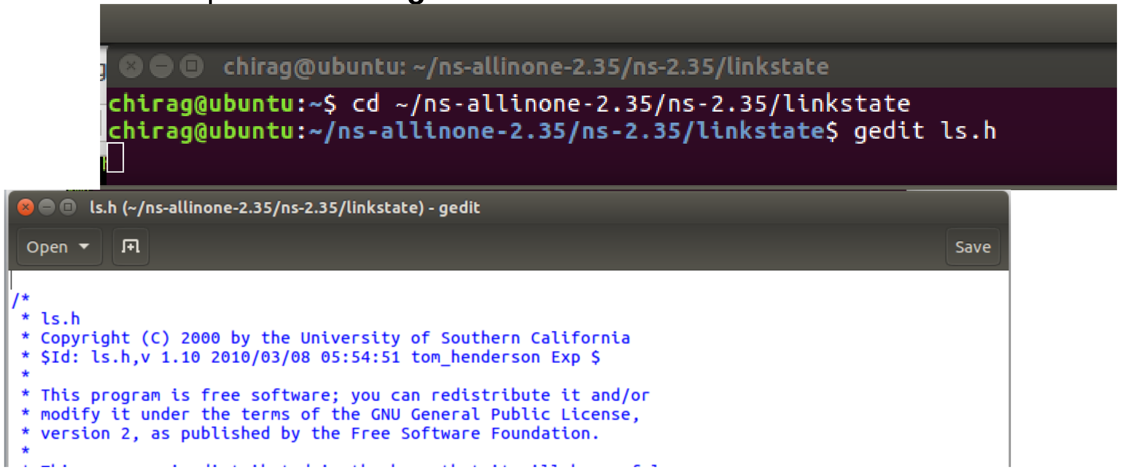
```
chirag@ubuntu: ~
chirag@ubuntu:~$ sudo apt-get install build-essential autoconf automake libxmu-d
ev
[sudo] password for chirag:
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
The following additional packages will be installed:
  autotools-dev libice-dev libpthread-stubs0-dev libsigsegv2 libsm-dev
  libx11-dev libx11-doc libxau-dev libxcb1-dev libxdmcp-dev libxext-dev
  libxmu-headers libxt-dev m4 x11proto-core-dev x11proto-input-dev
```

  - ◆ sudo apt-get install gcc-4.8



```
chirag@ubuntu: ~
chirag@ubuntu:~$ sudo apt-get install gcc-4.8
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cpp-4.8 gcc-4.8-base libasan0 libcloog-isl4 libgcc-4.8-dev
Suggested packages:
  gcc-4.8-locales gcc-4.8-multilib gcc-4.8-doc libgcc1-dbg libgomp1-dbg
  libitm1-dbg libatomic1-dbg libasan0-dbg libtsan0-dbg libquadmath0-dbg
```

- ➔ **STEP 3:** Once the installation is over , we have to make a change in the "ls.h" file. So navigate to the ls.h file in linkstate folder using the following commands
  - ◆ cd ~/ns-allinone-2.35/ns-2.35/linkstate
- ➔ Once inside the folder, we need to open the ls.h file. So use the following command to open the file in **gedit** file editor



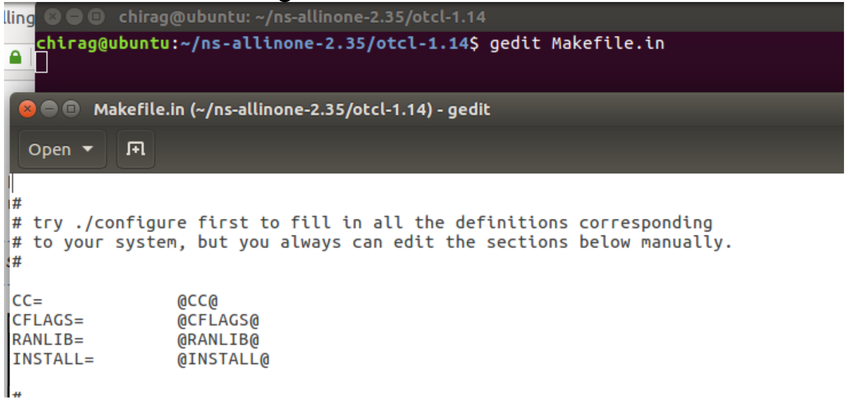
```
chirag@ubuntu: ~/ns-allinone-2.35/ns-2.35/linkstate
chirag@ubuntu:~$ cd ~/ns-allinone-2.35/ns-2.35/linkstate
chirag@ubuntu:~/ns-allinone-2.35/ns-2.35/linkstate$ gedit ls.h
```

```
/*
 * ls.h
 * Copyright (C) 2000 by the University of Southern California
 * $Id: ls.h,v 1.10 2010/03/08 05:54:51 tom_henderson Exp $
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License,
 * version 2, as published by the Free Software Foundation.
 *
```

- ➔ Now change **"erase"** on line 137 to **"this->erase"**.

```
        // 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);
        }
;
```

- Save and close the file.
- **STEP 4:** We also need the Ns to know which version of GCC are we using and so we need to make a small change in **"Makefile.in"**
- So first we have to go to the Makefile.in which is in ns-allinone-2.34/otcl-1.13

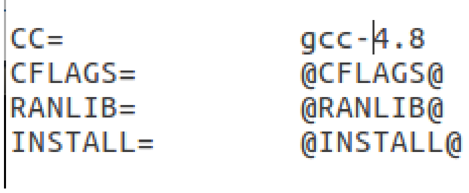


```
chirag@ubuntu:~/ns-allinone-2.35/otcl-1.14$ gedit Makefile.in
```

Makefile.in (~/ns-allinone-2.35/otcl-1.14) - gedit

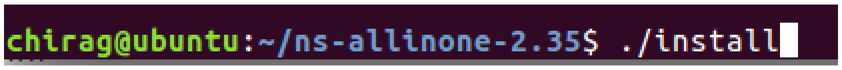
```
#
# try ./configure first to fill in all the definitions corresponding
# to your system, but you always can edit the sections below manually.
#

CC=             @CC@
CFLAGS=         @CFLAGS@
RANLIB=         @RANLIB@
INSTALL=        @INSTALL@
```

- In the file, change CC= @CC@ to CC=gcc-4.8, as shown in the image below.



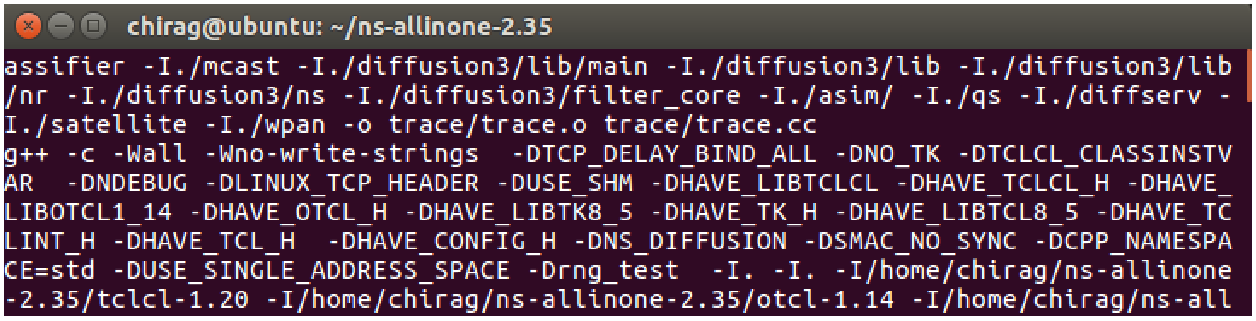
```
CC=             gcc-4.8
CFLAGS=         @CFLAGS@
RANLIB=         @RANLIB@
INSTALL=        @INSTALL@
```

- **STEP 5:** Finally we have to install Ns2. So navigate to the install file using the following command.
    - cd ~/ns-allinone-2.35/./install



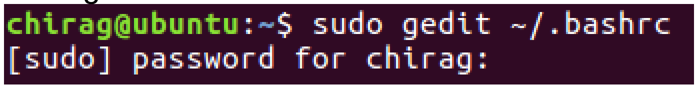
```
chirag@ubuntu:~/ns-allinone-2.35$ ./install
```

- The installation process begins and might take a while to complete.



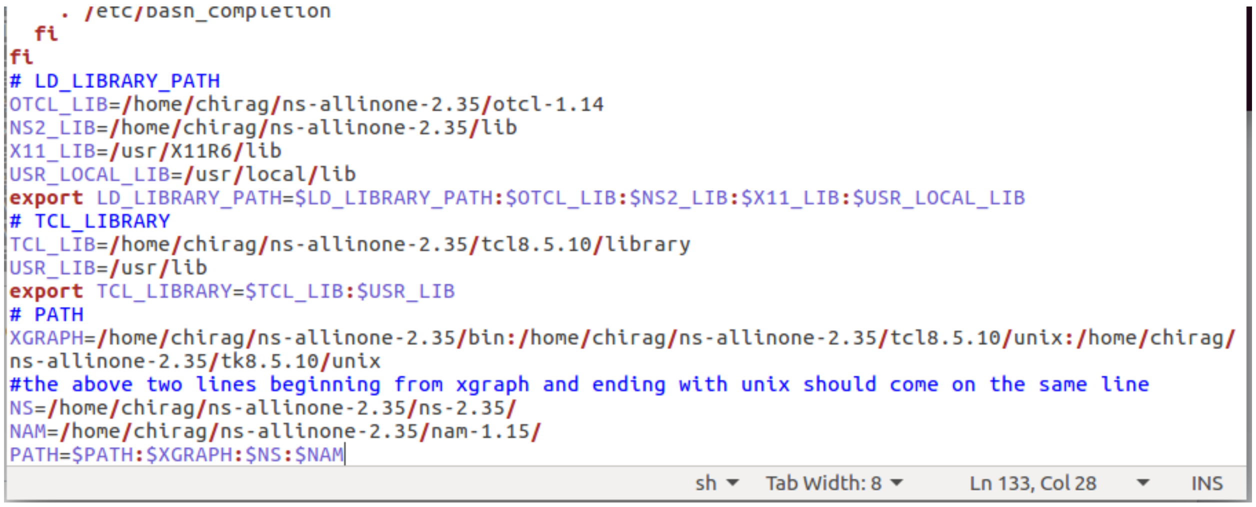
```
assifier -I./mcast -I./diffusion3/lib/main -I./diffusion3/lib -I./diffusion3/lib
/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -
I./satellite -I./wpan -o trace/trace.o trace/trace.cc
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTV
AR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL_H -DHAVE_
LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TC
LINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_NO_SYNC -DCPP_NAMESPA
CE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/chirag/ns-allinone
-2.35/tclcl-1.20 -I/home/chirag/ns-allinone-2.35/otcl-1.14 -I/home/chirag/ns-all
```

➔ **STEP 6:** Setting the environment path is the last step in the installation of Ns2.
➔ For this we need to add something in the **bashrc** file. So first open the file in gedit using the following command.
  ◆ sudo gedit ~/.bashrc



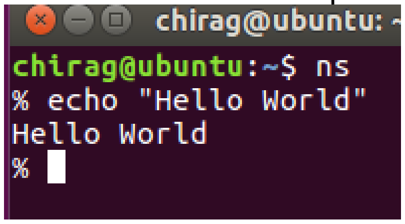
```
chirag@ubuntu:~$ sudo gedit ~/.bashrc
[sudo] password for chirag:
```

➔ Now we have to add the following lines in the end of the code.
  ◆ # LD_LIBRARY_PATH
OTCL_LIB=/home/chirag/ns-allinone-2.35/otcl-1.14
NS2_LIB=/home/chirag/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:$USR_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/home/chirag/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/home/chirag/ns-allinone-2.35/bin:/home/chirag/ns-allinone-2.35/tcl8.5.10/unix:/home/chirag/ns-allinone-2.35/tk8.5.10/unix
#the above two lines beginning from xgraph and ending with unix should come on the same line
NS=/home/chirag/ns-allinone-2.35/ns-2.35/
NAM=/home/chirag/ns-allinone-2.35/nam-1.15/
PATH=$PATH:$XGRAPH:$NS:$NAM



```
    . /etc/bash_completion
  fi
fi
# LD_LIBRARY_PATH
OTCL_LIB=/home/chirag/ns-allinone-2.35/otcl-1.14
NS2_LIB=/home/chirag/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:$USR_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/home/chirag/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/home/chirag/ns-allinone-2.35/bin:/home/chirag/ns-allinone-2.35/tcl8.5.10/unix:/home/chirag/
ns-allinone-2.35/tk8.5.10/unix
#the above two lines beginning from xgraph and ending with unix should come on the same line
NS=/home/chirag/ns-allinone-2.35/ns-2.35/
NAM=/home/chirag/ns-allinone-2.35/nam-1.15/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

➔ **STEP 7:** Finally, we check if Ns2 is installed correctly or not. For this we run Ns2 by using the following command.
  - ns
  - You can also run a simple command to be sure.



```
chirag@ubuntu:~$ ns
% echo "Hello World"
Hello World
% 
```

Conclusion: Thus Ns2 was successfully downloaded, extracted, installed and checked.

EXPERIMENT 3

AIM :

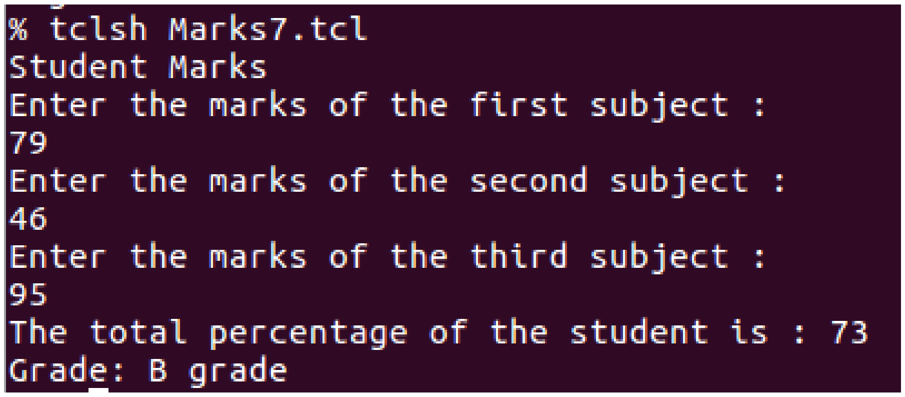
1. Write a script to read the marks of 3 subjects of a student and display his or her grade.
2. Write a script to display all the prime numbers in a given range.
3. Write a script to find the factorial of a number (Use procedures).

1] Write a script to read the marks of 3 subjects of a student and display his or her grade.

CODE:

```
puts "Student Marks"
puts "Enter the marks of the first subject :"
gets stdin marks1
puts "Enter the marks of the second subject :"
gets stdin marks2
puts "Enter the marks of the third subject :"
gets stdin marks3
set percent [expr ($marks1+$marks2+$marks3) /3 ]
puts "The total percentage of the student is : $percent"
if {$percent >= 80} {
   puts "Grade: A grade"
} elseif {$percent>=60 && $percent<80} {
   puts "Grade: B grade"
} elseif {$percent>=40 && $percent<60} {
   puts "Grade: C grade"
} else {
   puts "Fail"
}
```

OUTPUT:



```
% tclsh Marks7.tcl
Student Marks
Enter the marks of the first subject :
79
Enter the marks of the second subject :
46
Enter the marks of the third subject :
95
The total percentage of the student is : 73
Grade: B grade
```

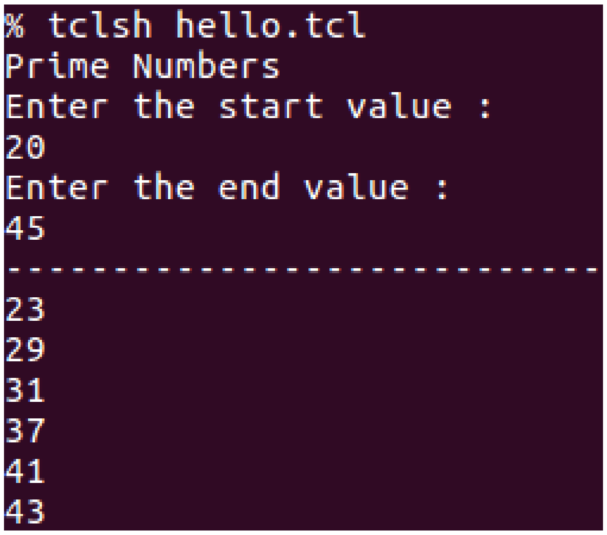
2] Write a script to display all the prime numbers in a given range.

CODE:

```
puts "Prime Numbers"
puts "Enter the start value :"
gets stdin start
```

```
puts "Enter the end value :"
gets stdin end
puts "----------------------------------"

for {set x $start} {$x<$end} {incr x} {
        set flag 1
        for {set i 2} {$i<$x} {incr i} {
        set y [expr $x % $i]
        if {$y == 0}  {
        #puts "$x: not a prime no"
        set flag 0
        break
        }
        }

        if {$flag ==1} {
        puts "$x"
        }
}
```

OUTPUT:



```
% tclsh hello.tcl
Prime Numbers
Enter the start value :
20
Enter the end value :
45
----------------------------
23
29
31
37
41
43
```

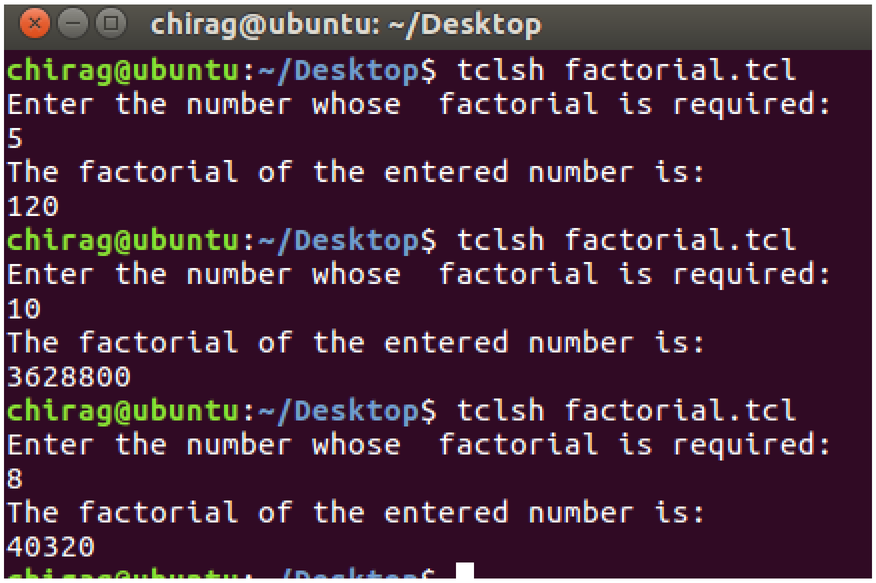
3] Write a script to find the factorial of a number (Use procedures).

CODE:

```
proc fact {n} {
        set result 1
        for { set i 1 } { $i <= $n } { incr i } {
                set result [expr {$result * $i}]
        }
        return $result
        puts $result
}
```

```
puts "Enter the number whose  factorial is required: "
gets stdin data
puts "The factorial of the entered number is:"
puts [fact $data]
```

OUTPUT:



```
chirag@ubuntu:~/Desktop$ tclsh factorial.tcl
Enter the number whose  factorial is required:
5
The factorial of the entered number is:
120
chirag@ubuntu:~/Desktop$ tclsh factorial.tcl
Enter the number whose  factorial is required:
10
The factorial of the entered number is:
3628800
chirag@ubuntu:~/Desktop$ tclsh factorial.tcl
Enter the number whose  factorial is required:
8
The factorial of the entered number is:
40320
```

EXPERIMENT 4

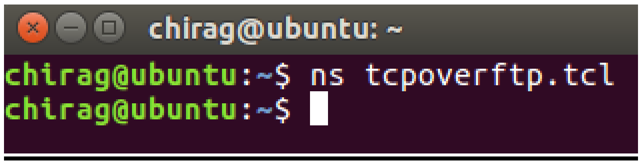
AIM: To create a TCP connection over FTP.

CODE:

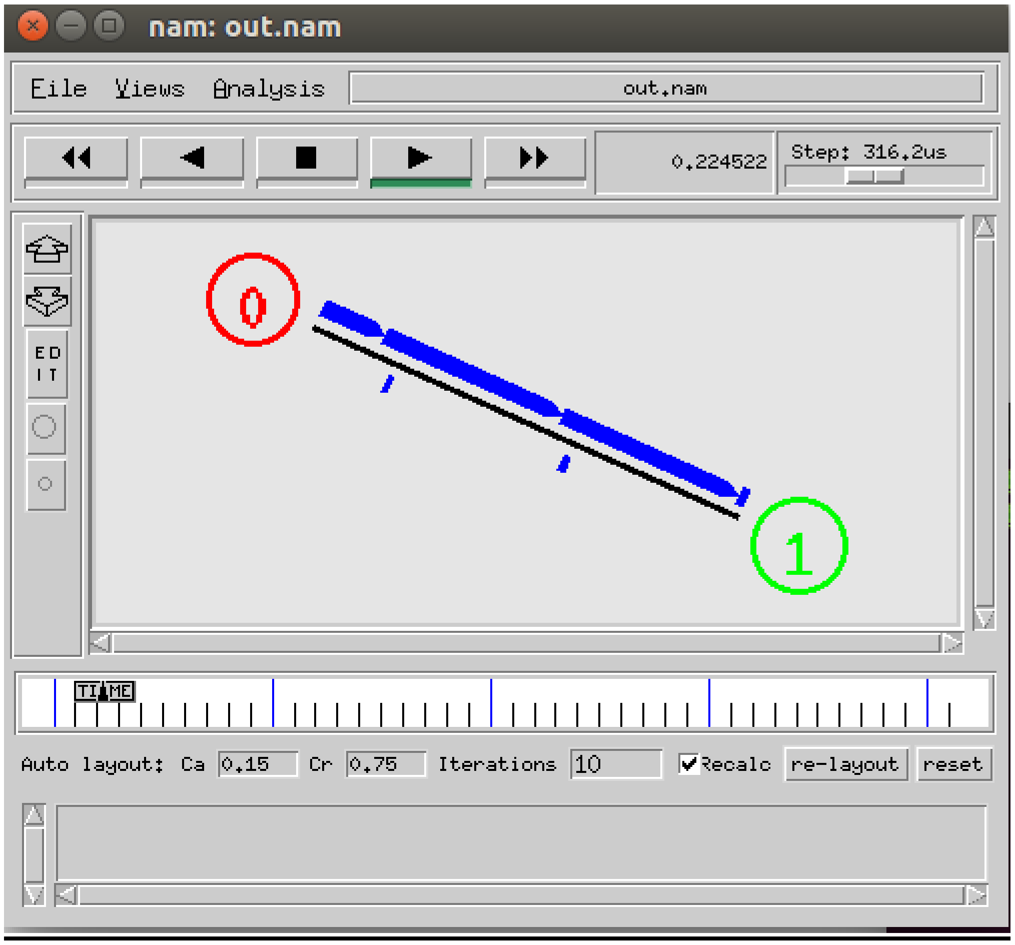
```
#Create a simulator object (God Object)
set ns [new Simulator]

#Set different colours for data flows
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define the finish procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on trace file
        exec nam out.nam &
        exit 0
}

#Create 2 nodes
set n0 [$ns node]
set n1 [$ns node]
$n0 color red
$n1 color green

#Create the links between the nodes
$ns duplex-link $n0 $n1 2Mb 10ms DropTail

#Setup a TCP Connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
```

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Schedule the events for FTP connection
$ns at 0.1 "$ftp start"
$ns at 4.0 "$ftp stop"

#Call the finish procedure
$ns at 4.2 "finish"

#Run the simulation
$ns run
```

OUTPUT :



```
chirag@ubuntu:~$ ns tcpoverftp.tcl
chirag@ubuntu:~$ 
```


nam: out.nam
File Views Analysis
out.nam
0.224522
Step: 316.2us
EDIT
0
1
TIME
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc re-layout reset

EXPERIMENT 5

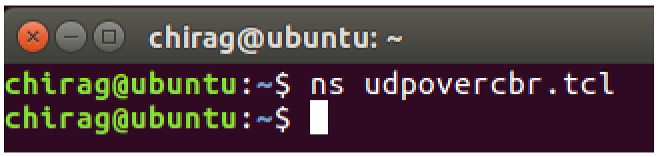
AIM: To create a UDP connection over CBR.

CODE:

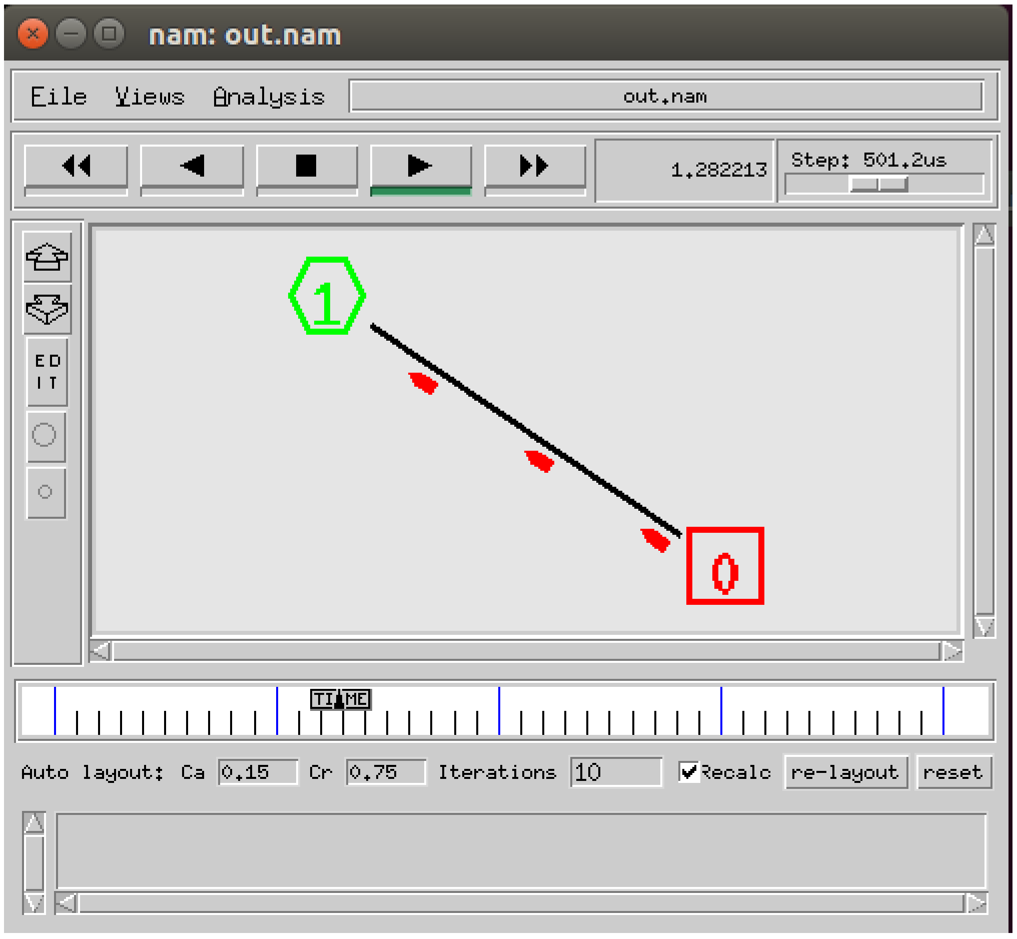
```
#Create a simulator object (God Object)
set ns [new Simulator]

#Set different colours for data flows
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define the finish procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on trace file
        exec nam out.nam &
        exit 0
}

#Create 2 nodes
set n0 [$ns node]
set n1 [$ns node]
$n0 color red
$n1 color green

#Create the links between the nodes=======
$ns duplex-link $n0 $n1 2Mb 10ms DropTail

#Set the shapes of the nodes
$n0 shape square
$n1 shape hexagon

#Setup a UDP Connection
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n1 $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
$udp set type_ CBR

#Schedule the events for FTP connection
$ns at 0.1 "$cbr start"
$ns at 4.0 "$cbr stop"

#Call the finish procedure
$ns at 4.2 "finish"

#Run the simulation
$ns run
```

OUTPUT:



```
chirag@ubuntu:~$ ns udpovercbr.tcl
chirag@ubuntu:~$
```


nam: out.nam
File Views Analysis
out.nam
1.282213
Step: 501.2us
EDIT
1
0
TIME
Auto layout: Ca 0.15 Cr 0.75 Iterations 10
Recalc
re-layout
reset

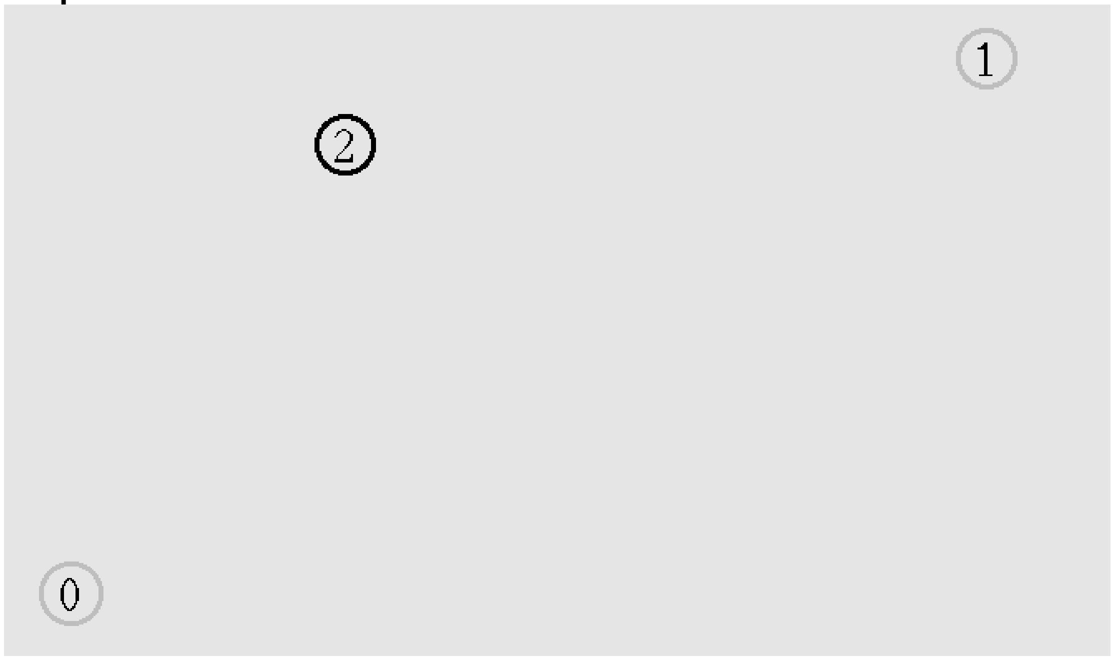
<u>**Experiment 6**</u>

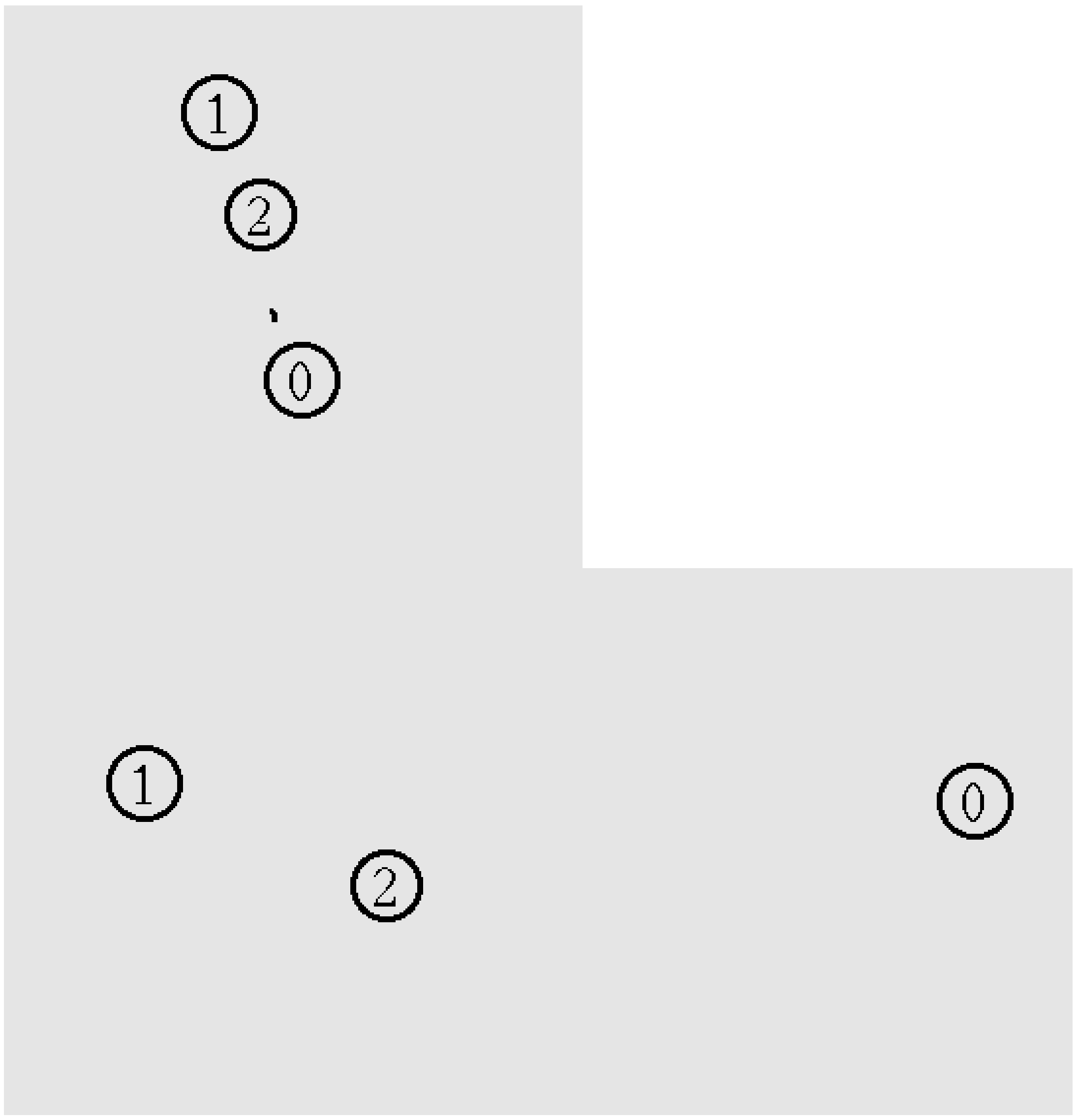
Aim: To study wireless connections between three wireless nodes.

Code :

```
# 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 mobilenodes
set val(rp) DSDV ;# 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

set ns [new Simulator]

set tracefd [open simple-dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace [open simwrls.nam w]

$ns trace-all $tracefd
$ns use-newtrace
$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 $val(nn)
# Create nn mobilenodes [$val(nn)] and attach them to the channel.
# 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]
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"

# 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 simwrls.nam &
exit
}
$ns run
```

Output:




1
2
0
1
2
0

Experiment No: 7

Aim: Analysis of network performance for QoS parameters

Theory : The network performance is measured with quality of service (QoS) parameters.
The QoS parameters include

1. Packet-to- delivery ratio : expressed as PDR=(no of packets received/ No of packets sent)
2. Throughput: Bytes received/ bytes transferred.

The QoS parameters are extracted from the trace file generated after executing the tcl script,
by applying an awk script on it.
Awk
An awk is a programmable, pattern-matching, and processing tool available in UNIX. awk is
not just a command, but a programming language too. It is a pattern scanning and processing
language. It searches one or more files to see if they contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or
incrementing a counter each time it finds a match.
Syntax: awk option 'selection_criteria {action}' file(s)
Here, selection_criteria filters input and select lines for the action component to act upon.
The selection_criteria is enclosed within single quotes and the action within the curly braces.
Both the selection_criteria and action forms an awk program.
Example: $ awk „/manager/ {print}‟ emp.lst
Variables Awk allows the user to use variables of their choice. You can now print a serial
number, using the variable kount, and apply it those directors drawing a salary exceeding
6700:
awk $3 == "director" &amp;&amp; $6 &gt; 6700 { count =count+1 printf " %f %s %s %d\n",
kount,$2,$3,$6 }‟ empn.lst

–f Option: Storing Awk Programs in a File
A large awk programs can be stored in separate file and provide them with the awk extension for easier identification.
You can now use awk with the –f filename option to obtain the same output:

awk –f empawk.awk empn.lst

The begin and end sections

Awk statements are usually applied to all lines selected by the address, and if there are no
addresses, then they are applied to every line of input. But, if you have to print something
before processing the first line, for example, a heading, then the BEGIN section can be used
gainfully. Similarly, the end section useful in printing some totals after processing is over.
The BEGIN and END sections are optional and take the form
BEGIN {action} END {action}

These two sections, when present, are delimited by the body of the awk program. You can
use them to print a suitable heading at the beginning and the average salary at the end.

Steps to create and execute tcl script:
Step 1: Open any text editor (vi, nano)
Step 2: Write the program using ns2 tcl script and save with extension as filename.tcl
Step 3: Execute tcl script as “ns filename.tcl”
Step 4: Print the packet-to- delivery ratio by executing following command:
“ awk –f pdr.awk simple-dsdv.tr ”
Step 5: Print the thruoghput by executing following command:
“ awk –f thru.awk simple-dsdv.tr”
Step 6: Observe the output.

Code:

1.
```
# 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 mobilenodes
set val(rp) DSDV ;# 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
set ns [new Simulator]
set tracefd [open simple-dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace [open simwrls.nam w]
$ns trace-all $tracefd
$ns use-newtrace
```

```
$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)

#
# Create nn mobilenodes [$val(nn)] and attach them to the channel.
#
# 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 &lt; $val(nn) } { incr i } {
set node_($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
$ns at 10.0 &quot;$node_(0) setdest 250.0 250.0 3.0&quot;
$ns at 15.0 &quot;$node_(1) setdest 45.0 285.0 5.0&quot;
$ns at 110.0 &quot;$node_(0) setdest 480.0 300.0 5.0&quot;

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
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

$ns at 10.0 &quot;$ftp start&quot;

# Define node initial position in nam
for {set i 0} {$i &lt; $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 &lt; $val(nn) } { incr i } {
$ns at $val(stop) &quot;$node_($i) reset&quot;;
}
# ending nam and the simulation
$ns at $val(stop) &quot;$ns nam-end- wireless $val(stop)&quot;
$ns at $val(stop) &quot;stop&quot;
$ns at 150.01 &quot;puts \&quot;end simulation\&quot; ; $ns halt&quot;
proc stop {} {
global ns tracefd namtrace
$ns flush-trace
close $tracefd
close $namtrace
exec nam simwrls.nam &amp;
exit
}
$ns run
```

2. Packet to delivery ratio : pdr.awk

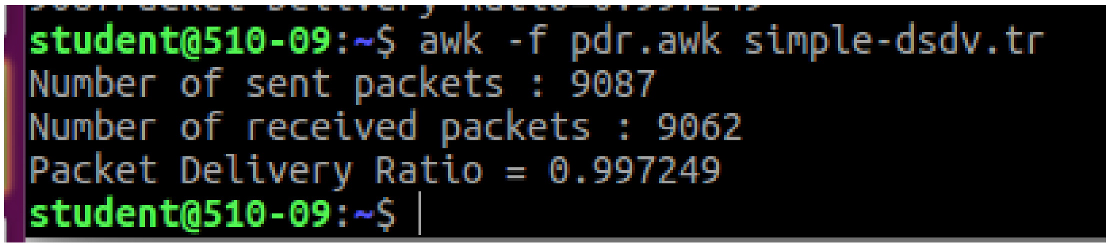
```
BEGIN{ sendLine = 0; recvLine = 0; }

# count packets sent
# $0 is the whole line of arguments
$0 ~/^s.* AGT/ { sendLine ++ ; }
# count packets received
$0 ~/^r.* AGT/ { recvLine ++ ; }

END {
# calculate the PDR
printf(&quot;\nPacket Delivery Ratio:%.4f\n&quot;, (recvLine/sendLine));
}
```

3. Throughput awk script : thru.awk

```
BEGIN {
recvdSize = 0 # received packet size

startTime = 400 # high random start time
stopTime = 5 #low random stop time
}
{# from the trace file obtain…..
event = $1 # send or received
time = $3 # time of transaction
pkt_size = $37 # packet size
level = $19 # application agent or routing protocol data
# Find the starting time of simulation
if (level == &quot;AGT&quot; &amp;&amp; event == &quot;s&quot; ){
if (time &lt; startTime){
startTime = time;
}
}
# Update total received packets' size and store packets arrival time, to finally get the
# end time of the simulation
if (level == &quot;AGT&quot; &amp;&amp; event == &quot;r&quot; ){
if (time &gt; stopTime){
stopTime = time;
}
recvdSize += pkt_size;
}
}
END{
# calculate the throughput
printf(&quot;Average Throughput[kbps] = %.2f\n&quot;,(recvdSize/(stopTime-
startTime)))
}
```

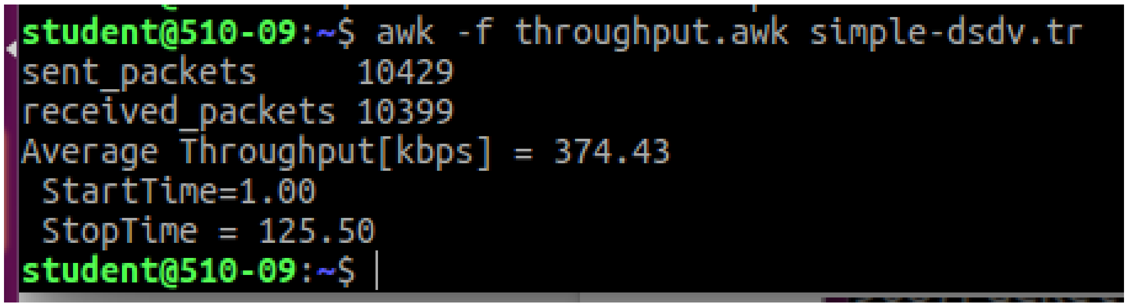
Output:

PDR –



```
student@510-09:~$ awk -f pdr.awk simple-dsdv.tr
Number of sent packets : 9087
Number of received packets : 9062
Packet Delivery Ratio = 0.997249
student@510-09:~$
```

Throughput-



```
student@510-09:~$ awk -f throughput.awk simple-dsdv.tr
sent_packets     10429
received_packets 10399
Average Throughput[kbps] = 374.43
 StartTime=1.00
 StopTime = 125.50
student@510-09:~$
```

Conclusion: Thus we have found analysis for the QoS parameters of a network using AWK script.

EXPERIMENT 8

AIM: To execute TCP Socket Program.

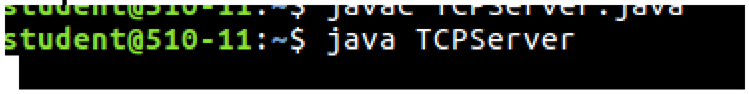
Client:
Program:-

```
import java.io.*;

import java.net.*;
class TCPClient
{
public static void main(String argv[]) throws Exception
{
   String sentence;
   String modifiedSentence;
   BufferedReader inFromUser =new BufferedReader(new
InputStreamReader(System.in));

   Socket clientSocket = new Socket("192.168.107.48", 9080);
        //creating an object of socket

   DataOutputStream outToServer =new
DataOutputStream(clientSocket.getOutputStream());

   BufferedReader inFromServer =new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        //for getting the modified string from the server

   sentence = inFromUser.readLine(); //reads input string
   outToServer.writeBytes(sentence + '\n'); //sends the string to the server
   modifiedSentence = inFromServer.readLine(); //gets the modified string from the
server
   System.out.println("FROM SERVER: " + modifiedSentence); //modified string is
printed
   clientSocket.close();
}
}
```

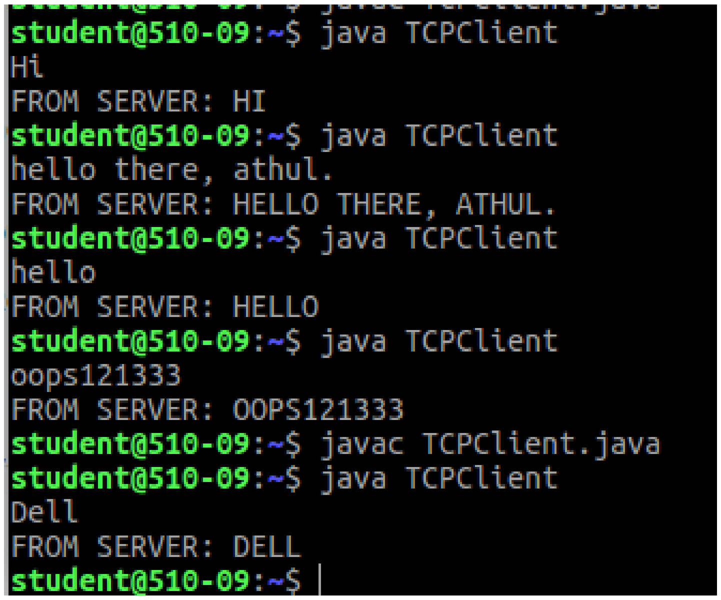
Server:
Program:-

```
import java.io.*;
import java.net.*;
class TCPServer
{
public static void main(String argv[]) throws Exception
{
//String entererd by the client
String clientSentence;
//String sent by server after capitalization
String capitalizedSentence;
//Creating an object of socket
ServerSocket welcomeSocket = new ServerSocket(9080);
while(true)
{
//Accepting connection request from the client
Socket connectionSocket = welcomeSocket.accept();
//getting input from client
BufferedReader inFromClient =
new BufferedReader(
new InputStreamReader(
connectionSocket.getInputStream()));
//Providing output to the client
DataOutputStream outToClient =
new DataOutputStream(
connectionSocket.getOutputStream());
clientSentence = inFromClient.readLine();
capitalizedSentence = clientSentence.toUpperCase() + '\n';
outToClient.writeBytes(capitalizedSentence);
}
}
}
```

Output:-



```
student@510-11:~$ javac TCPServer.java
student@510-11:~$ java TCPServer
```



```
student@510-09:~$ java TCPClient
Hi
FROM SERVER: HI
student@510-09:~$ java TCPClient
hello there, athul.
FROM SERVER: HELLO THERE, ATHUL.
student@510-09:~$ java TCPClient
hello
FROM SERVER: HELLO
student@510-09:~$ java TCPClient
oops121333
FROM SERVER: OOPS121333
student@510-09:~$ javac TCPClient.java
student@510-09:~$ java TCPClient
Dell
FROM SERVER: DELL
student@510-09:~$
```

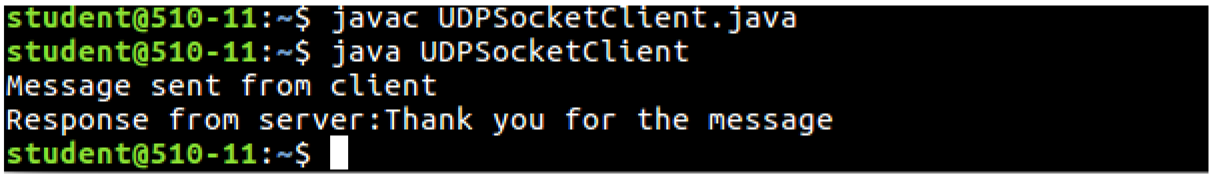
CONCLUSION:
Thus TCP Client Server connection was studied successfully.

EXPERIMENT 9

Aim: To study UDP Client Server connections

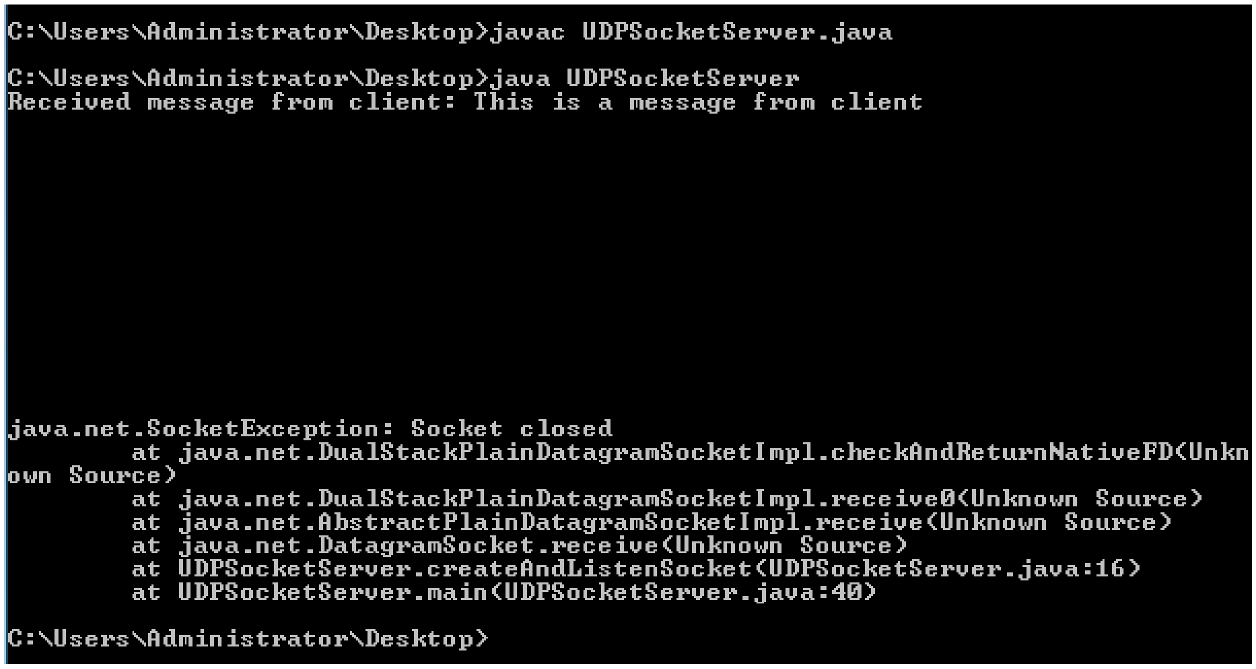
CODE :
Client:

```
import java.io.IOException;
import java.net.*;
public class UDPSocketClient {
DatagramSocket Socket;
public UDPSocketClient() {
}
public void createAndListenSocket() {
try {
Socket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName("192.168.105.30");
byte[] incomingData = new byte[1024];
String sentence = "This is a message from client";
byte[] data = sentence.getBytes();
DatagramPacket sendPacket = new DatagramPacket(data, data.length, IPAddress,
1024);
Socket.send(sendPacket);
System.out.println("Message sent from client");
DatagramPacket incomingPacket = new DatagramPacket(incomingData,
incomingData.length);
Socket.receive(incomingPacket);
String response = new String(incomingPacket.getData());
System.out.println("Response from server:" + response);
Socket.close();
} catch (UnknownHostException e) {
e.printStackTrace();
} catch (SocketException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}

}
public static void main(String[] args) {
UDPSocketClient client = new UDPSocketClient();
client.createAndListenSocket();
}
}
```


```
student@510-11:~$ javac UDPSocketClient.java
student@510-11:~$ java UDPSocketClient
Message sent from client
Response from server:Thank you for the message
student@510-11:~$
```

Server:

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
public class UDPSocketServer {
DatagramSocket socket = null;
public UDPSocketServer() {
}
public void createAndListenSocket() {
try {
socket = new DatagramSocket(1024);
byte[] incomingData = new byte[1024];
while (true) {
DatagramPacket incomingPacket = new DatagramPacket(incomingData,
incomingData.length);
socket.receive(incomingPacket);
String message = new String(incomingPacket.getData());
System.out.println("Received message from client: " + message);
InetAddress IPAddress = incomingPacket.getAddress();
int port = incomingPacket.getPort();
String reply = "Thank you for the message";
byte[] data = reply.getBytes();
DatagramPacket replyPacket =

new DatagramPacket(data, data.length, IPAddress, port);
socket.send(replyPacket);
Thread.sleep(2000);
socket.close();
}
} catch (SocketException e) {
e.printStackTrace();
} catch (IOException i) {
i.printStackTrace();
} catch (InterruptedException e) {
e.printStackTrace();
}
}
public static void main(String[] args) {
UDPSocketServer server = new UDPSocketServer();
server.createAndListenSocket();
}
}
```



```
C:\Users\Administrator\Desktop>javac UDPSocketServer.java

C:\Users\Administrator\Desktop>java UDPSocketServer
Received message from client: This is a message from client











java.net.SocketException: Socket closed
        at java.net.DualStackPlainDatagramSocketImpl.checkAndReturnNativeFD(Unkn
own Source)
        at java.net.DualStackPlainDatagramSocketImpl.receive0(Unknown Source)
        at java.net.AbstractPlainDatagramSocketImpl.receive(Unknown Source)
        at java.net.DatagramSocket.receive(Unknown Source)
        at UDPSocketServer.createAndListenSocket(UDPSocketServer.java:16)
        at UDPSocketServer.main(UDPSocketServer.java:40)

C:\Users\Administrator\Desktop>
```

Experiment No: 10

Aim : To analysis the packets using Wireshark.

Theory:

Packet Analysis Wireshark is an open source cross-platform packet capture and analysis tool, with versions for Windows and Linux. The GUI window gives a detailed breakdown of the network protocol stack for each packet, colorising packet details based on protocol, as well as having functionality to filter and search the traffic. Wireshark can also save packet data to files for offline analysis and export/import packet captures to/from other tools. Statistics can also be generated for packet capture files.

Wireshark can be used for network troubleshooting, to investigate security issues, and to analyse and understand network protocols. The packet sniffer can exploit information passed in plaintext, i.e. not encrypted. Examples of protocols which pass information in plaintext are Telnet, FTP, SNMP, POP, and HTTP.

Procedure

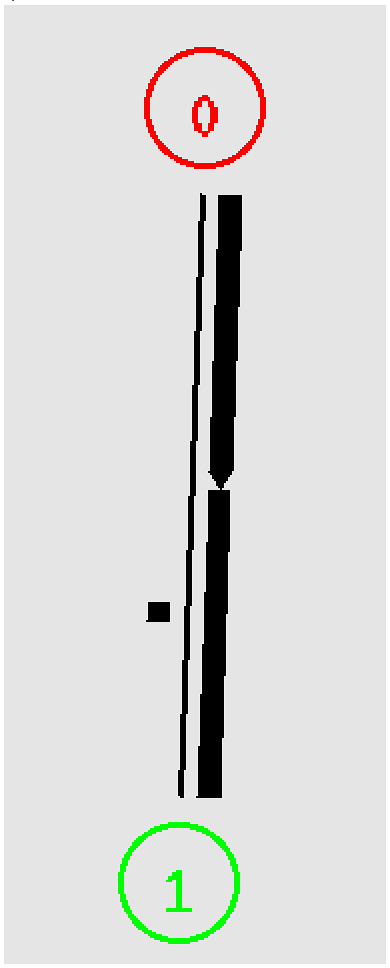
- Capture ICMP Protocol Traffic using Wireshark
  1. Start a Wireshark capture. Open a Windows console window, and generate some ICMP traffic by using the Ping command line tool to check the connectivity.
  2. The ICMP packets are difficult to pick out.
  3. Create a display filter to only show ICMP packets.
  4. By clicking on the trace, the packet details can be obtained as follows

Conclusion: Thus analysis the packets using Wireshark is done successfully.

ASSIGNMENT- 1

1)change colour of the node

```
set ns [new Simulator]

set n0 [$ns node]
$n0 color red

set n1 [$ns node]
$n1 color green

#Linking file handler for ns
set tr [open out.tr w]
$ns namtrace-all $tr

#linking file handler for nam trace file
set ftr [open "out.nam" w]
$ns namtrace-all $ftr

#setting the connection
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#creating agents and attaching them to nodes
set tcp1 [new Agent/TCP]
set sink [new Agent/TCPSink]

$ns attach-agent $n0 $tcp1
$ns attach-agent $n1 $sink

#initializing the connection
$ns connect $tcp1 $sink

#FTP Application that is going to be working on tcp/ip
set ftp [new Application/FTP]
$ftp attach-agent $tcp1
$ns at 0.1 "$ftp start"
$ns at 2.0 "$ftp stop"
$ns at 2.1 "finish"

proc finish {} {
 global ns tr ftr

 $ns flush-trace

 close $tr
 close $ftr

 exec nam out.nam &
 exit
}

$ns run
```



2)change the colour of the data path

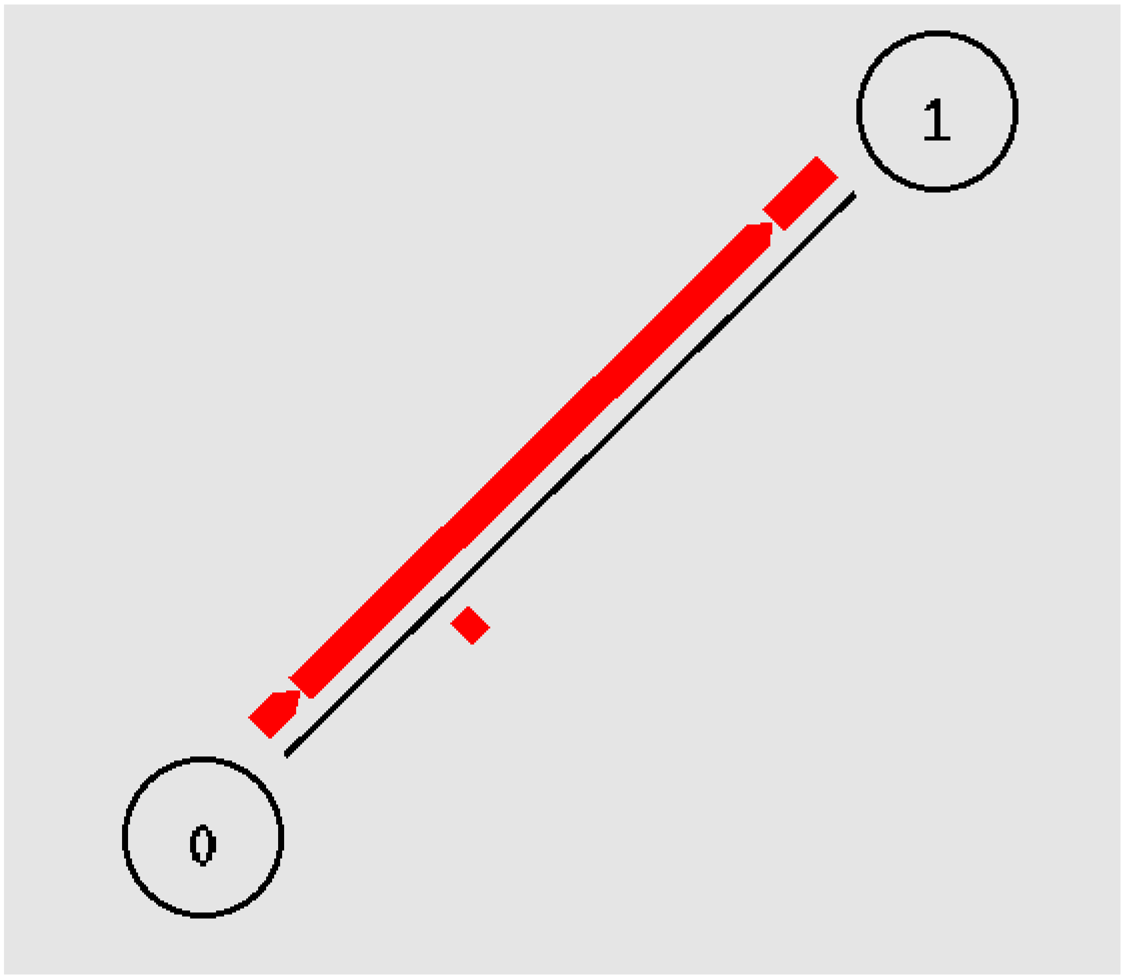
set ns [new Simulator]

$ns color n0 Blue
$ns color n1 Red

#Linking file handler for ns
set tr [open out.tr w]
$ns namtrace-all $tr

#linking file handler for nam trace file
set ftr [open "out.nam" w]
$ns namtrace-all $ftr

```
#creating nodes that will communicate with each other
set n0 [$ns node]
set n1 [$ns node]

#setting the connection
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#creating agents and attaching them to nodes
set tcp1 [new Agent/TCP]
set sink [new Agent/TCPSink]

$ns attach-agent $n0 $tcp1
$ns attach-agent $n1 $sink

#initializing the connection
$ns connect $tcp1 $sink

#FTP Application that is going to be working on tcp/ip
set ftp [new Application/FTP]
$ftp attach-agent $tcp1

$ns at 0.1 "$ftp start"
$ns at 2.0 "$ftp stop"
$ns at 2.1 "finish"

proc finish {} {
 global ns tr ftr

 $ns flush-trace

 close $tr
 close $ftr

 exec nam out.nam &
 exit
}

$ns run
```



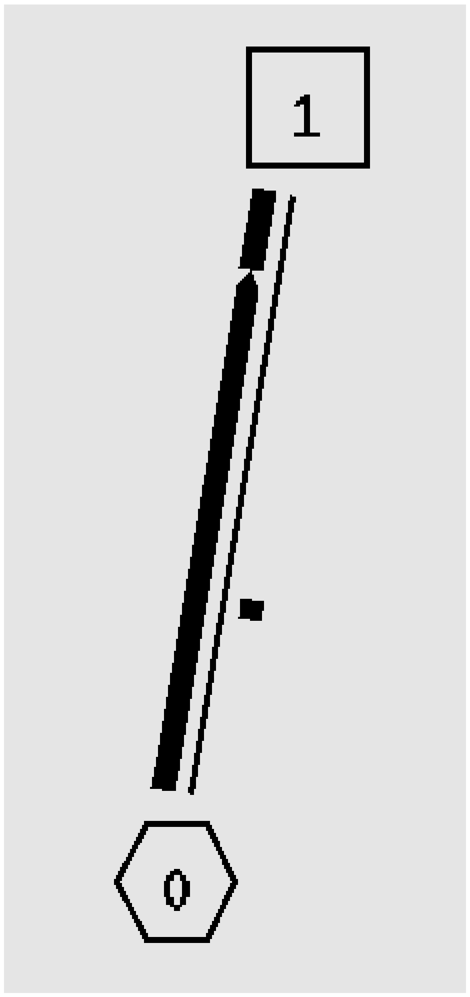
3)change shape of the node

```
set ns [new Simulator]

set n0 [$ns node]
```
$n0 shape hexagon

```
set n1 [$ns node]
```
$n1 shape box

```
#Linking file handler for ns
set tr [open out.tr w]
$ns namtrace-all $tr

#linking file handler for nam trace file
set ftr [open "out.nam" w]
$ns namtrace-all $ftr

#setting the connection
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

```
#creating agents and attaching them to nodes
set tcp1 [new Agent/TCP]
set sink [new Agent/TCPSink]

$ns attach-agent $n0 $tcp1
$ns attach-agent $n1 $sink

#initializing the connection
$ns connect $tcp1 $sink

#FTP Application that is going to be working on tcp/ip
set ftp [new Application/FTP]
$ftp attach-agent $tcp1
$ns at 0.1 "$ftp start"
$ns at 2.0 "$ftp stop"
$ns at 2.1 "finish"

proc finish {} {
 global ns tr ftr

 $ns flush-trace

 close $tr
 close $ftr

 exec nam out.nam &
 exit
}

$ns run
```

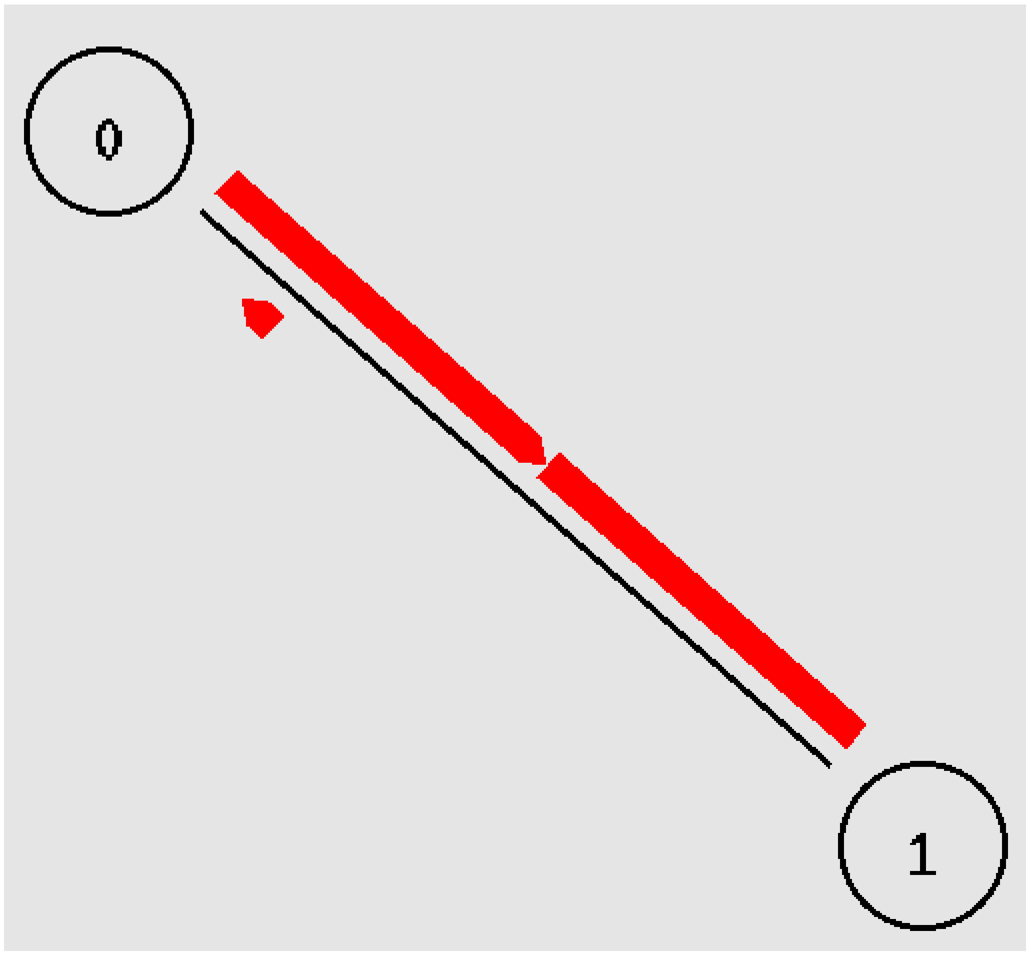


4)change the connection type(simplex and duplex a s well)

```
set ns [new Simulator]

$ns color n0 Blue
$ns color n1 Red

#Linking file handler for ns
set tr [open out.tr w]
$ns namtrace-all $tr

#linking file handler for nam trace file
set ftr [open "out.nam" w]
$ns namtrace-all $ftr

#creating nodes that will communicate with each other
set n0 [$ns node]
set n1 [$ns node]

#setting the connection
```

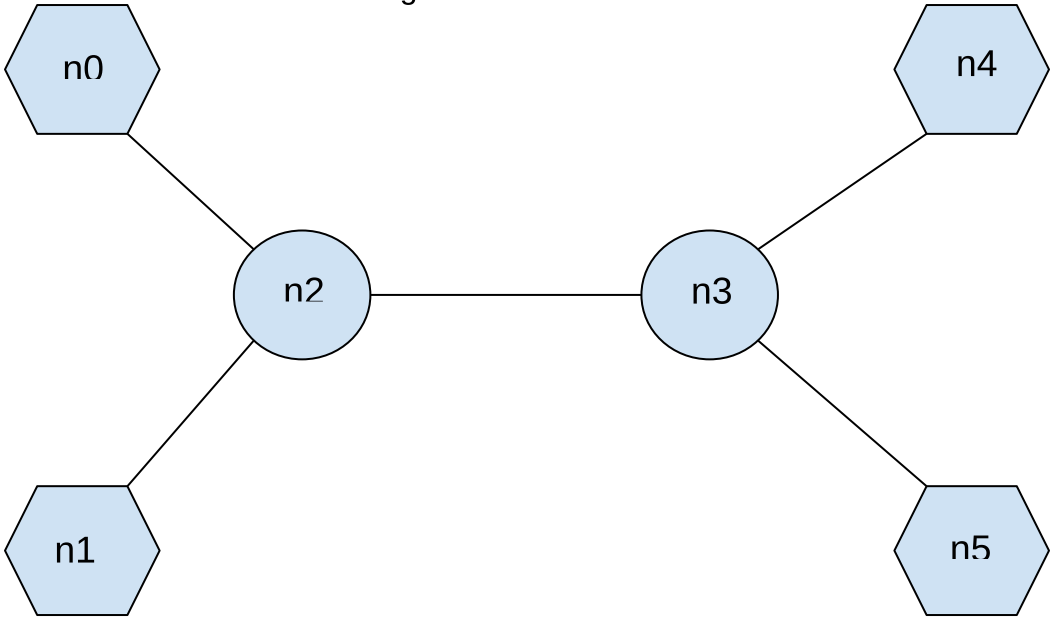
$ns simplex-link $n0 $n1 3Mb 2ms DropTail

```
$ns simplex-link $n1 $n0 2Mb 5ms DropTail

#creating agents and attaching them to nodes
set tcp1 [new Agent/TCP]
set sink [new Agent/TCPSink]

$ns attach-agent $n0 $tcp1
$ns attach-agent $n1 $sink

#initializing the connection
$ns connect $tcp1 $sink

#FTP Application that is going to be working on tcp/ip
set ftp [new Application/FTP]
$ftp attach-agent $tcp1

$ns at 0.1 "$ftp start"
$ns at 2.0 "$ftp stop"
$ns at 2.1 "finish"

proc finish {} {
 global ns tr ftr

 $ns flush-trace

 close $tr
 close $ftr

 exec nam out.nam &
 exit
}

$ns run
```


0
1

Assignment 2

Aim: Construct the following connection in NS2



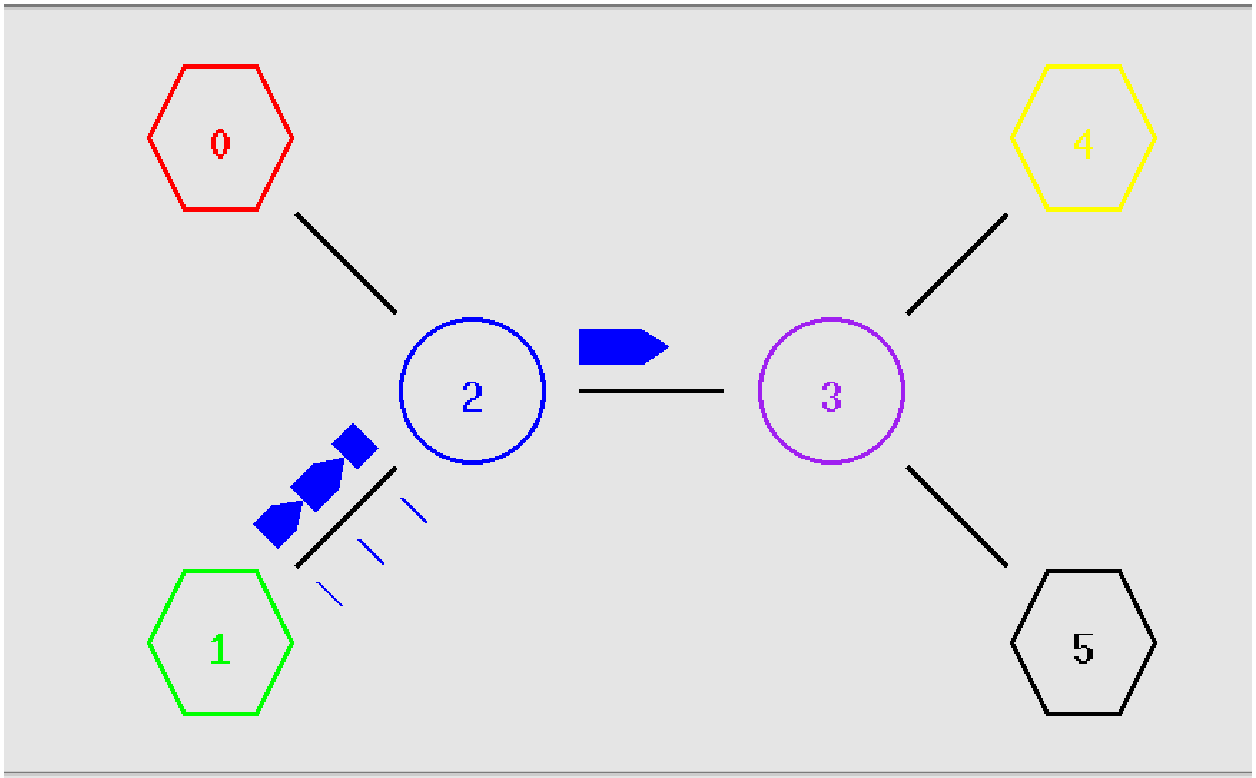
CODE :

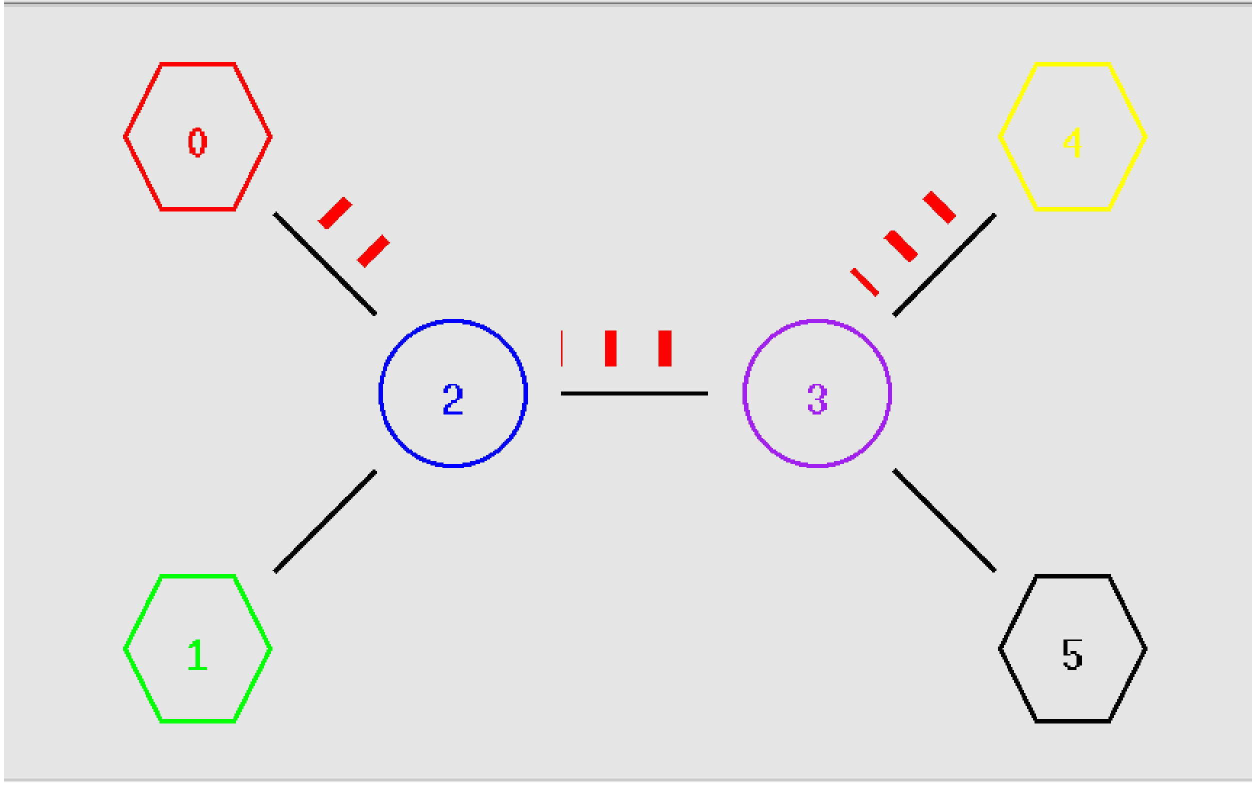
```
#Define the finish procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on trace file
        exec nam out.nam &
        exit 0
}

#Create a simulator object (God Object)
set ns [new Simulator]

#Set different colours for data flows
$ns color 1 Blue
$ns color 2 Red

set ftr [open out1.tr w]
$ns namtrace-all $ftr

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf
```

```
#Create 6 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 color red
$n1 color green
$n2 color blue
$n3 color purple
$n4 color yellow
$n5 color black


#Shapes of The Nodes
$n0 shape hexagon
$n1 shape hexagon
$n4 shape hexagon
$n5 shape hexagon

#Create the 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 2Mb 10ms DropTail
$ns duplex-link $n4 $n3 2Mb 10ms DropTail
$ns duplex-link $n5 $n3 2Mb 10ms DropTail

#Create duplex links
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n5 $n3 orient left-up
$ns duplex-link-op $n4 $n3 orient left-down
$ns duplex-link-op $n2 $n3 orient right

#Create a queue limit
$ns queue-limit $n2 $n3 10

#Setup a TCP Connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

```
#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 $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n4 $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
$udp set type_ CBR

#Schedule the events for FTP connection
$ns at 0.1 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.1 "$cbr start"
$ns at 8.0 "$cbr stop"

#Call the finish procedure
$ns at 8.2 "finish"

#Run the simulation
$ns run
```

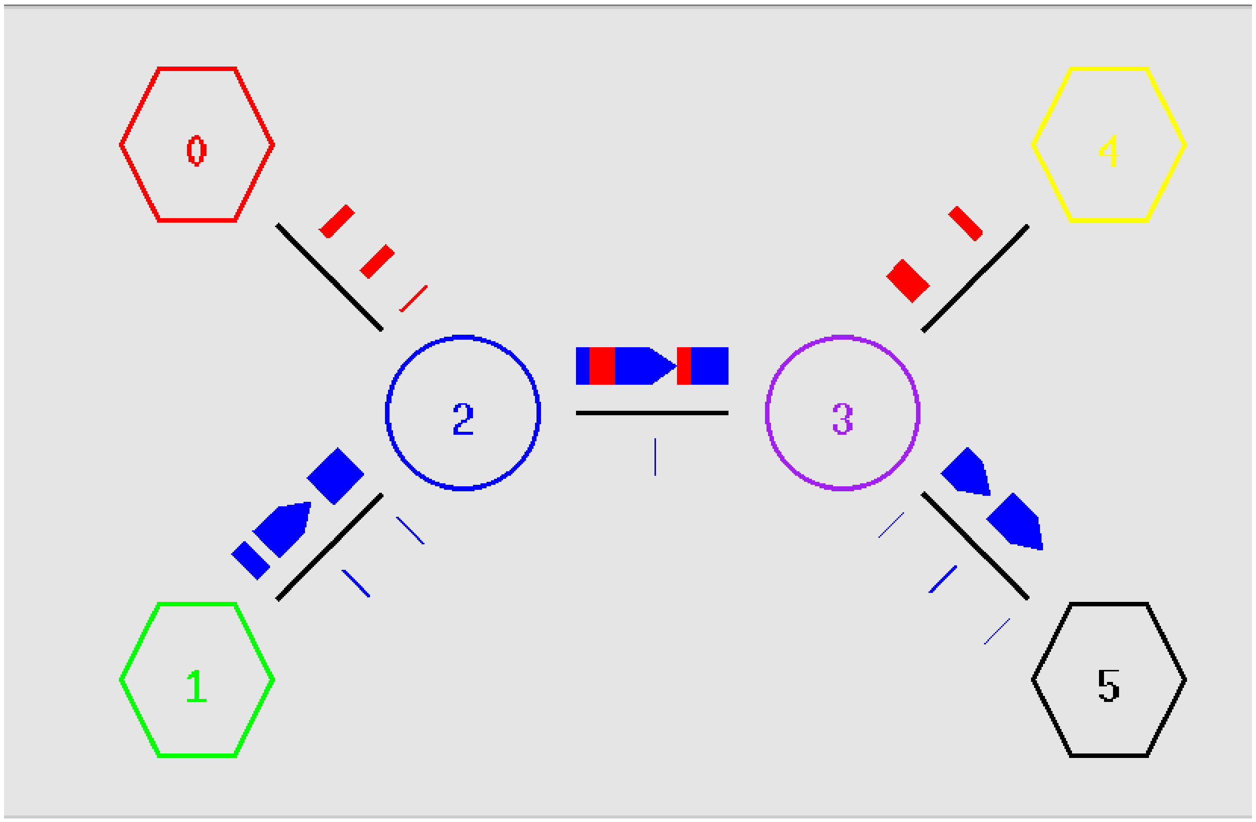
OUTPUT :


0
1
2
3
4
5


0
1
2
3
4
5

To make both the connections to work simultaneously we need to just chnage the start time of the CBR connections ie

```
#Schedule the events for FTP connection
$ns at 0.1 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 0.1 "$cbr start"
$ns at 4.0 "$cbr stop"

#Call the finish procedure
$ns at 4.2 "finish"
```



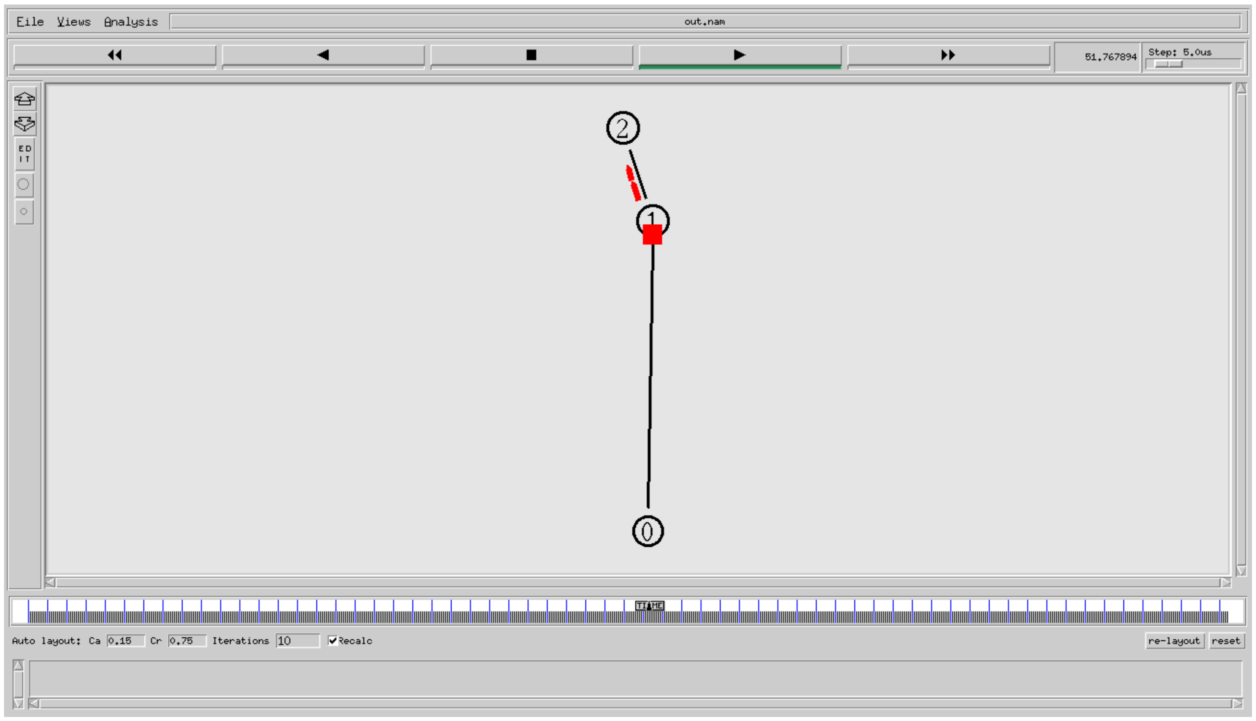
ASSIGNMENT 3

CODE :

```
#Create a simulator object
set ns [new Simulator]

$ns color 1 blue
$ns color 2 red

#Create Copy of Trace File
set tr [ open out.tr w]
$ns namtrace-all $tr

#Open the NAM trace file
set ftr [open out.nam w]
$ns namtrace-all $ftr

#Define a 'finish' procedure
proc finish {} {
        global ns tr ftr
        $ns flush-trace
        close $tr
        close $ftr
        exec nam out.nam &
        exit
}

#Create three nodes
set n1 [ $ns node ]
set n2 [ $ns node ]
set n3 [ $ns node ]

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp

set null [new Agent/Null]
$ns attach-agent $n2 $null

set null [new Agent/Null]
$ns attach-agent $n3 $null

$ns connect $udp $null
$udp set fid_ 2

#Create links between the nodes
$ns duplex-link $n1 $n2 100mb 5ms DropTail
$ns duplex-link $n2 $n3 5mb 5ms DropTail
```

```
$ns queue-limit $n1 $n2 10
$ns queue-limit $n2 $n3 4

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

#assigning constant size of packets
$cbr set packetSize_ 10000

#rate_   sending rate.
#interval_ (optional) interval between packets.
#random_ whether or not to introduce random noise in the scheduled
                   departure times. defualt is off.
#maxpkts_ maximum number of packets to send.

$cbr set random_ 1
puts "CBR random value [$cbr set random_ ]"

#Schedule events for the  agents
$ns at 0.1 "$cbr start"
$ns at 121.0 "$cbr stop"

#Call the finish procedure after 100.0 seconds of simulation time
$ns at 100.0 "finish"

#Run the simulation
$ns run
```

Output:



CONCLUSION :
Packet Drop and Congestion at the node can be avoided by decreasing the packet size, Increasing Bandwidth, increasing Queue limit and setting the random. Also the maximum number of packets that hasto transferred can be fixed.

---xxxxxxxxxxxxxxxxxxxxxxxxx---------------------------------------