Configuration Manual

MSc Research Project
MSc in Cloud Computing

Shantanu Dileep Paranjpe
Student ID: x19115644

School of Computing
National College of Ireland

Supervisor: Mr. Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shantanu Dileep Paranjpe
Student ID:	x19115644
Programme:	MSc in Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Mr. Vikas Sahni
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual
Word Count:	1994
Page Count:	17

Signature:	
Date:	16th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shantanu Dileep Paranjpe
x19115644

1 Introduction

Configuration manual is a document that provides all the details related to the downloading, installation, and setup details. In this configuration manual, the steps for installing the NS2 simulator, the detailed steps for downloading and installing the NSG (Network Scenario Generator) tool along with the xgraph tool are provided in detail. The manual is split into 3 sections in which section 1 provides information related to the downloading of the software and tools required for the project, Sections 2 deals with the installation of the tools and Section 3 provides the software setup related details and other implementation details. The project emphasizes on the network simulation using NS2 simulator in combination with NSG2 which is used as tool for building the scenarios related to the simulationIssariyakul and Hossain (2009). To test the setup, a virtual machine with Ubuntu OS is created in Oracle Virtual Boc for simulating the wireless network topology.

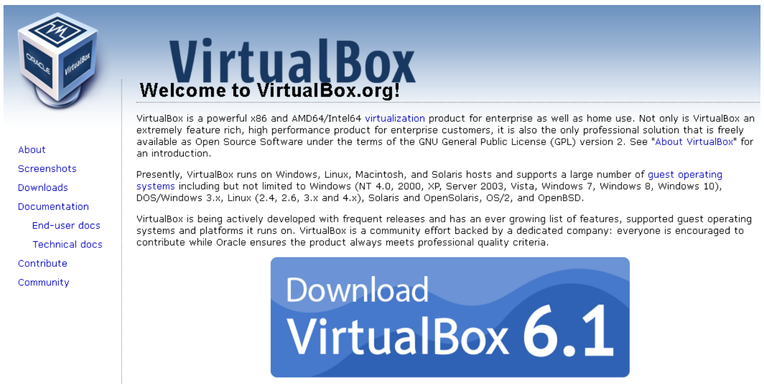
2 Downloading of Software and Tools

2.1 Oracle Virtual Box

The Oracle Virtual Box software is a hosted hypervisor that is installed and executed on the top of the host operating system. The Oracle Virtual Box is a open source software providing high performance and multiple features that are offered to users with a user-friendly graphical user interface Li (2010). It allows the users to create several guest operating system on a single physical machine. Oracle Virtual Box has been used in this project for creating a 64-bit virtual machine with Ubuntu guest operating using the below configuration:

RAM: 4GB
vCPU: 1
Storage: 40GB [1]

[1]https://www.virtualbox.org/



Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Figure 1: Oracle Virtual Box Configuration

2.2 Network Scenario Generator 2

The NSG2 is a software that can be used for building the simulation scenarios in a user-friendly environment. It was built by Pen-Jung Wu at the National University in Taiwan Huang and Shahnasser (2011). The network scenario generator has been used in our project in combination with the NS2 simulator to generate the wireless topology for testing the customized algorithm using SDN. Instead of manually writing the tcl code, the user can create the .tcl file in an automated manner using the NS2 network scenario generator. The highlighting characteristics of this tool are as follows:

1. The NSG2 is a tool that offers both the simplex and duplex network links for wired as well as wireless topology.

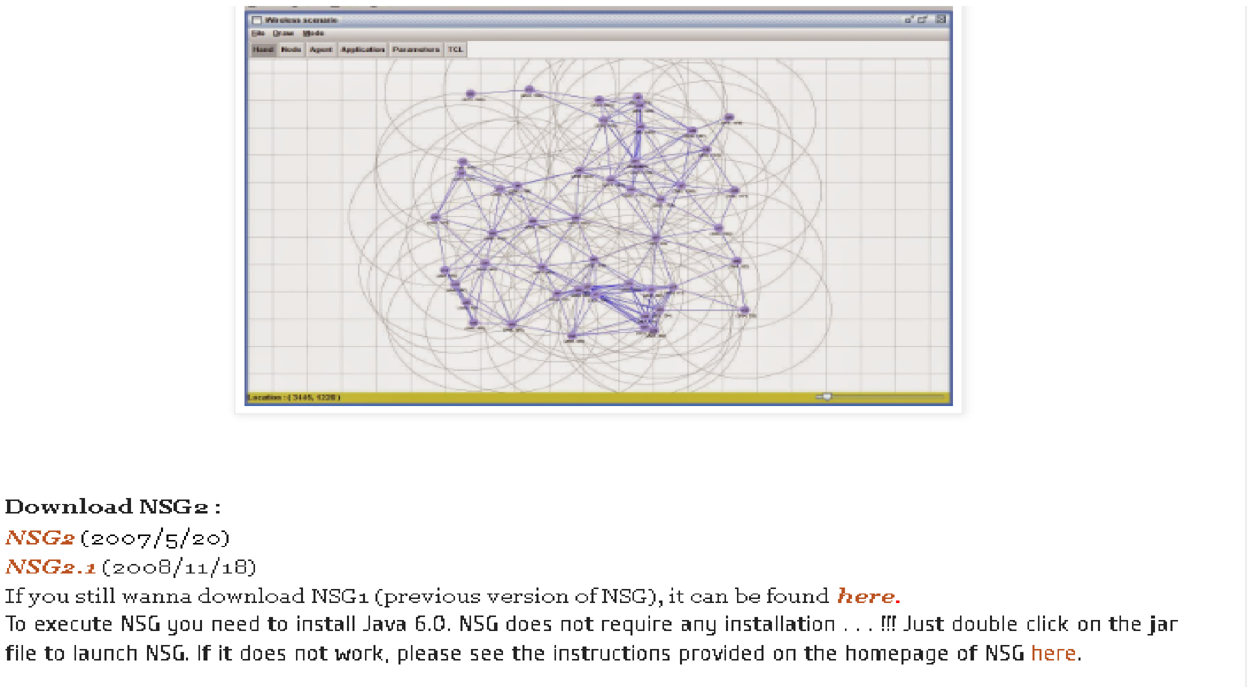
2. It supports different types of application-based traffic like the CBR (Constant Bit Rate, FTP (File transfer Protocol) and UDP (User Datagram protocol).

3. It also provides support for numerous routing protocols like DSR, AODV and DSDV and TORA.

4. It also enables the user to set the total duration of the simulation as per the requirements of the project and to save the generated simulation scenario in the form of a tcl file that can be used along with the ns command to run the simulation.

The NSG2(Network Simulator 2) tool can be downloaded in the form of a.jar file from the below source: [2]

[2]https://ns2blogger.blogspot.com/2014/04/nsg-21-tcl-script-generator.html



Download NSG2 :
NSG2 (2007/5/20)
NSG2.1 (2008/11/18)
If you still wanna download NSG1 (previous version of NSG), it can be found ***here.***
To execute NSG you need to install Java 6.0. NSG does not require any installation . . . !!! Just double click on the jar file to launch NSG. If it does not work, please see the instructions provided on the homepage of NSG here.

Figure 2: Network Scenario Generator for NS2

2.3 Network Simulator 2

NS2 is a simulator that makes use of events to perform the simulation of both wireless and wired networks. It also helps in the simulation of wireless routing protocols and network functions. The NS2 simulator offers users the ability to control and manage the behaviour of these routing algorithms in real-time. The NS2 simulator simulates the network protocols such as the UDP and TCP, traffic patterns like the Telnet, Constant Bit Rate (CBR), and routing protocols like Dynamic Source Routing and Ad-hoc Distance vector Routing. Due to its modular and flexible nature, NS2 has acquired worldwide popularity as a stable network simulator which can be extensively used in the networking and research fields. In our project, NS3 has been used for creating a wireless network topology to demonstrate the use of the SDN in combination of link weight routing algorithm. The network topology mainly comprises of the wireless sensor nodes in which the wireless sensor nodes generate the IoT traffic and pass the traffic to the SDN controller which receives the packet and sends the traffic to the destination node using the best available path. NAM is an animation tool that is used for viewing the NS2 simulation traces and events that are created with help of the Network Scenario Generator tool. The NAM window is activated upon the execution of the Tcl script. The url for downloading NS2 is given as a footnote. [3]

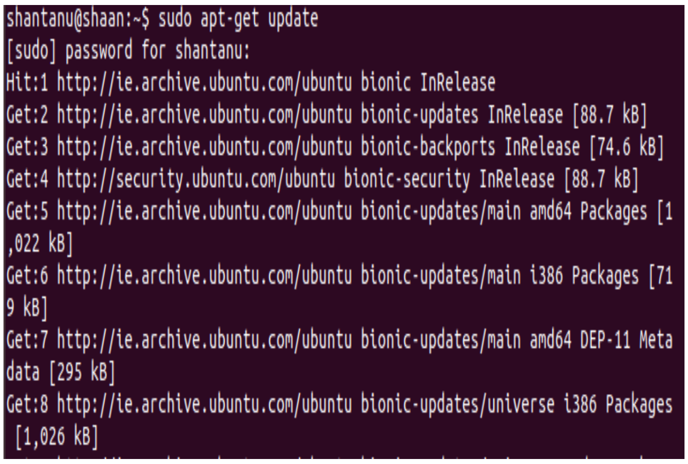
[3]https://www.absingh.com/ns2

3 Installation of Software Tools and Packages

Once all the required tools and softwares are downloaded for our project, the next step involves installation of all the tools. The installation of the tools is performed as below:
1. Installation of NS2 Package
2. Installation of Network Scenario Generator
3. Installation of xgraph

3.1 Installation of NS2 package

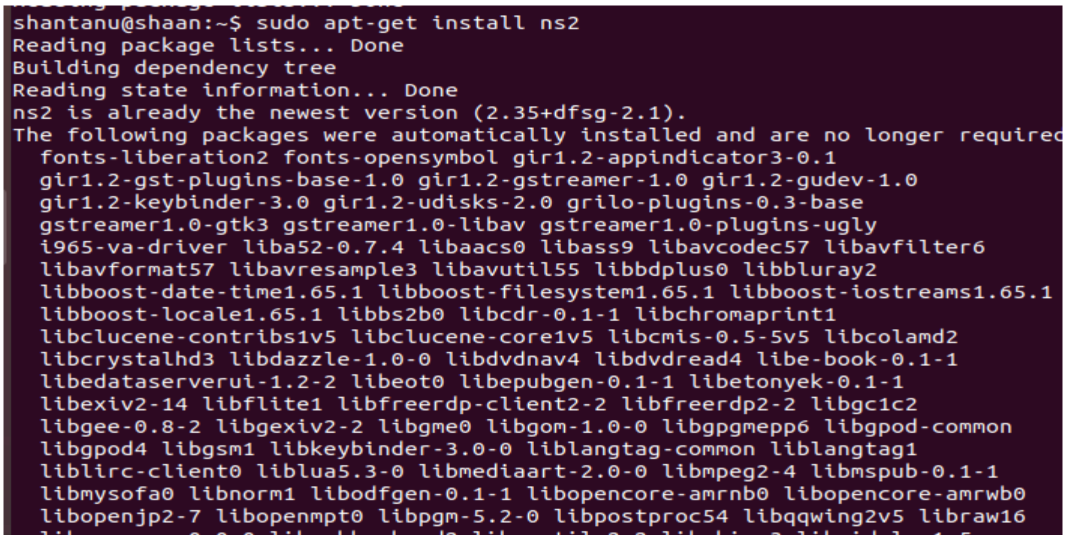
The below steps are used for installing the Network Simulator 2 on Ubuntu 64-bit operating system:
Step1: The prerequisite for installing the NS2 package is to update the Ubuntu repositories using the below command:



```
shantanu@shaan:~$ sudo apt-get update
[sudo] password for shantanu:
Hit:1 http://ie.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://ie.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://ie.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:5 http://ie.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [1
,022 kB]
Get:6 http://ie.archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [71
9 kB]
Get:7 http://ie.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Meta
data [295 kB]
Get:8 http://ie.archive.ubuntu.com/ubuntu bionic-updates/universe i386 Packages
 [1,026 kB]
```

Figure 3: Updating the Ubuntu Operating system

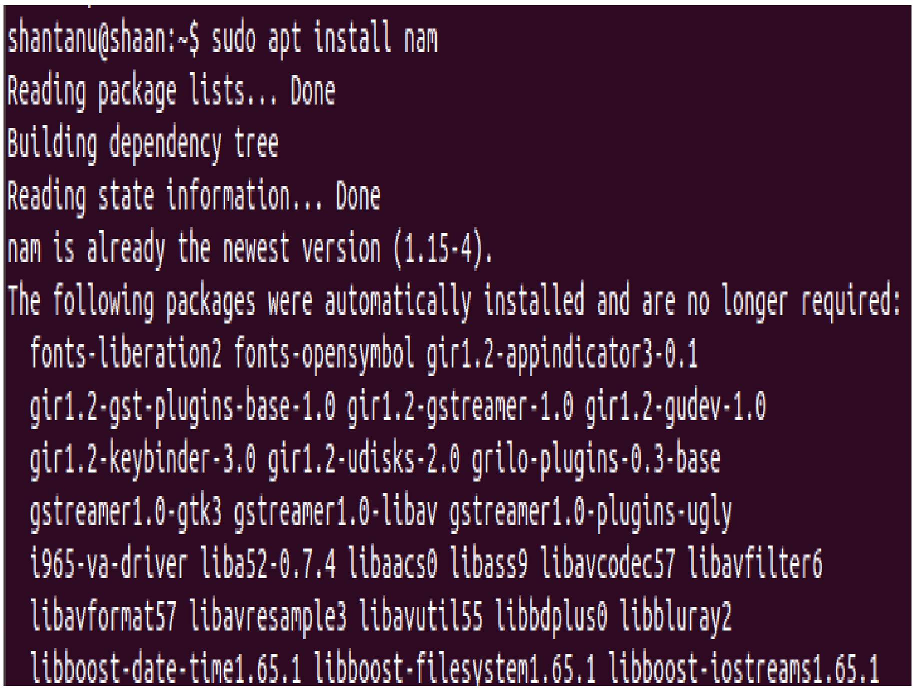
Step2: The Network Simulator 2 package can be installed using the below command.



```
shantanu@shaan:~$ sudo apt-get install ns2
Reading package lists... Done
Building dependency tree
Reading state information... Done
ns2 is already the newest version (2.35+dfsg-2.1).
The following packages were automatically installed and are no longer requirec
  fonts-liberation2 fonts-opensymbol gir1.2-appindicator3-0.1
  gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gudev-1.0
  gir1.2-keybinder-3.0 gir1.2-udisks-2.0 grilo-plugins-0.3-base
  gstreamer1.0-gtk3 gstreamer1.0-libav gstreamer1.0-plugins-ugly
  i965-va-driver liba52-0.7.4 libaacs0 libass9 libavcodec57 libavfilter6
  libavformat57 libavresample3 libavutil55 libbdplus0 libbluray2
  libboost-date-time1.65.1 libboost-filesystem1.65.1 libboost-iostreams1.65.1
  libboost-locale1.65.1 libbs2b0 libcdr-0.1-1 libchromaprint1
  libclucene-contribs1v5 libclucene-core1v5 libcmis-0.5-5v5 libcolamd2
  libcrystalhd3 libdazzle-1.0-0 libdvdnav4 libdvdread4 libe-book-0.1-1
  libedataserverui-1.2-2 libeot0 libepubgen-0.1-1 libetonyek-0.1-1
  libexiv2-14 libflite1 libfreerdp-client2-2 libfreerdp2-2 libgc1c2
  libgee-0.8-2 libgexiv2-2 libgme0 libgom-1.0-0 libgpgmepp6 libgpod-common
  libgpod4 libgsm1 libkeybinder-3.0-0 liblangtag-common liblangtag1
  liblirc-client0 liblua5.3-0 libmediaart-2.0-0 libmpeg2-4 libmspub-0.1-1
  libmysofa0 libnorm1 libodfgen-0.1-1 libopencore-amrnb0 libopencore-amrwb0
  libopenjp2-7 libopenmpt0 libpgm-5.2-0 libpostproc54 libqqwing2v5 libraw16
```

Figure 4: NS2 Package Installation

Step3: Once NS2 is installed, the NAM package is installed for graphically representing the network simulation traces.

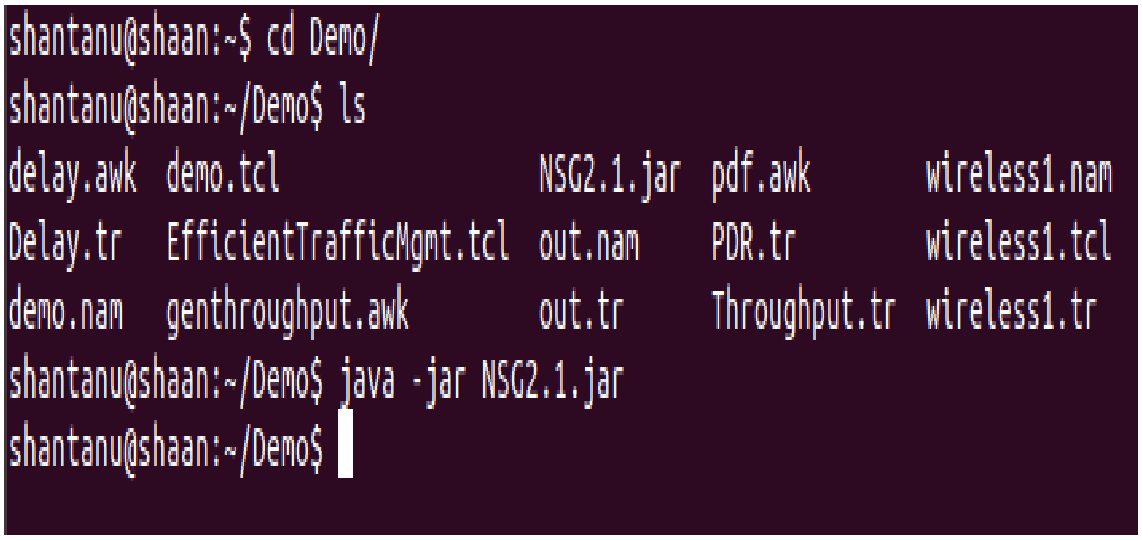


```
shantanu@shaan:~$ sudo apt install nam
Reading package lists... Done
Building dependency tree
Reading state information... Done
nam is already the newest version (1.15-4).
The following packages were automatically installed and are no longer required:
  fonts-liberation2 fonts-opensymbol gir1.2-appindicator3-0.1
  gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gudev-1.0
  gir1.2-keybinder-3.0 gir1.2-udisks-2.0 grilo-plugins-0.3-base
  gstreamer1.0-gtk3 gstreamer1.0-libav gstreamer1.0-plugins-ugly
  i965-va-driver liba52-0.7.4 libaacs0 libass9 libavcodec57 libavfilter6
  libavformat57 libavresample3 libavutil55 libbdplus0 libbluray2
  libboost-date-time1.65.1 libboost-filesystem1.65.1 libboost-iostreams1.65.1
```

Figure 5: Network Animator Installation

3.2 Network Scenario Generator Execution

After installing OpenJDK, the Network Scenario Generator is executed as a jar file. The NSG2 jar file can be executed using the below command



```
shantanu@shaan:~$ cd Demo/
shantanu@shaan:~/Demo$ ls
delay.awk  demo.tcl                NSG2.1.jar  pdf.awk        wireless1.nam
Delay.tr   EfficientTrafficMgmt.tcl out.nam     PDR.tr         wireless1.tcl
demo.nam   genthroughput.awk       out.tr      Throughput.tr  wireless1.tr
shantanu@shaan:~/Demo$ java -jar NSG2.1.jar
shantanu@shaan:~/Demo$ 
```

Figure 6: Network Scenario Generator Execution

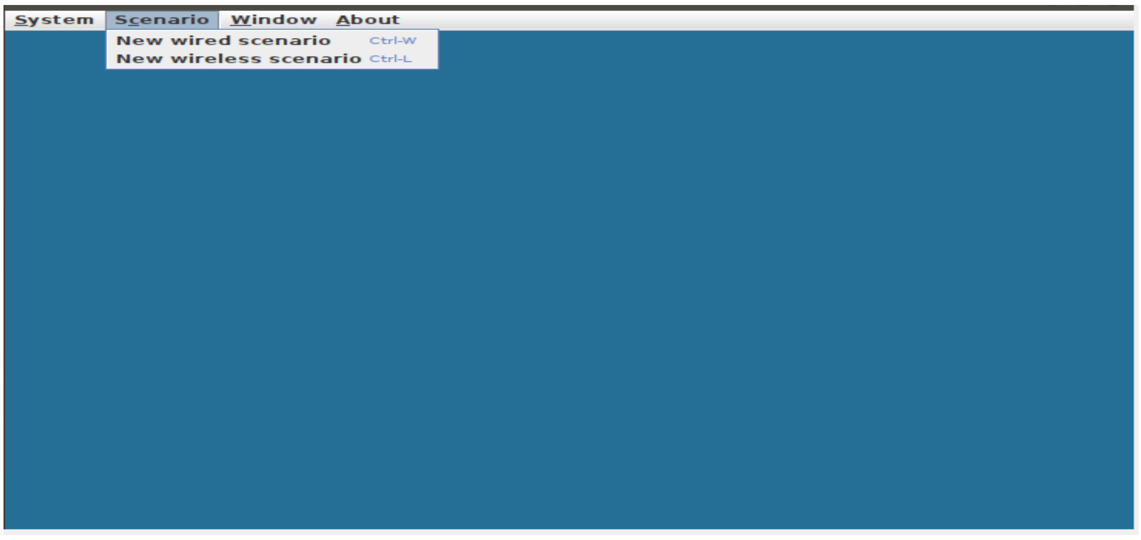


Figure 7: Network Scenario Generator Main Window

3.3 Wireless Topology Setup

In order to create a wireless topology, the “New wireless scenario” option is selected. On clicking new wireless scenario, a window is opened with all the options for creating wireless topology.

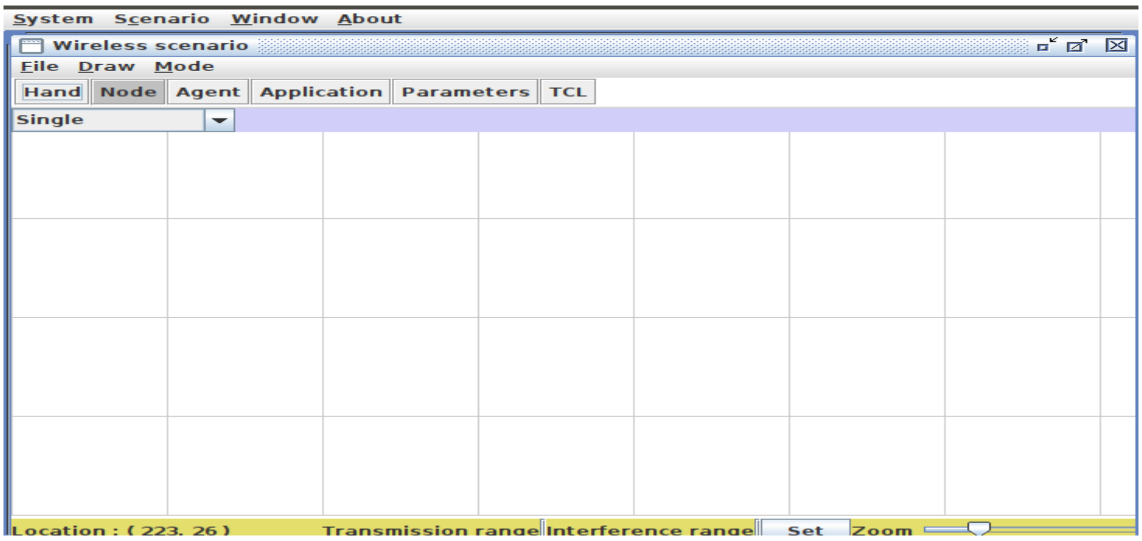


Figure 8: Network Scenario Generator Wireless Scenario

The wireless scenario can be generated by creating the wireless sensor nodes by clicking on the option called as “Node”. The nodes will be numbered as n0, n1. . . .so on.

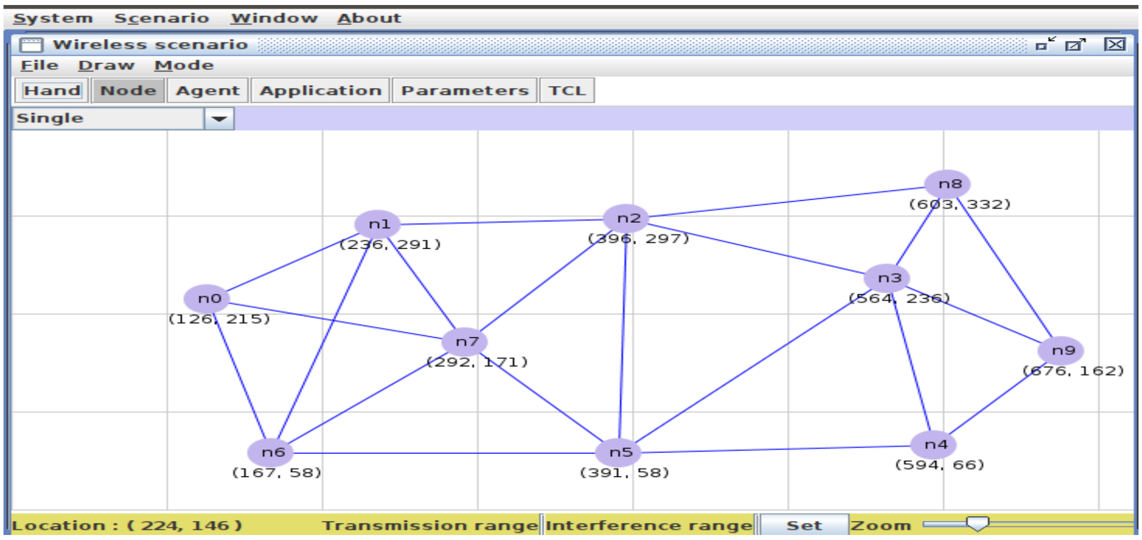


Figure 9: NSG2 Topology creation

Next, a TCP agent and a TCP Sink agent is attached to the source node and the Sink node so as to establish a TCP connection between the wireless node(n0) and the Sink node(n3).

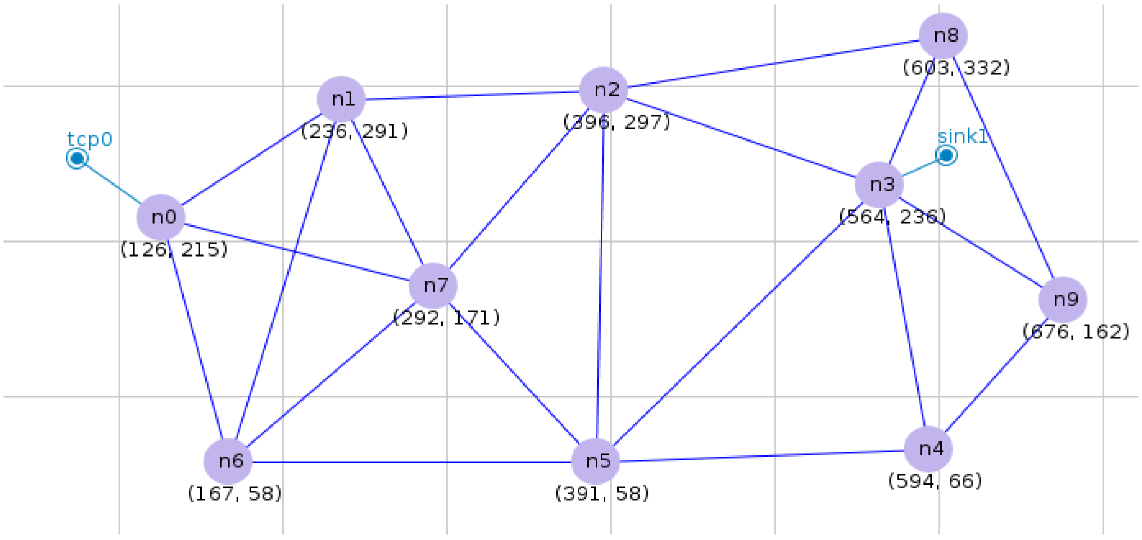


Figure 10: TCP agent and Sink Agent

For transferring the data through the TCP connection a cbr or ftp agent is attached to the TCP agent.

In the next step, the simulation time can be defined by entering the Start Time as 1sec and Stop time as 15sec.

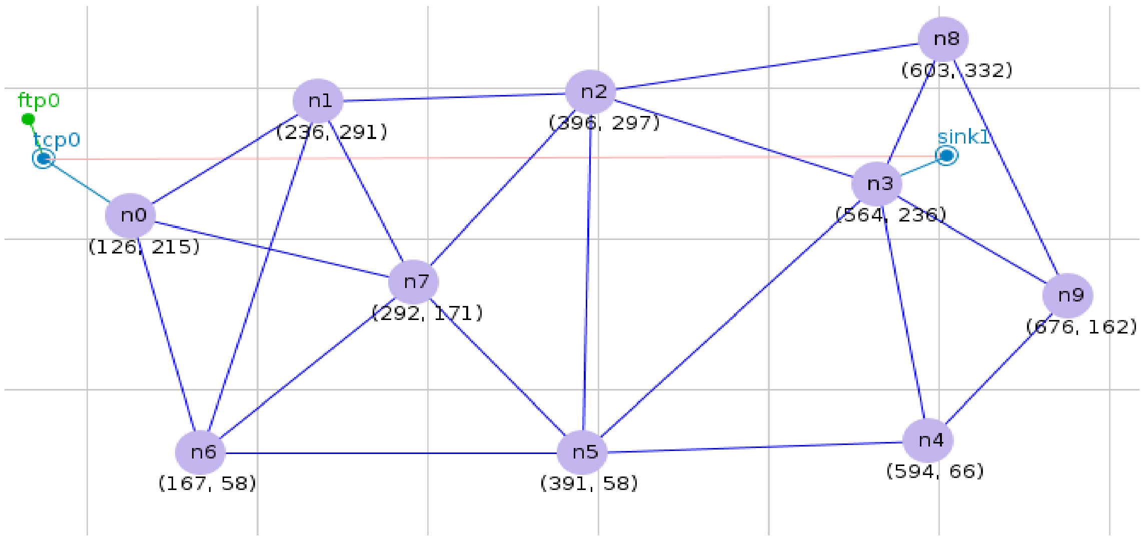
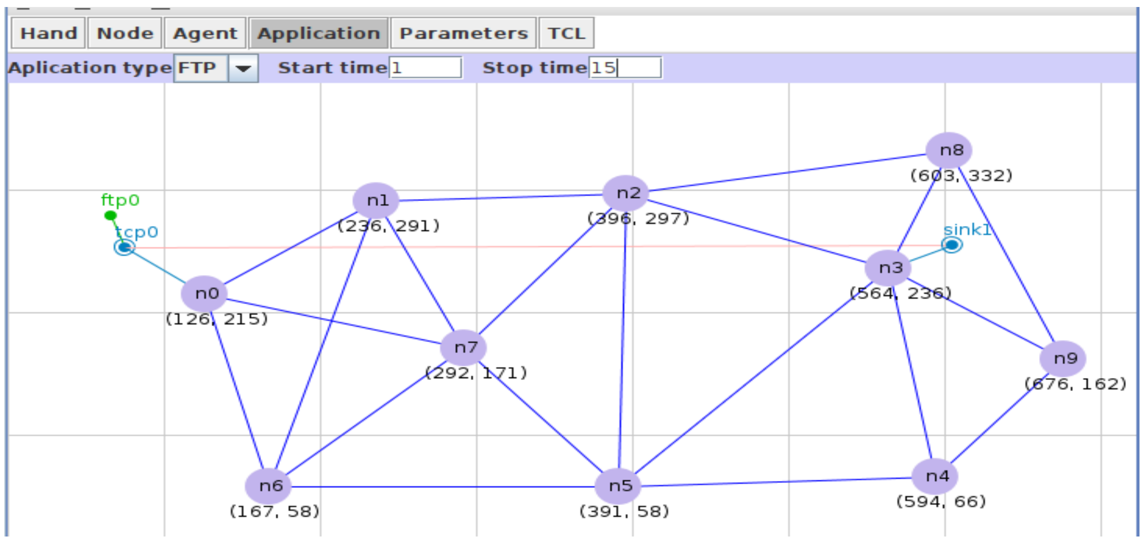


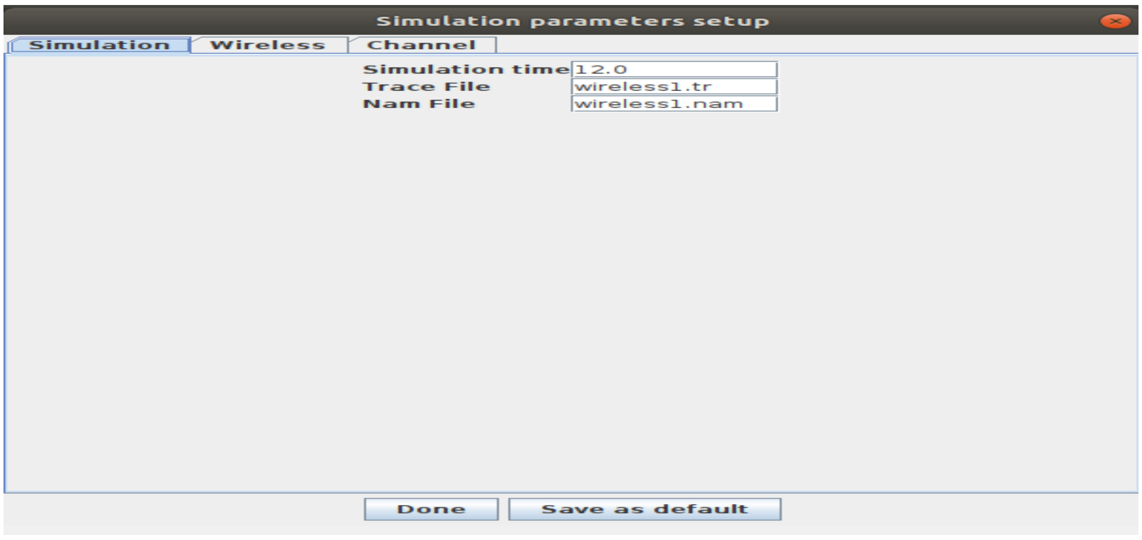
Figure 11: FTP agent attach to the TCP agent



Aplication type FTP Start time 1 Stop time 15

Figure 12: Simulation Time

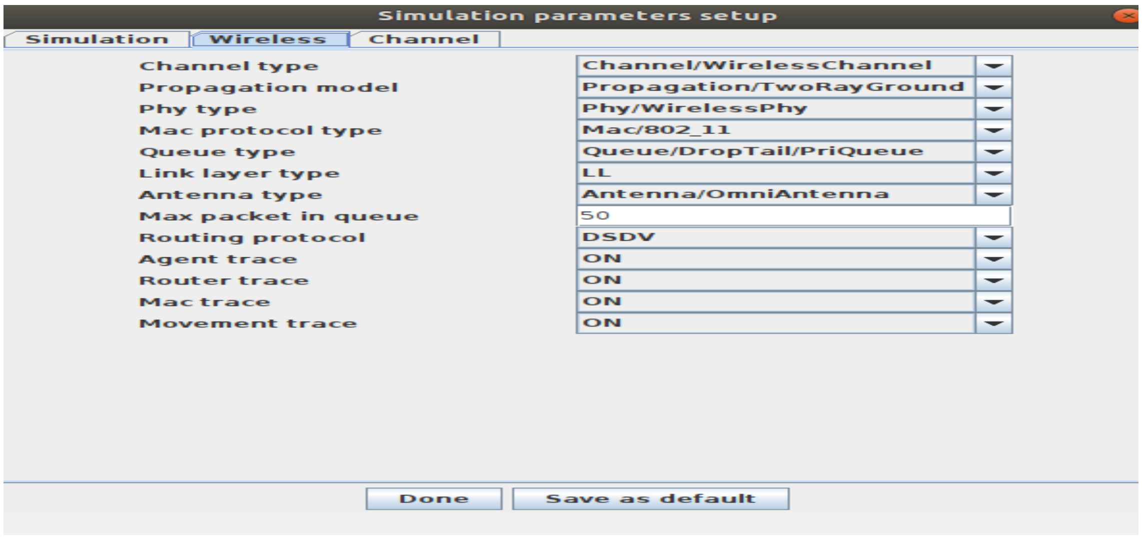
Lastly, the simulation parameters like the total simulation time, name of the trace file and name of the Nam output file are defined.



Simulation time	12.0
Trace File	wireless1.tr
Nam File	wireless1.nam

Figure 13: Simulation Setup

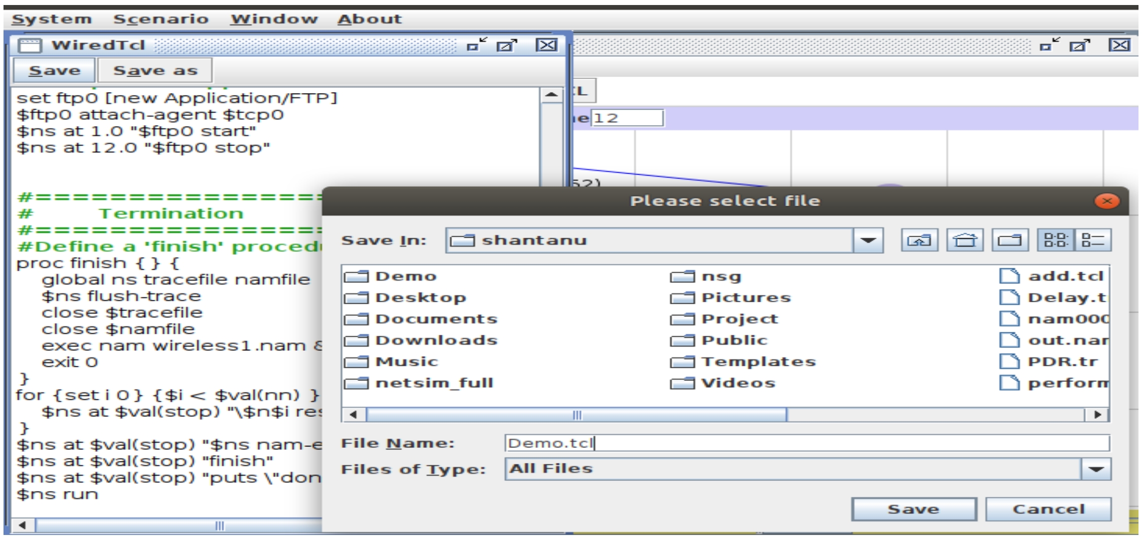
Before saving the simulation, wireless routing protocol like AODV, DSDV and DSR can be set by clicking on a tab named wireless.



Channel type	Channel/WirelessChannel
Propagation model	Propagation/TwoRayGround
Phy type	Phy/WirelessPhy
Mac protocol type	Mac/802_11
Queue type	Queue/DropTail/PriQueue
Link layer type	LL
Antenna type	Antenna/OmniAntenna
Max packet in queue	50
Routing protocol	DSDV
Agent trace	ON
Router trace	ON
Mac trace	ON
Movement trace	ON

Figure 14: Simulation Parameters

In the final step, the simulation is saved in the form of a .tcl file which can be used with the ns command to display the simulation workflow.



```
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 12.0 "$ftp0 stop"


#================
#     Termination
#================
#Define a 'finish' proced
proc finish { } {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam wireless1.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) }
    $ns at $val(stop) "\$n$i res
}
$ns at $val(stop) "$ns nam-e
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"don
$ns run
```

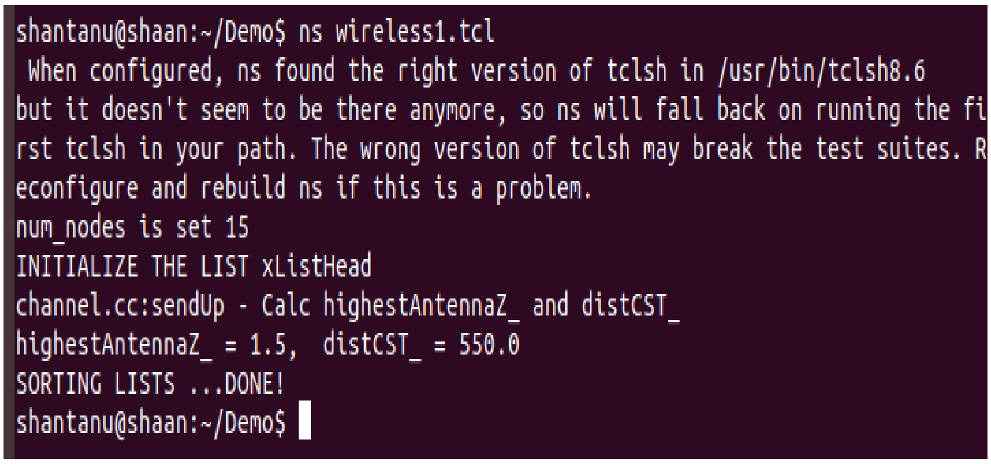
Save In: shantanu

Demo	nsg	add.tcl
Desktop	Pictures	Delay.t
Documents	Project	nam00
Downloads	Public	out.nar
Music	Templates	PDR.tr
netsim_full	Videos	perforn

File Name: Demo.tcl
Files of Type: All Files

Figure 15: TCL Scripting

After saving the simulation as a TCL file, the tcl file can be executed for running the simulation scenarios. For running the simulation, ns command is used followed by the name of the tcl file. The name of the tcl file used in the project is wireless1.tcl. When the tcl file is executed, it calculates the number of nodes that are part of the topology design and generates the corresponding number of wireless nodes as per the topology setup and also labels the nodes and performs the initialization of the tcp, udp or cbr traffic. The simulator runs the simulation as per the time defined in the NSG2 wireless scenario.



```
shantanu@shaan:~/Demo$ ns wireless1.tcl
 When configured, ns found the right version of tclsh in /usr/bin/tclsh8.6
but it doesn't seem to be there anymore, so ns will fall back on running the fi
rst tclsh in your path. The wrong version of tclsh may break the test suites. R
econfigure and rebuild ns if this is a problem.
num_nodes is set 15
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
shantanu@shaan:~/Demo$
```

Figure 16: NS2 TCL script Execution

When the TCL script is executed, it transfers the call to a .nam file and displays the simulation output. As seen in the below figure, the topology consists of wireless nodes.

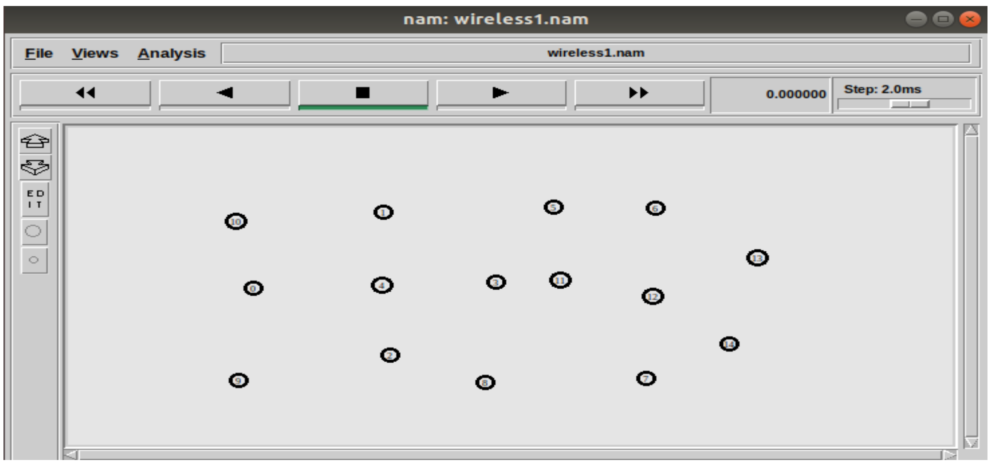
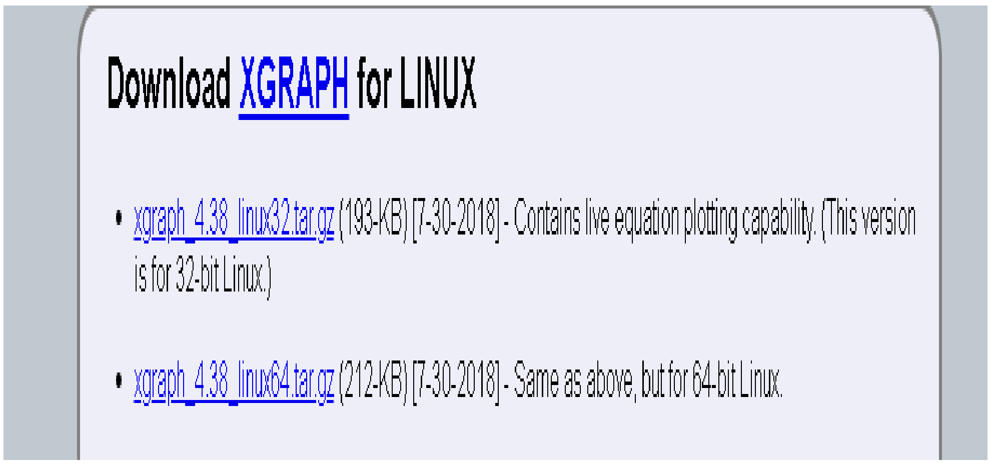


Figure 17: NS2 NAM Output

4 Plotting Simulation Traces on XGraph

For evaluation of results, a tool known as xgraph is used which helps to plot the information related to different performance like Throughput, response time and average end-to-end delay in a graphical format. xgraph can downloaded in the form of a tar ball using the below link:

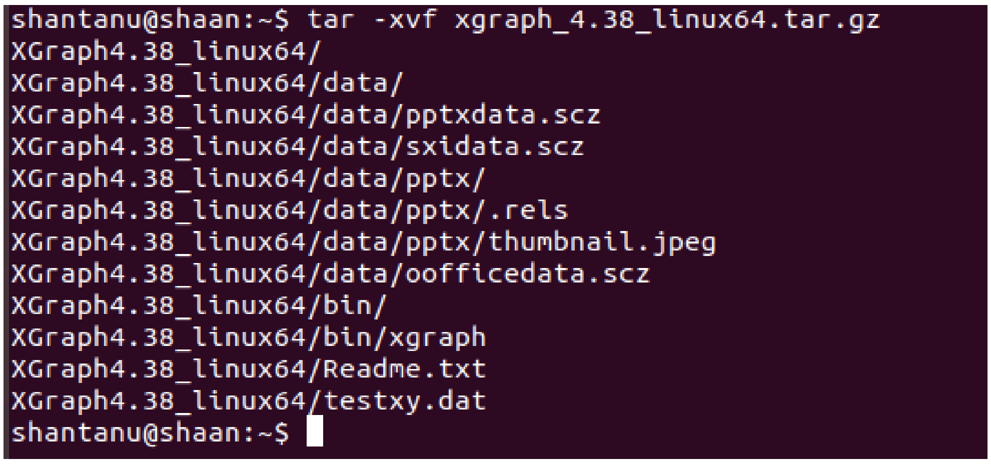


Download XGRAPH for LINUX

- xgraph_4.38_linux32.tar.gz (193-KB) [7-30-2018] - Contains live equation plotting capability. (This version is for 32-bit Linux.)
- xgraph_4.38_linux64.tar.gz (212-KB) [7-30-2018] - Same as above, but for 64-bit Linux.

Figure 18: XGraph

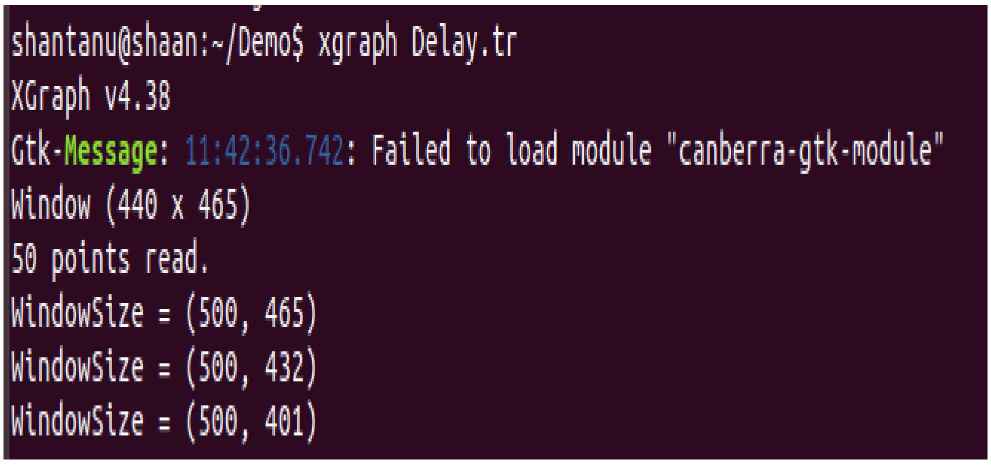
In order to use xgraph, the contents of the tar gzip file are extracted. For extracting the contents of the gzip tar file, -xvf option is used in the tar command where x stands for extract, v stands for verbose and f stands for the name of the tar file. Xgraph is a tool that is used to present the graphical output of the simulation scenarios. Xgraph plots the graph using a X11 window by capturing the data from either the simulation trace files or through the standard input. It annotates the graphical display with axis labels, tables and tick marks or grid lines legend and grid labels Patel et al. (2019). There are several alternatives to regulate the appearance of the components of the Xgraph. The graphical interface used to indicate the size and location and volume of the window is dependent on the window manager which is being used. As soon as the X window is opened, all the points in the data set or the simulation trace file are graphically displayed



```
shantanu@shaan:~$ tar -xvf xgraph_4.38_linux64.tar.gz
XGraph4.38_linux64/
XGraph4.38_linux64/data/
XGraph4.38_linux64/data/pptxdata.scz
XGraph4.38_linux64/data/sxidata.scz
XGraph4.38_linux64/data/pptx/
XGraph4.38_linux64/data/pptx/.rels
XGraph4.38_linux64/data/pptx/thumbnail.jpeg
XGraph4.38_linux64/data/oofficedata.scz
XGraph4.38_linux64/bin/
XGraph4.38_linux64/bin/xgraph
XGraph4.38_linux64/Readme.txt
XGraph4.38_linux64/testxy.dat
shantanu@shaan:~$
```

Figure 19: Xgraph

along with a legend. The trace file generated after the completion of the simulation is used as an input to the xgraph command for generating the graphical representation of the simulation events. [4]



```
shantanu@shaan:~/Demo$ xgraph Delay.tr
XGraph v4.38
Gtk-Message: 11:42:36.742: Failed to load module "canberra-gtk-module"
Window (440 x 465)
50 points read.
WindowSize = (500, 465)
WindowSize = (500, 432)
WindowSize = (500, 401)
```

Figure 20: Xgraph-Command

[4]https://www.xgraph.org/linux/index.html

The output of the xgraph command is as follows:

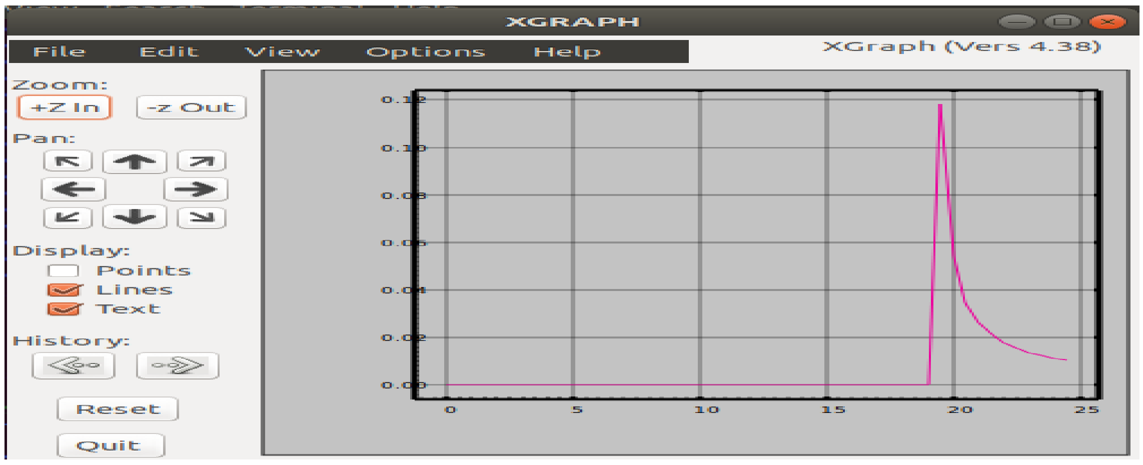
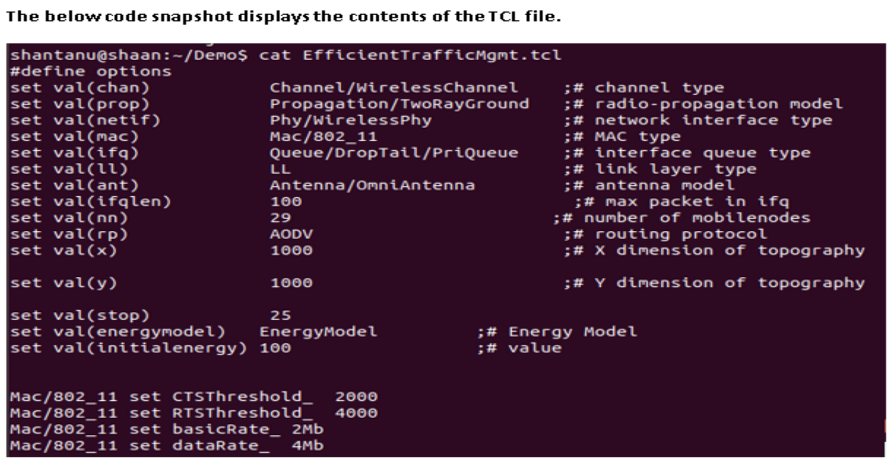


Figure 21: XGraph Sample Output

5 Implementation

The .tcl file built for creating the wireless topology is used along with the ns command to run the simulation. The TCL script consists of 28 wireless nodes that communicate with each other using the wireless links. The wireless nodes communicate with each other by exchanging the ftp and cbr traffic over a TCP connection.

5.1 Code for Simulator Object Declaration:



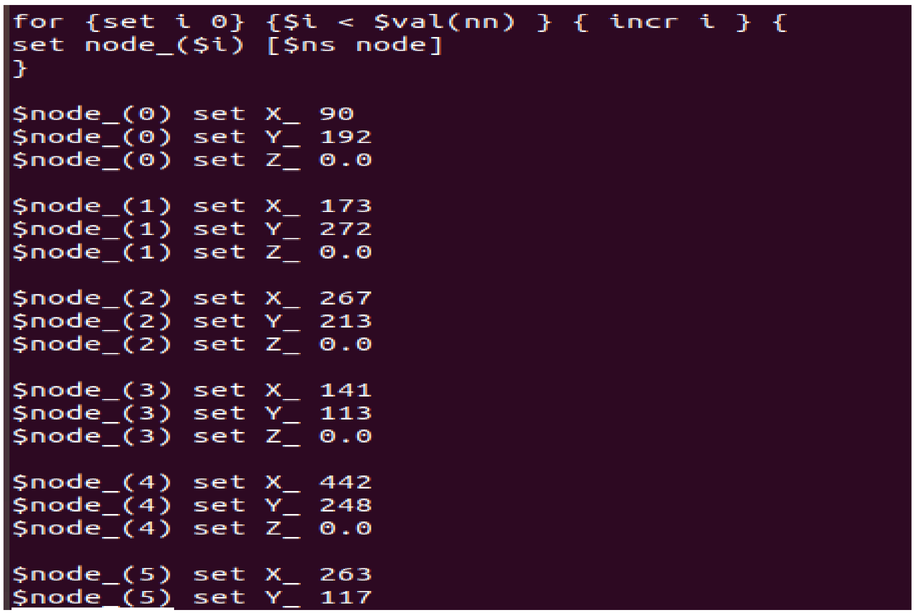
The below code snapshot displays the contents of the TCL file.

```
shantanu@shaan:~/Demo$ cat EfficientTrafficMgmt.tcl
#define options
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         100                         ;# max packet in ifq
set val(nn)             29                        ;# number of mobilenodes
set val(rp)             AODV                       ;# routing protocol
set val(x)              1000                       ;# X dimension of topography

set val(y)              1000                       ;# Y dimension of topography

set val(stop)           25
set val(energymodel)   EnergyModel         ;# Energy Model
set val(initialenergy) 100                 ;# value


Mac/802_11 set CTSThreshold_  2000
Mac/802_11 set RTSThreshold_  4000
Mac/802_11 set basicRate_ 2Mb
Mac/802_11 set dataRate_  4Mb
```

Figure 22: Code for Simulator object Declaration

5.2 TCL Code for Node Generation:

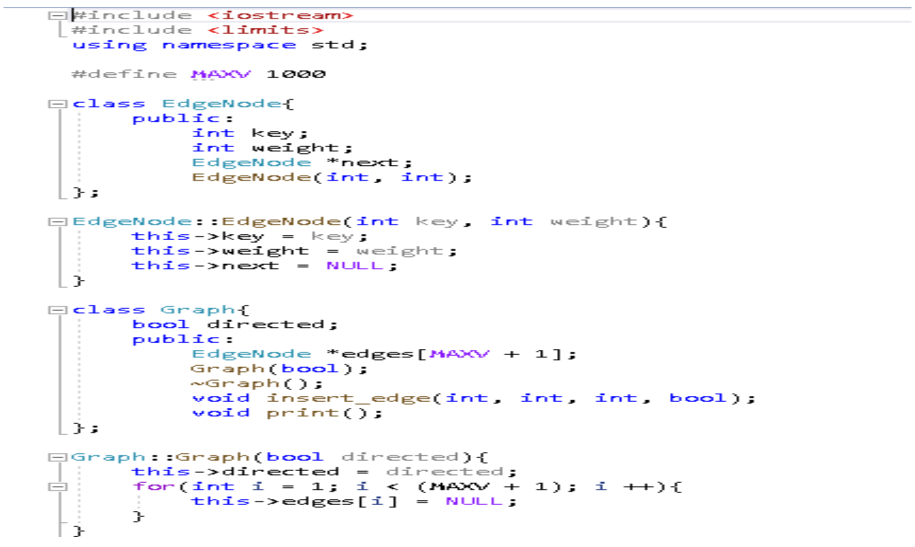
The wireless topology in our project consists of 28 wireless nodes attached with RFID tags. The experiment is carried out by performing 5 iterations and varying the count of nodes in each of the iterations. The position for the wireless nodes is set by defining the position relative to the X, Y axis. These values are set by using the Network Scenario Generator whereby the user drags the node to a particular position and the scenario generator sets the position of the wireless node in the respective tcl file.



```
for {set i 0} {$i < $val(nn) } { incr i } {
set node_($i) [$ns node]
}

$node_(0) set X_ 90
$node_(0) set Y_ 192
$node_(0) set Z_ 0.0

$node_(1) set X_ 173
$node_(1) set Y_ 272
$node_(1) set Z_ 0.0

$node_(2) set X_ 267
$node_(2) set Y_ 213
$node_(2) set Z_ 0.0

$node_(3) set X_ 141
$node_(3) set Y_ 113
$node_(3) set Z_ 0.0

$node_(4) set X_ 442
$node_(4) set Y_ 248
$node_(4) set Z_ 0.0

$node_(5) set X_ 263
$node_(5) set Y_ 117
```

Figure 23: TCL for Node generation

5.3 Code for Link Weight Routing Algorithm



```
#include <iostream>
#include <limits>
using namespace std;

#define MAXV 1000

class EdgeNode{
    public:
        int key;
        int weight;
        EdgeNode *next;
        EdgeNode(int, int);
};

EdgeNode::EdgeNode(int key, int weight){
    this->key = key;
    this->weight = weight;
    this->next = NULL;
}

class Graph{
    bool directed;
    public:
        EdgeNode *edges[MAXV + 1];
        Graph(bool);
        ~Graph();
        void insert_edge(int, int, int, bool);
        void print();
};

Graph::Graph(bool directed){
    this->directed = directed;
    for(int i = 1; i < (MAXV + 1); i ++){
        this->edges[i] = NULL;
    }
}
```

Figure 24: Code for Link Weight Routing Algorithm

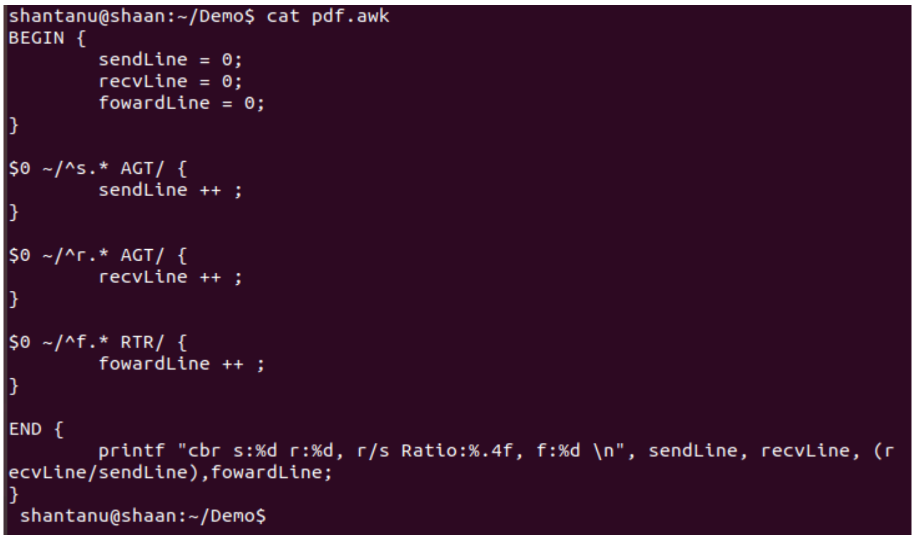
6 Scripts for Recording the Performance Statistics

The performance of the NS2 simulation is recorded by making the use of a performance script. The scripting language for capturing the performance statistics is known as the awk script. The awk is a language that is used for recording the below performance parameters:

- Packet Delivery Ratio
- Average end-to-end Delay
- Throughput

6.1 Packet Delivery Ratio

The packet delivery ratio is the count of packets that were successfully received to the count of the packets that were sent. The packet delivery ratio is measured in the form of percentage. The results are captured by varying the number of nodes used in the topology for the existing as well as the proposed algorithm. The awk script used for measuring the packet delivery ratio is as follows:

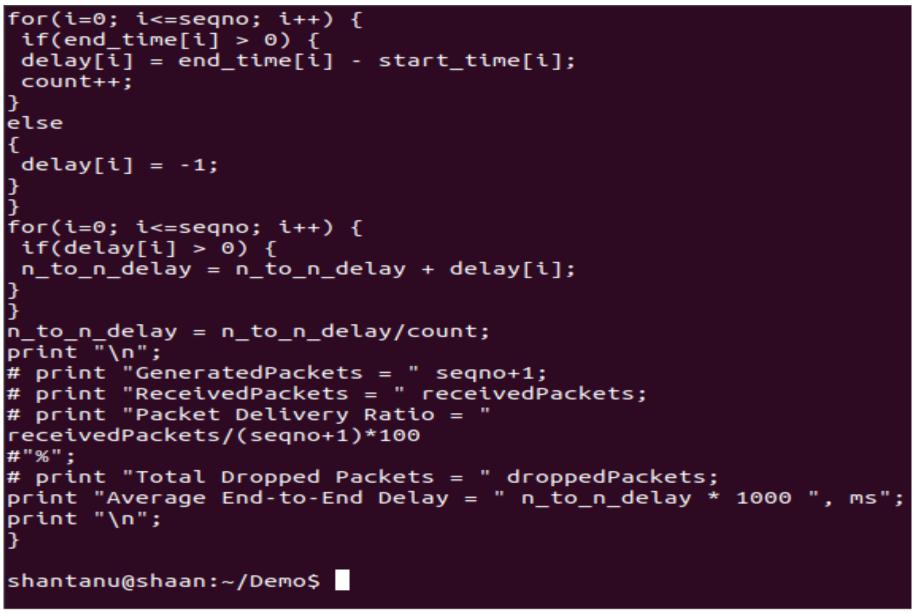


```
shantanu@shaan:~/Demo$ cat pdf.awk
BEGIN {
        sendLine = 0;
        recvLine = 0;
        fowardLine = 0;
}

$0 ~/^s.* AGT/ {
        sendLine ++ ;
}

$0 ~/^r.* AGT/ {
        recvLine ++ ;
}

$0 ~/^f.* RTR/ {
        fowardLine ++ ;
}

END {
        printf "cbr s:%d r:%d, r/s Ratio:%.4f, f:%d \n", sendLine, recvLine, (r
ecvLine/sendLine),fowardLine;
}
 shantanu@shaan:~/Demo$
```

Figure 25: AWK script for Packet Delivery Ratio

6.2 Average End-to-End Delay

The average end-to-end delay is the additional amount of time taken by a packet to reach the destination after leaving the source node. It is computed as the difference between the time taken by the packet and the expected time to reach the destination. The average end-to-end delay is computed by using an awk script. The delay is captured based on the count of the number used in the topology. The awk script used for analysing the end-to-end is as follows:

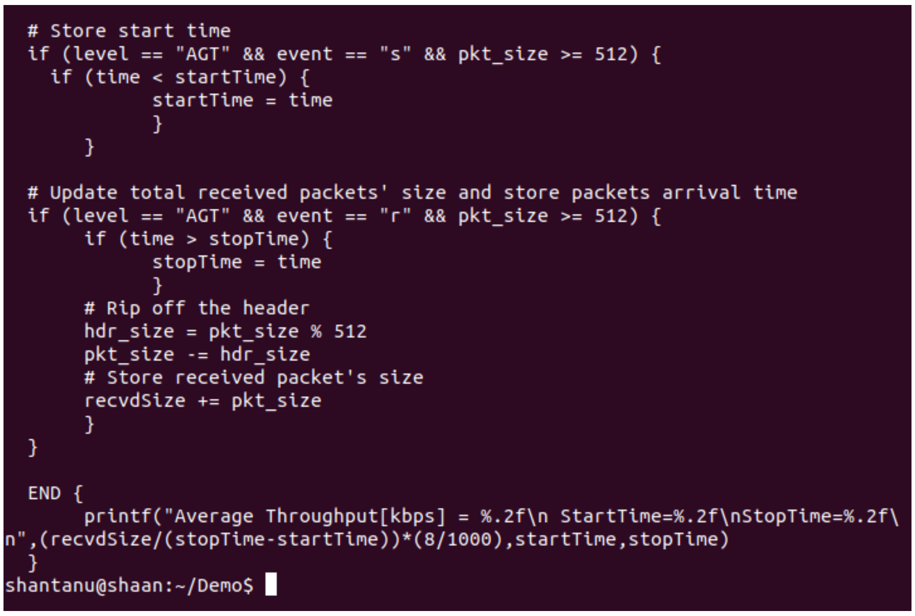


```
for(i=0; i<=seqno; i++) {
 if(end_time[i] > 0) {
 delay[i] = end_time[i] - start_time[i];
 count++;
}
else
{
 delay[i] = -1;
}
}
for(i=0; i<=seqno; i++) {
 if(delay[i] > 0) {
 n_to_n_delay = n_to_n_delay + delay[i];
}
}
n_to_n_delay = n_to_n_delay/count;
print "\n";
# print "GeneratedPackets = " seqno+1;
# print "ReceivedPackets = " receivedPackets;
# print "Packet Delivery Ratio = "
receivedPackets/(seqno+1)*100
#"%";
# print "Total Dropped Packets = " droppedPackets;
print "Average End-to-End Delay = " n_to_n_delay * 1000 ", ms";
print "\n";
}

shantanu@shaan:~/Demo$
```

Figure 26: AWK script for Average End-to-End Delay

6.3 Throughput

The throughput of a network is measured by the rate at which the packets were transferred from the source to the destination node. Whenever there is congestion in the network, it results in packet loss and network congestion. The delay time can be calculated with use of an awk script. The awk script used for analysing the throughput is as follows:



```
  # Store start time
  if (level == "AGT" && event == "s" && pkt_size >= 512) {
    if (time < startTime) {
             startTime = time
             }
      }

  # Update total received packets' size and store packets arrival time
  if (level == "AGT" && event == "r" && pkt_size >= 512) {
      if (time > stopTime) {
             stopTime = time
             }
      # Rip off the header
      hdr_size = pkt_size % 512
      pkt_size -= hdr_size
      # Store received packet's size
      recvdSize += pkt_size
      }
  }

  END {
      printf("Average Throughput[kbps] = %.2f\n StartTime=%.2f\nStopTime=%.2f\
n",(recvdSize/(stopTime-startTime))*(8/1000),startTime,stopTime)
  }
shantanu@shaan:~/Demo$
```

Figure 27: AWK script for Throughput

References

Huang, J. and Shahnasser, H. (2011). A preprocessor tcl script generator for ns-2 communication network simulation, *2011 International Conference on Communications and Information Technology (ICCIT)*, IEEE, pp. 184–187.

Issariyakul, T. and Hossain, E. (2009). Introduction to network simulator 2 (ns2), *Introduction to network simulator NS2*, Springer, pp. 1–18.

Li, P. (2010). Selecting and using virtualization solutions: our experiences with vmware and virtualbox, *Journal of Computing Sciences in Colleges* **25**(3): 11–17.

Patel, R., Patel, N. and Patel, S. (2019). An approach to analyze behavior of network events in ns2 and ns3 using awk and xgraph, *Information and Communication Technology for Competitive Strategies*, Springer, pp. 137–147.