

# Survey of Various Open Source Network Simulators

Uma R Pujeri<sup>1</sup>, Dr. V Palanisamy<sup>2</sup>

<sup>1</sup>Assistant Professor, Research Scholar Anna University Chennai, Tamil Nadu, India

<sup>2</sup>Principal, Info Institute of Technology, Coimbatore, Tamil Nadu, India

**Abstract:** *Computer networks can be defined as networks of computers and checking the performance of the network like throughput of network, delay, channel utilization, packet loss etc is very difficult in real time. A single test bed network consists of many routers, switches, computers, servers and other networking devices hence implementing any new routing algorithm or protocol and testing it in this test bed network requires lot of time as well as cost. The simulator helps the network developer to check whether his routing algorithm designed for the network is able to work in real world. Network simulator is useful for the network designer to implement and test his own protocol or new routing algorithm in controlled and reproductive manner thus saving lot of cost and time. There are many sophisticated and powerful simulating tools available, but selecting the proper simulator that gives proper and appropriate result needs the knowledge of various simulating tools available along with their strength and weakness. It is very important to verify the results generated by these simulating tools are valid, accurate and truthful. The objective of this paper is to survey classify and compare different network simulator so that the network researcher can select appropriate simulating tools which will help him to carry out his research. In this paper we have discussed different open source simulators its features, its strength and weakness. We hope this paper will provide the researcher guidelines to select appropriate open-source network simulator to carry out their research.*

**Keywords:** NS2, NS3, OmNet++, JSim, Glomosim

## 1. Introduction

Network simulator allows the researcher to design and test protocol or algorithm that are difficult and expensive to test in real test bed network.. Researcher can design his own topology by using various types of nodes (host, hub, bridges, routers and mobile units etc) and can test his protocol or algorithm in this topology. Researchers can also compare his new algorithm or new protocol with the exiting protocol by using these simulation tools. Strength and weakness of these simulation tools are discussed below

### Strength

1. The System can be studied analyzed without building it.
2. Results are accurate compared to analytical model.
3. Helps to find bugs, unexpected phenomenon and behavior of system in advance.
4. Easy to perform analysis.

### Weakness

1. Uncertainty in results, sometimes difficult to interpret the simulation result.
2. Expensive to build simulation model and conduct simulation.

We can say that network simulator is used to test, analyze, evaluate the performance of new algorithm or new protocol and compare the new algorithm with the existing algorithm. However the result generated by the simulator may not be exact or accurate which may make your algorithm to fail in real time. To overcome this it is very important for the researcher to select appropriate simulating tool that is easy to use, more flexible in model development modification and validation and also does appropriate analysis of output data. To select the credible simulation tool the researcher should have the knowledge of different simulation tools available along with their strength and weakness.

The network simulator can be classified based on

1. range (i.e. simple, moderate or complex)

2. Whether simulator supports GUI (graphical user interface)
3. Whether simulator determines the numbers of nodes in network, number of links between the node etc.
4. Whether the simulator analyze traffic between the nodes.
5. Whether the simulator supports text based application to support advance form of customization
6. Whether the simulator supports programming oriented tool to create an application to simulate the networking environment to be tested.
7. Whether the simulator shows the performance of network by calculating the throughput, packet loss delay, channel utilization etc in the network.

In this paper we have done a survey on the open source network simulators. Open source network simulators have an advantage that the source code is available for modification or enhancement by anyone. We have surveyed the following open source network simulating tools

1. NS2
2. NS3
3. OmNet++
4. JSim
5. Glomosim

We have classified and compared the above simulator based on type and deployment node along with the protocol supported. The researcher and developer can use this study in selecting the most appropriate simulator.

## 2. Open Source Software's

Open source is software whose source code is available publicly for modification, enhancement, testing, redistribution etc. Open source must have the following criteria

**1. Free redistribution**

The software must be available at free of cost for public and must prohibit any party from selling this open source software

**2. Source code**

Source code must be available publicly with all the files, library files, patch files and compilation form.

**3. Derived work**

Original software can be modified and the derived work can be distributed with the same terms as the license of the original.

**4. No discrimination against the person or group**

The license must not be discriminated against any person or group.

**5. Technology Neutral**

The license should support all technologies and interfaces and must not be specific for particular technology or interface.

**6. Should support all fields of endeavor**

License should not be for specific field for example open source software can be used in business field, medical field, engineering field etc.

**7. License should not be restricted to specific product or software.**

Advantages of open source software

1. Open source software are free to use modify and redistribute and are hence cost effective.
2. Researcher or developer can build a new software by modifying the existing which saves lot of time and also avoids writing a code from scratch.
3. New derived software can be freely distributed.
4. Open source does not restrict the developer to develop the code for specific vendor, specific software, specific hardware or specific technology.
5. It is continually evolving in real time because developers keep on adding the module improving the quality, security and performance of the software.

Disadvantages of open source software

1. Open source software may not be user friendly or may not have good graphical interface because less attention is paid for development of user interface.
2. The monitoring of open source software is very difficult because anyone can use the software, modify software and distribute the software.
3. The codes are very complicated and are difficult for the beginners to understand.
4. Though it is free of cost but indirect cost may be involved for paying for external support
5. Malicious user can potentially view open source software and exploit any vulnerability.

**3. OMNET (Optical Micro Network Plus Plus)**

OMNet++ is an open source network simulator with GUI support. It is component based modular and open architecture discrete event based framework. Its INET package provides a comprehensive collection of Internet protocol. OMNet++ is made up of modules which can be classified as simple module and compound module. Simple module: Simple modules are used to define algorithm in

which event can occur, these modules are active components of the system which defines the behavior of the system. Compound modules: Compound modules are a collection of simple modules which interact with each other.

Languages Supported:

Strength

1. OMNet++ supports GUI(graphical user interface)
2. Compiler for NED topology description language
3. Command line interface for simulation execution
4. Documentation
5. OMNet++ has modular architecture and supports implementation of modules and protocols
6. Has graphical output vector plotting tool
7. Supports number of utilities like random number seed generation tool, make file creation tool etc.
8. OMNet++ support reusable modules
9. Large scale simulation as well as queuing simulation can be carried using this simulator. It can be used in other areas also.
10. Supports Unix/Linux , Windows and Mac OS.

Weakness

1. It is bit difficult to use
2. Novice user may long time t learn this new simulator
3. It is slow comparative to other simulators
4. It consumes lot of memory
5. Does not support all types of simulation.

Future work

1. OMNet++ is a platform network simulation where academia as well as industry can contribute adding more modules so that it can support more and more simulation.
2. Also research is needed to improve its speed and memory consumption.
3. More documentation and tutorials can be made available which may help novice user to learn this simulator easily.
4. More workshops on this simulator can be conducted so that more and more researcher can contribute in this field.

**4. NS2**

NS2 is an open source network simulator many network researcher use this simulator to write a new protocol and test its performance using this simulator. It is an object oriented discrete event driven network simulator. It is developed at UC Berkeley and is written in C++ and OTCL.

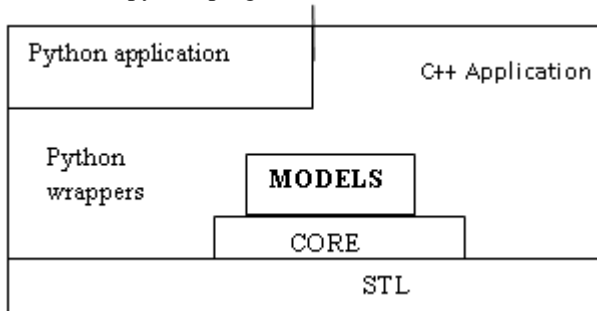
Overview- NS2 can be used to implement protocol like TCP and UDP, traffic source behavior such as Telnet, FTP, Web, CBR and VBR router queue management such as DropTail, CBR and RED routing algorithm such as Dijkstra and more.

Network research can write his own protocol in c++ but to setup and run network simulation researcher should write an OTcl script that initiates an event scheduler setup an network topology using the object and plumbing function in library and tell traffic sources when to start and stop transmitting packets through the event scheduler. Plumbing makes NS2 very powerful because when a user wants to

make a new network object, user can write a new object or even make compound object from object library and plumb data through object, another major component of NS2 is event scheduler.

## 5. NS3

NS3 is a free open source software discrete event network simulator for internet system. Overview: It allows network researcher to implement new internet protocol in a controlled environment. NS3 is written in C++ with binding available for python i.e. simulation program are C++ executable or python programs



Architecture of NS3

NS3 lacks an integrated development visualization environment (IDE) and NS3 is not backward compatible with ns2.

### Features

- Extensible software core: The Ns3 is written in C++ with python scripting interface.
- Attention to realism: Protocol entities are designed closer to real computer network.
- Software integration: Incorporates of more open-source networking software and reduces the need to rewrite code or software.
- Support for virtualization and test beds
- Flexible tracing and statistics.
- Attribute System
- New model.

### Limitations:

- Python binding do not work on cygwin
- Only IPV4 is supported.

### Future work

- NS3 is a new simulating tool and major challenge in NS3 is still unresolved
- NS3 needs lots of researcher to participate in research community
- Lots of NS3 for NS3 and writing a script and code in NS3 should be hosted
- Simulation credibility needs to be improved by
  - Capturing level of community acceptance
  - Publication lists and cross reference
  - Validation techniques.
  - Needs to identify maintainers : In ns3 the active maintainer are required to respond to the user questions and bug reports and help to test validating the system.

## 6. Glomosim (Global Mobile Information Simulator)

GLOMOSIM is a network simulator that simulates wireless and wired communication network system. GLOMOSIM is designed using parallel discrete event simulation capability provided by Parsec(Parsec is C based simulation language developed by Parallel computing laboratory developed by UCLA that supports parallel programming language).Currently GLOMOSIM supports only wireless protocols. GLOMOSIM can simulate thousands of node linked by heterogeneous communication network using ad-hoc network and traditional Internet Protocol. The table lists the GLOMOSIM layers

Physical(Radio Propagation)	Free space, Two-Ray
Data Link (MAC)	CSMA,MACA, TSMA,802.11
Network(Routing)	Bellman-Ford, FSR, OSPF,DSR,AODV
Transport	TCP,UDP
Application	Telnet, FTP

### Advantages

1. Supports implementation and simulation of wireless protocols
2. Layered architecture and each layer can be modeled separately according to the complexity.
3. GLOMOSIM Library is modular, extensible library for network model.
4. Built-in statistics collection at each layer
5. Coding is in C language and Parsec

### Disadvantages

1. No support for wired networks
2. Has only fixed protocol layers cannot add or delete Layers
3. No longer under active development

### Future Use

More researchers can contribute on this simulator so that more and more information and tutorial can be made available on this simulator. This work can be extended so that the simulator can also support wire network protocol.

## 7. JSIM

J-Sim is an application development environment based on the component-based software architecture, *Autonomous Component Architecture* or ACA. For network modeling and simulation generalized packet switched network model is defined on the top of ACA. This network model defines the generic structure of node in a network, components of the network and the base classes are used to implement the protocol across various layers. J-Sim has been developed entirely in [Java™](#). J-SIM is truly a platform independent, extensible and reusable environment. J-SIM has a script interface which allows the integration of different script language like Perl, Tcl and Python. J-Sim is a dual language simulation environment in which classes are written in Java and glued together using Tcl/java

**Advantages**

- JSIM has good reusability
- J-Sim contains large number of protocols; this simulator can also support data diffusions, routings and localization simulations in WSNs
- J-Sim provides a GUI library, which can help users to trace and debug programs
- J-Sim can simulate larger number of sensor nodes, around 500
- J-Sim can save lots of memory sizes.

- **J-Sim is relatively complicated to use**

- The execution time is much longer than that of NS-2.

**Future Work**

- Use JPF(Java Path Finder) to model-check the network protocols in J-Sim
- Compare the model checking framework in J-Sim with that of JPF
- Assess the pros and cons of building a model checker in J-Sim instead of using an existing model checker for Java programs such as JPF

**Limitations**

**Analysis and comparison of different Network Simulators**

Features	Glomosim	NS-2	NS-3	JSIM	Omnet++
<b>Language Supported</b>	Parsec C	C++/OTCL	C++	java	C++
<b>GUI support</b>	POOR	POOR	GOOD	GOOD	GOOD
<b>Time taken to learn</b>	Long	Long	Moderate	Moderate	Moderate
<b>Time take for download and installation</b>	Long time	Moderate time	Long time to download and install all necessary patches and supporting software.	Easy to download and install	Very easy takes very less time. Easily available
<b>Platform</b>	Linux, Windows	Linux, unix, Microsoft windows, Cygwin	Linux, unix, Microsoft windows,	Microsoft windows, linux, Matlab	Linux, unix, Microsoft windows, MAC OS
<b>Network Visualization tool</b>	It supports network visualization tool	It supports network visualization tool	It supports network visualization tool	It supports network visualization tool	It supports network visualization tool
<b>Availability of analysis tool</b>	It has analysis tool	It has analysis tool	It has analysis tool	It has analysis tool	It has analysis tool
<b>Crates trace file</b>	It creates trace file	It creates trace file	It creates trace file	It creates trace file	It creates trace file
<b>Possibility to design and modify the network scenarios</b>	It is Possible	It is Possible	It is Possible	It is Possible	It is Possible
<b>Design and Implementation Protocols</b>	Supports only wireless simulation of protocols	Supports both wired and wireless simulation of protocols	Supports both wired and wireless simulation of protocols	Supports both wired and wireless simulation of protocols	Supports both wired and wireless simulation of protocols
<b>Interaction with real time system</b>	It is Possible	It is Possible	It is Possible	It is Possible	It is Possible
<b>Fast simulation capabilities</b>	Poor	Moderate	Moderate	Poor	Moderate
<b>Merits</b>	-Supports implementation and simulation of wireless protocols - Layered architecture and each layer can be modeled separately according to the complexity. - GLOMOSIM Library is modular ,extensible library for network model. - Built-in statistics collection at each layer Coding is in C language and Parsec	-Easy to add new protocols. -A large number of protocols available publicly. -Availability of a visualization tool ns-3 Overview (June 2008). <a href="http://www.nsnam.org/docs/ns-3-overview.pdf">http://www.nsnam.org/docs/ns-3-overview.pdf</a> , June 2008.	-NS-3 is not an extension of NS-2; it is a new simulator. - NS-3 is open-source	-Provides support for energy modeling, with the exception of radio energy consumption - Support mobile wireless networks and sensor networks. - Component-oriented architecture	Powerful graphical User Interface (making tracing and bugging easier) - Simulate power Consumption problem

<b>De merits</b>	-No support for wired networks -Has only fixed protocol layers cannot add or delete layers -No longer under active development	-Supports only two wireless MAC protocols, 802.11, and a single-hop TDMA protocol. -Need to familiar with writing scripting language	-Python bindings do not work on Cygwin. -Only IPv4 is supported.	-Low efficiency of simulation. -The only MAC protocol provided for wireless networks is 802.11. - Unnecessary run-time overhead	-Number of protocol is not large enough. - Compatibility problem (not portable)
------------------	--	--	---	---	---

## 8. Conclusion

In this paper we have investigated five different open source network simulating tools. In the survey it was found that ns3 and Omnet++ are capable of carrying large scale simulation in an efficient way. Omnet++ supports a high GUI capability but it lacks in performance with respect to ns3. Overall performance among the five simulator ns3 gives the best performance but it is at the early stage and has only few simulation models. As the rich collection of models for ns-2 still needs to be ported from ns-2 to ns-3. In conclusion, the question of which simulator to use is a difficult one, and the answer is largely dependent on the specific use case. However, if scalability is the main concern ns-3 and Omnet++ are smart choices.

## 9. Future Work

Researchers spend lots of time in first downloading and installing the simulators. Researcher should learn the particular programming language for particular simulators like tcl for ns2, java for jsim and c++ and python for ns3. It takes lot of time for the researcher to understand the working of simulator itself. Hence after doing a survey on various simulator I felt that research community strongly need a open simulator that supports more than five programming language like python, java, c++, c or c# or tcl etc that gives researcher the freedom to choose the programming to implement his algorithm. Simulator should support good or high quality GUI and should be easy to learn and easy to download and install and should take less time to install. Results produced through the simulator should match the real time environment.

## References

- [1] Jianli Pan, Prof. Raj Jain, Project, "A Survey of Network Simulation Tools: Current Status and Future Developments", report.
- [2] Mrs. Saba Siraj, Mr. Ajay Kumar Gupta, Mrs Rinku Badgajar, "Network Simulation Tools Survey", International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 4, June 2012, ISSN :2278 – 1021
- [3] Karsten M. Reineck. "Evaluation and Comparison of Network Simulation Tools", Master Thesis. 29th August 2008
- [4] Analysis and comparison of different network simulators Vinita Mishra<sup>1</sup>, Smita Jangale<sup>2</sup>
- [5] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, "J-Sim: A Simulation Environment for Wireless Sensor Networks", Proceedings of the 38th Annual Simulation Symposium (ANSS'05), 2005 IEEE
- [6] [NS2] NS2 official website, <http://www.isi.edu/nsnam/ns/>
- [7] [NS2-wiki] NS2 resource webpage, [http://nsnam.isi.edu/nsnam/index.php/Main\\_Page](http://nsnam.isi.edu/nsnam/index.php/Main_Page)
- [8] [NS3] NS3 official website, <http://www.nsnam.org/documents.html>
- [9] [OMNeT] OMNeT++ official website, <http://www.omnetpp.org/>
- [10] M. Greis. NS-2 tutorial. Retrieved January 5, 2011, from <http://www.isi.edu/nsnam/ns/tutorial/index.html>
- [11] <http://sites.google.com/site/jsimofficial/j-sim-tutorial>
- [12] <http://cobweb.cs.uga.edu/~jam/jsim/>
- [13] <http://j-sim.cs.uiuc.edu/>
- [14] NS by Example. <http://nile.wpi.edu/NS/>
- [15] [http://en.wikipedia.org/wiki/Network\\_simulation](http://en.wikipedia.org/wiki/Network_simulation)
- [16] Global Mobile Information System Simulator in Fedora Linux  
Ayyaswamy Kathirvel a,  
Rengaramanujam Srinivasan b
- [17] András Varga, Rudolf Hornig, AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT, SIMUTools, March 03 –07, 2008, Marseille, France
- [18] Andras Varga. Omnet++ discrete event simulation system – user manual. <http://www.omnetpp.org/doc/manual/usman.html>, 2005
- [19] ns-3 Overview (June 2008). <http://www.nsnam.org/docs/ns-3-overview.pdf>, June 2008.
- [20] A performance comparison of recent network simulators Elias Weingartner, Hendrik vom Lehn and Klaus Wehrle
- [21] <http://en.wikipedia.org/wiki/GloMoSim>
- [22] GloMoSim: A Scalable Network Simulation Environment Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Ken Tang, Rajive Bagrodia, Mario Gerla Computer Science Department University of California, Los Angeles Los Angeles, CA 90095
- [23] [http://people.ac.upc.edu/amir/index\\_files/Glomosim.pdf](http://people.ac.upc.edu/amir/index_files/Glomosim.pdf)
- [24] [www.nsnam.org/docs/manual/ns-3-manual.pdf](http://www.nsnam.org/docs/manual/ns-3-manual.pdf)
- [25] [http://www.isi.edu/nsnam/ns/doc/ns\\_doc.pdf](http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf)

## APPENDIX

Sample tutorials on OMnet++

1. Create a working directory called tictoc, and change to this directory
2. Describe your example network by creating a topology file
3. Implement the functionality
4. Create the Makefile
5. Compile and link the simulation
6. Create the omnetpp.ini file
7. Start the executable using graphical user interface
8. Press the Run button on the toolbar to start the simulation
9. You can play with slowing down the animation or making it faster
10. You can exit the simulation program ...

### 1. Step 1(windows Os)

Double click on file mingwenv.cmd

	Makefile	3/11/2014 5:37 PM	File	12 KB
	Makefile.inc	5/23/2014 4:19 PM	INC File	5 KB
	Makefile.inc.in	3/7/2014 5:37 PM	IN File	5 KB
	MIGRATION.txt	3/7/2014 5:38 PM	Text Document	17 KB
	mingwenv.cmd	3/7/2014 5:38 PM	Windows Comma...	1 KB
	README.txt	3/7/2014 5:38 PM	Text Document	4 KB
	setenv	3/7/2014 5:37 PM	File	1 KB
	Version	3/7/2014 5:37 PM	File	1 KB
	WHATSNEW.txt	3/7/2014 5:38 PM	Text Document	149 KB

### 2. Command window will open type omnetpp

```

MINGW32:~
Dell@Dell-PC ~/samples/project
$ make
project.cc
project.cc:26:6: error: 'Txc23' has not been declared
project.cc: In function 'void initialize()':
project.cc:33:31: error: 'getName' was not declared in this scope
project.cc:38:24: error: 'send' was not declared in this scope
project.cc: At global scope:
project.cc:42:6: error: 'Txc23' has not been declared
project.cc: In function 'void handleMessage(cMessage*)':
project.cc:48:20: error: 'send' was not declared in this scope
make: *** [out/gcc-debug//project.o] Error 1

Dell@Dell-PC ~/samples/project
$ cd ..

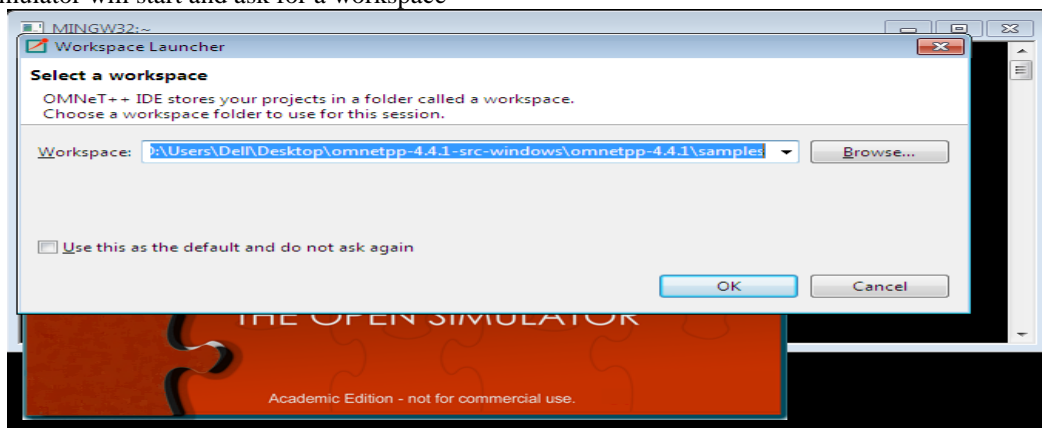
Dell@Dell-PC ~/samples
$ cd..
sh: cd..: command not found

Dell@Dell-PC ~/samples
$ cd ..

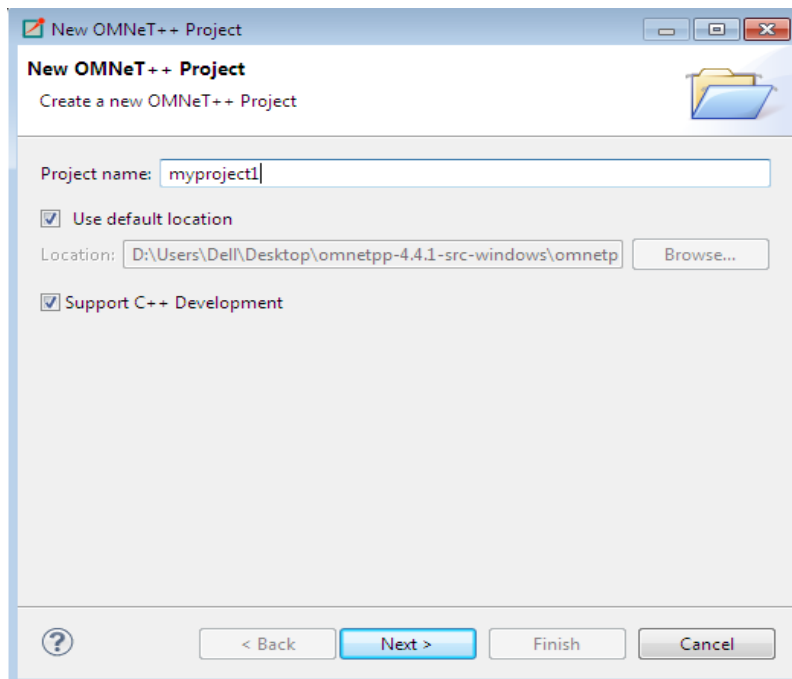
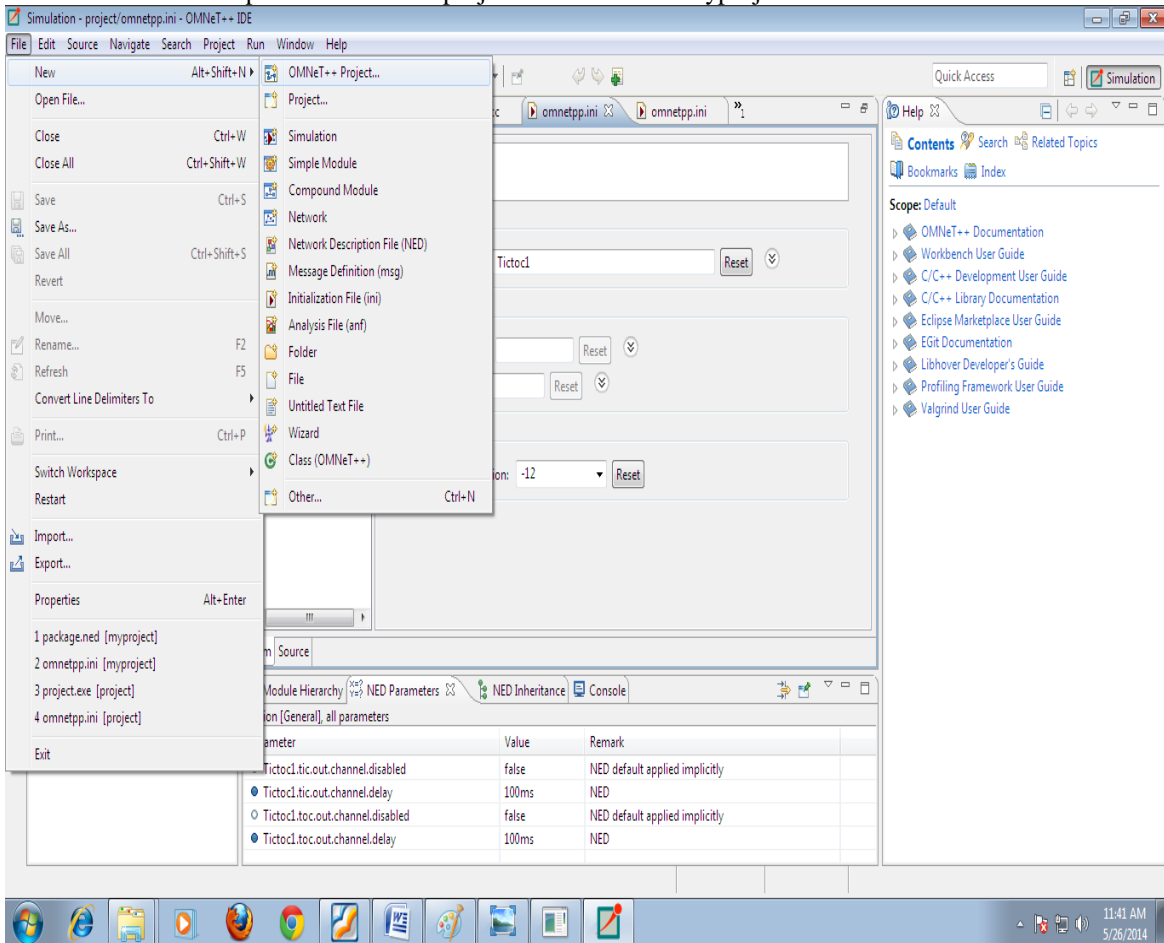
Dell@Dell-PC ~
$ omnetpp

```

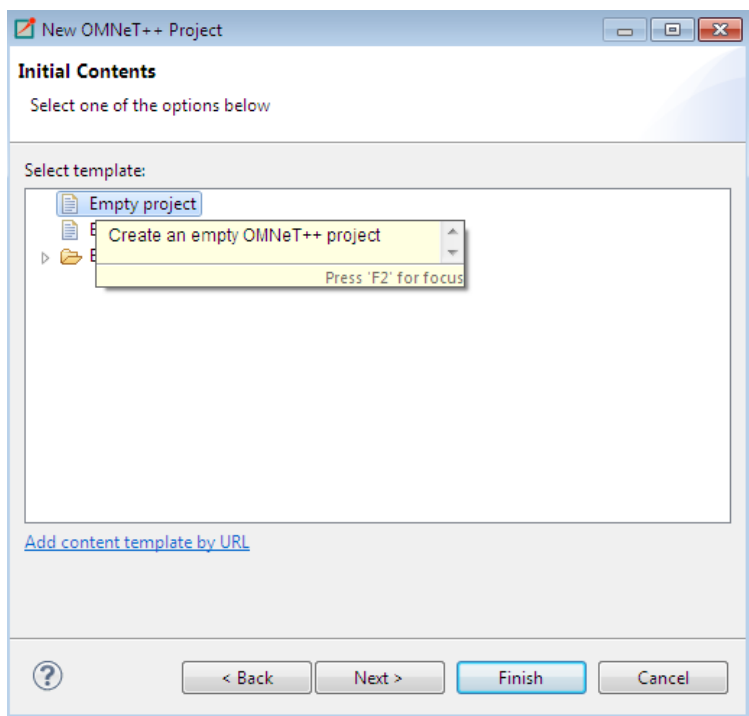
### 3. Omnet simulator will start and ask for a workspace



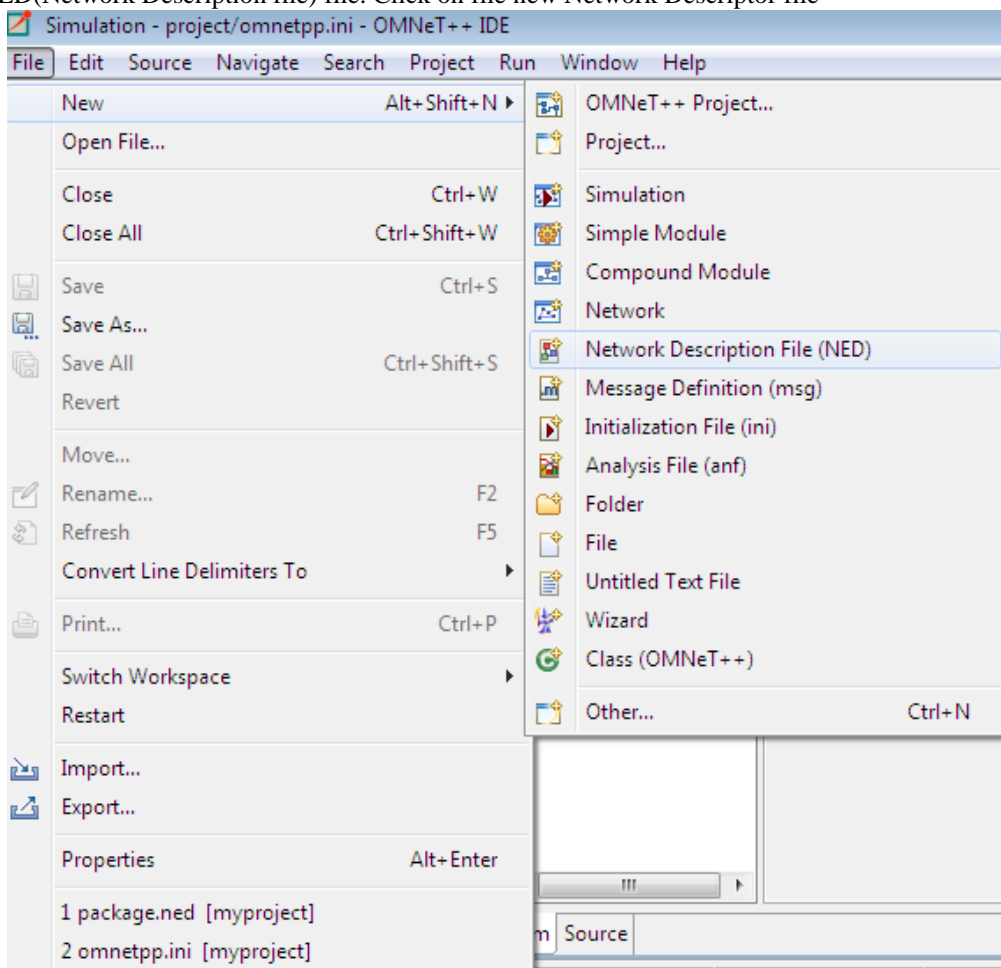
4. OMNet++ simulator will open. Create a new project and name it as myproject1



5. Click on next and then click on project



6. Create a NED(Network Description file) file. Click on file new Network Descriptor file





7. Type the following code in newly created .ned file

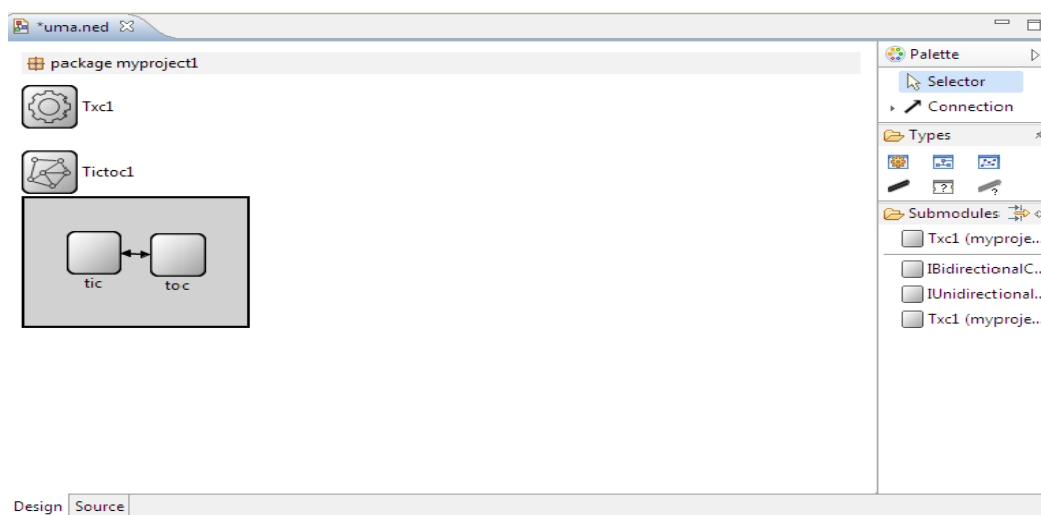
```

// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

simple Txc1
{
    gates:
        input in;
        output out;
}

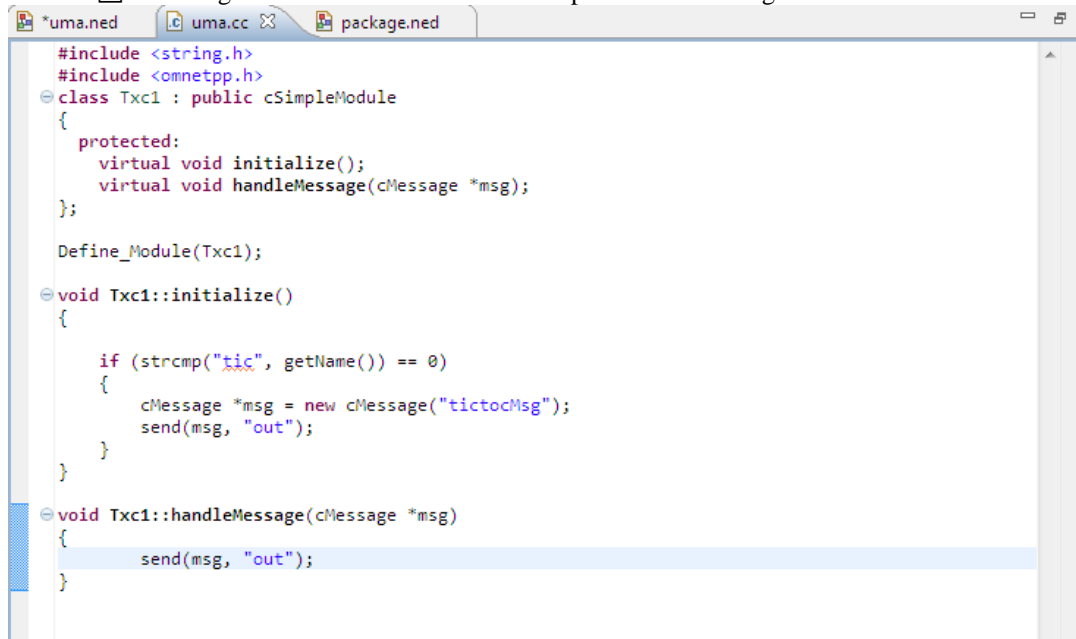
//
// Two instances (tic and toc) of Txc1 connected both ways.
// Tic and toc will pass messages to one another.
//
network Tictoc1
{
    submodules:
        tic: Txc1;
        toc: Txc1;
    connections:
        tic.out --> { delay = 100ms; } --> toc.in;
        tic.in <-- { delay = 100ms; } <-- toc.out;
}
    
```

8. When you click on design you will see the following



**9. Create a .cc file**

Click on File-> New File and give file name with extension.cc paste the following code in .cc file

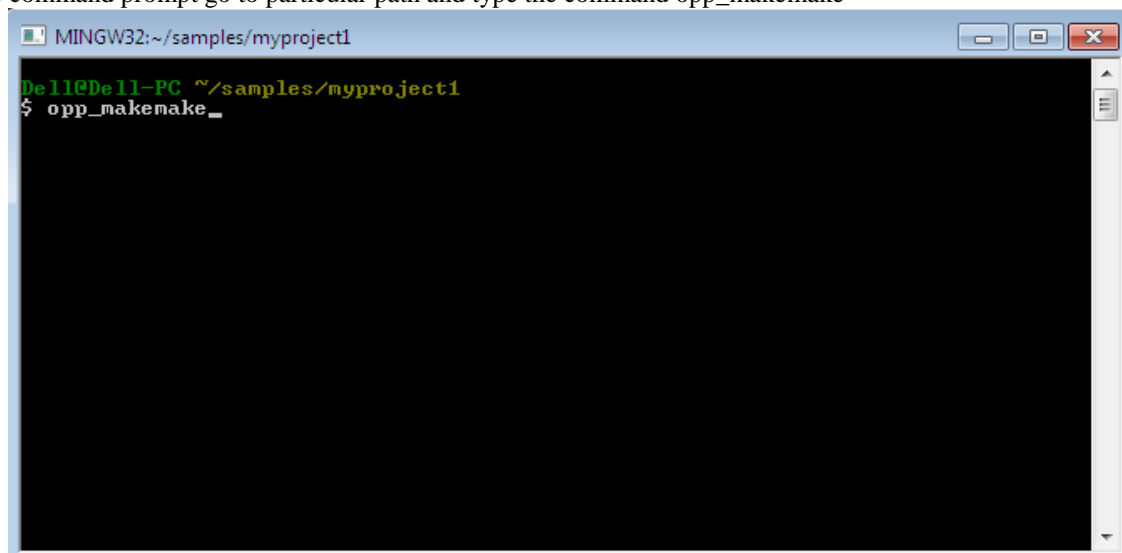


```
#include <string.h>
#include <omnetpp.h>
class Txc1 : public cSimpleModule
{
protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

Define_Module(Txc1);

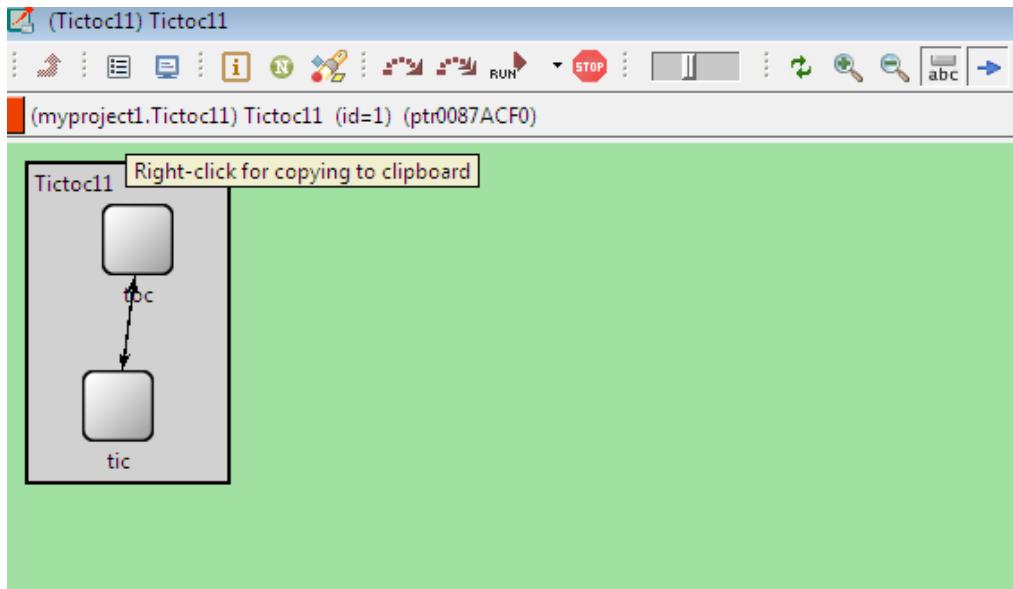
void Txc1::initialize()
{
    if (strcmp("tic", getName()) == 0)
    {
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}

void Txc1::handleMessage(cMessage *msg)
{
    send(msg, "out");
}
```

**10. Go to command prompt go to particular path and type the command opp\_makemake**

```
MINGW32:~/samples/myproject1
Dell@Dell-PC ~/samples/myproject1
$ opp_makemake_
```

**11. Type the command make****12. Create a .ini Click on file New the click New initialization .ini file****13. Then type ./myproject1 n in the command prompt to run the your simulation**



## Sample tutorials on NS2

### TCL script for the topology

1. Open a note pad paste this code in it and save it with out.tcl

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
#Queue/DropTail set drop_front_ true
set nf [open out.nam w]
$ns namtrace-all $nf
set tf [open out.tr w]
$ns trace-all $tf
proc finish {} {
    global ns nf tf
    $ns flush-trace
    close $nf
    close $tf
    exec nam out.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
```

Volume 3 Issue 12, December 2014

[www.ijsr.net](http://www.ijsr.net)

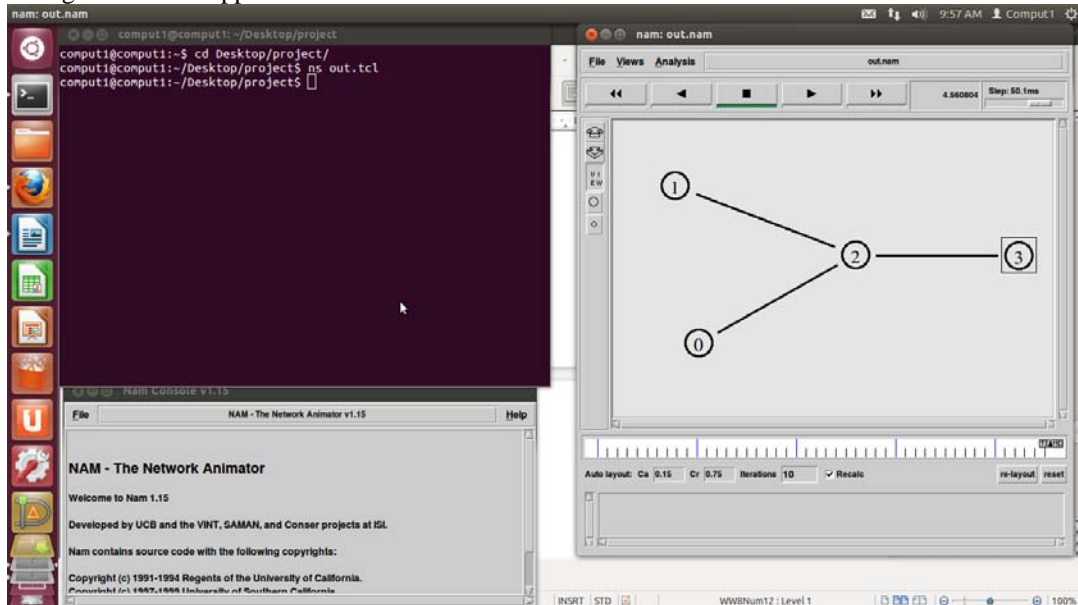
[Licensed Under Creative Commons Attribution CC BY](http://www.ijsr.net)

```

$ns attach-agent $n3 $null0
$udp0 set class_ 1
$udp1 set class_ 2
$ns connect $udp0 $null0
$ns connect $udp1 $null0
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run

```

- To run the ns file type command **ns out.tcl**
- Following Screen will appear



- Create an awk file to display number packet dropped with the name drop.awk

```

BEGIN {
#Initialization of the parameters can be done here
count=0;
}
{
#Main logic of the program goes here
if($1 == "d")
count++;
}
END {
#Printing the results can be done here
printf("Total Number of Dropped Packets =%d\n", count);
}

```

- Run the awk file by typing following command  
**awk-f drop.awk out.tr**

```

comput1@comput1: ~/Desktop/project
comput1@comput1:~$ cd Desktop/project/
comput1@comput1:~/Desktop/project$ ns out.tcl
comput1@comput1:~/Desktop/project$ awk -f drop.awk out.tr
Total Number of Dropped Packets =435
comput1@comput1:~/Desktop/project$ █

```

6. To plot the graph give the following create a awk file with a name graph.awk

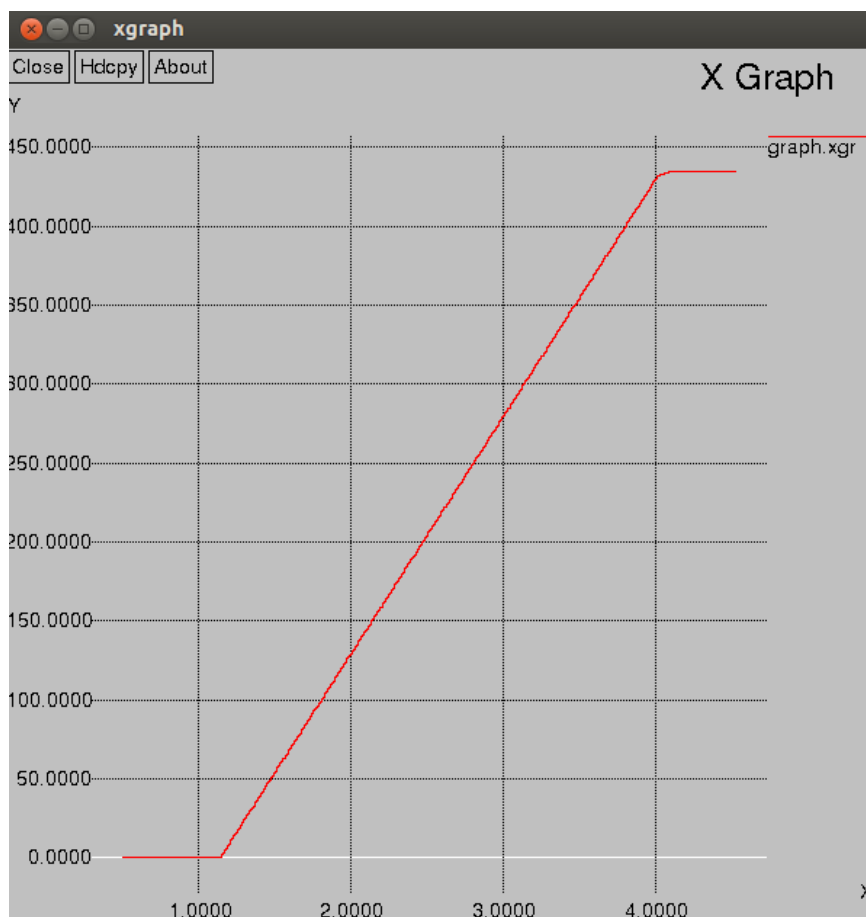
```
BEGIN {
#Initialization of the parameters can be done here
count=0;
}
{
#Main logic of the program goes here
if($1 == "d")
count++;
printf("%f\t%d\n", $2, count)
}
END {
}
```

7. Run the file by giving command `awk -f graph.awk out.tr > graph.xgr`

```
comput1@comput1: ~/Desktop/project
comput1@comput1:~/Desktop/project$ awk -f graph.awk out.tr > graph.xgr
comput1@comput1:~/Desktop/project$
```

8. Draw the graph by giving command `xgraph graph.xgr`

```
comput1@comput1: ~/Desktop/project
comput1@comput1:~$ cd Desktop/project/
comput1@comput1:~/Desktop/project$ xgraph graph.xgr
```



Sample tutorial for

NS3

First.cc

Volume 3 Issue 12, December 2014

[www.ijsr.net](http://www.ijsr.net)

Licensed Under Creative Commons Attribution CC BY

```

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
int
main (int argc, char *argv[])
{
    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    NodeContainer nodes;
    nodes.Create (2);
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
    NetDeviceContainer devices;
    devices = pointToPoint.Install (nodes);
    InternetStackHelper stack;
    stack.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (devices);
    UdpEchoServerHelper echoServer (9);
    ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
    serverApps.Start (Seconds (1.0));
    serverApps.Stop (Seconds (10.0));
    UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
    echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
    echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
    echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));
    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
    clientApps.Start (Seconds (2.0));
    clientApps.Stop (Seconds (10.0));
    Simulator::Run ();
    Simulator::Destroy ();
    return 0;
}

```

To run use the command: `./waf --run first`

```

root@umapujer: /home/umapujeri/ns-allinone-3.21/ns-3.21
root@umapujer: /home/umapujeri/ns-allinone-3.21/ns-3.21# ./waf --run first
Waf: Entering directory `/home/umapujeri/ns-allinone-3.21/ns-3.21/build'
Waf: Leaving directory `/home/umapujeri/ns-allinone-3.21/ns-3.21/build'
'build' finished successfully (5.858s)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
root@umapujer: /home/umapujeri/ns-allinone-3.21/ns-3.21#

```

### Sample

#### Tutorial for JSIM

To run J-SIM

1. Type "ant" (C:\jsim-1.3>ant).

```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\exam>cd\
C:\>cd jsim-1.3
C:\jsim-1.3>ant
Buildfile: C:\jsim-1.3\build.xml

help:
[echo] DRCL J-Sim v1.3 build file
[echo]
[echo] Available targets are:
[echo]
[echo]   run      --> runs the J-Sim
[echo]   compile  --> compiles the source files
[echo]   clean    --> cleans up J-Sim binaries
[echo]   cleanall --> cleans up all the binaries
[echo]   javadoc  --> generates the API documentation
[echo]

BUILD SUCCESSFUL
Total time: 0 seconds
C:\jsim-1.3>
```

2. Type "ant compile" (C:\jsim-1.3>ant compile).

```
C:\WINDOWS\system32\cmd.exe
C:\jsim-1.3>ant compile
Buildfile: C:\jsim-1.3\build.xml

init:
[echo] 26-November-2014 05:09 PM

compile:
[javac] C:\jsim-1.3\build.xml:75: warning: 'includeantruntime' was not set,
defaulting to build.sysclasspath=last; set to false for repeatable builds

BUILD SUCCESSFUL
Total time: 2 seconds
C:\jsim-1.3>
```

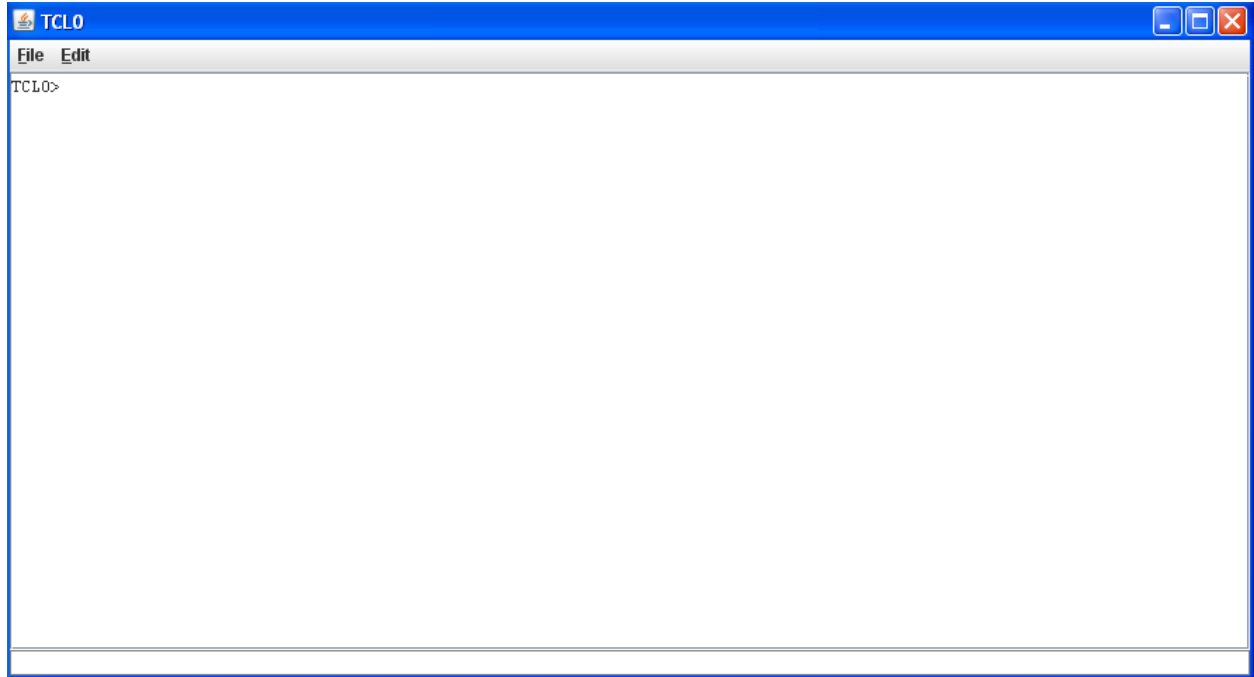
3. Finally, type "ant run" (C:\jsim-1.3>ant run).

```
C:\WINDOWS\system32\cmd.exe - ant run
C:\jsim-1.3>ant run
Buildfile: C:\jsim-1.3\build.xml

init:
[echo] 26-November-2014 05:12 PM

compile:
[javac] C:\jsim-1.3\build.xml:75: warning: 'includeantruntime' was not set,
defaulting to build.sysclasspath=last; set to false for repeatable builds

run:
```



4. To Print Hello World in JSIM, type the following commands in the terminal

```
set f [java::new java.awt.Frame "Tcljava"]
set l [java::new java.awt.Label "Hello World!"]
$l setFont [java::new {java.awt.Font String int int} "TimesRoman" [java::field java.awt.Font BOLD] 40]
$f add $l [java::field java.awt.BorderLayout CENTER]
$f pack
$f show
```

