FlexiSim: A user friendly Environment for Network Simulations

A. Harshit Agarwal[1], **B. Antriksh Saxena**[1], **C. P. Venkata Krishna** [1], **and D. V.Saritha**[1]
[1]School of Computing Science and Engineering, VIT University, Vellore, Tamil Nadu, India

Abstract - *NS2 is a discrete event simulator targeted at networking research. NS2 provides substantial support for simulation of routing protocols over wired and wireless (local and satellite) networks. NS2 uses TCL script for its front end and C++ as its back end [1,5]. NS2 is completely command based and one needs to create a TCL file for running any network simulation. Therefore, it's very tedious for a user to go through all the syntaxes and semantics of the required commands. This paper presents the development of a new graphical network simulator named FlexiSim which is based on NS2 [1]. Additionally, in this paper we discuss the design of FlexiSim and the implementation of a routing protocol through it. FlexiSim remarkably simplifies the procedure involved in running the simulations on NS2, by providing an interactive and flexible environment to the user therefore making it user friendly. No simulator existed for the code generation of NS2 prior to the design of FlexiSim. The most important advantage of FlexiSim is that it is a simple simulator that can be used for performing simulation exercises of moderate complexity under realistic network characteristics.*

Keywords: FlexiSim, NS2, Wireless Networks, Simulation, Simulator Development

1 Introduction

A network simulator is a software program that suggests the working of a computer network. In network simulators the network is modeled and it is generally used to analyze the performance. Using actual test bed by setting up routers, computers and data links may involve high cost and time. Simulators emulate the real world scenario therefore provide relatively inexpensive and fast way. A popular example of a network simulator is NS2 which is used in the simulation of routing protocols and it is heavily used in ad-hoc networking research. NS2 is an open source model with online documentation. However, modeling is a very complex and time consuming task in NS2 since it has no GUI [1].

In this paper, we report the development of a new NS2 based simulator, FlexiSim, which is built to analyze different wireless protocols in a more flexible way. One of the major problems with simulating wireless networks on NS2 is the complexity involved in using NS2 [1]. It is extremely challenging for the user to first go through the complete working of NS2 [1] then write the protocol and topology. FlexiSim was developed with extensibility in mind. In this paper, we discuss, as well, the issues involved in the implementation of AOR-GLU [2] using FlexiSim. This serves to demonstrate a case of the use of FlexiSim to perform wireless simulations.

A variety of network simulators (e.g., GloMoSim [3] and OMNeT++ [4]) is already available and is widely used by the industry and the academia for various purposes. However, use of these simulators by a novice may turn out to be a tough job for him because of the perplexity involved. Since working with NS2 is not an easy task, a user has to go through all the commands and the available functionalities before simulating even a simple network topology. So the primary purpose of FlexiSim is to simplify this process, make it faster, flexible and more robust. Instead of memorizing all the syntaxes and semantics of the commands the user just needs to have a basic overview of the topology and the protocols, the TCL code [5] used to implement this simulation will be produced by FlexiSim. FlexiSim incorporates its own code verifying mechanism and checks for the validity of the input given by the user, thereby preventing the generation of erroneous code. It consists of various menu options through which the user can create new nodes, agents, applications, links etc and configure them as well. The application automatically generates the TCL code [5] for the input given by the user. When this code is run in this application, FlexiSim automatically calls NS2 which generates the trace (.tr) file [6] and .nam file [6]. These generated files are saved in the project folder and can be used later for further analysis. Once .nam file is generated the FlexiSim opens up the NAM (Network Animator) [6] and runs the file on it, so that user can visualize the simulation. Tracegraph, a free network trace analyzer for NS2 can be launched from FlexiSim with the help of which user can make scrupulous study of the results of the simulation. For FlexiSim to work, it is necessary that the target machine should be installed with ns-2.34 [1] and nam-1.14[6]. At present FlexiSim 1.0 is designed only for wired and wireless topologies (local).

FlexiSim is developed on a Linux platform. Most of development was carried out on Ubuntu 10.04 [7], but the software was so designed that it could be run on all variants of Linux and UNIX systems. NetBeans 6.9.1 [8] development environment was used for rapid and organized development. NetBeans offered unparallel support for development and debugging in Java [11], the language used for development. FlexiSim is a standalone Java based desktop application which provides a GUI for NS-2 [1].

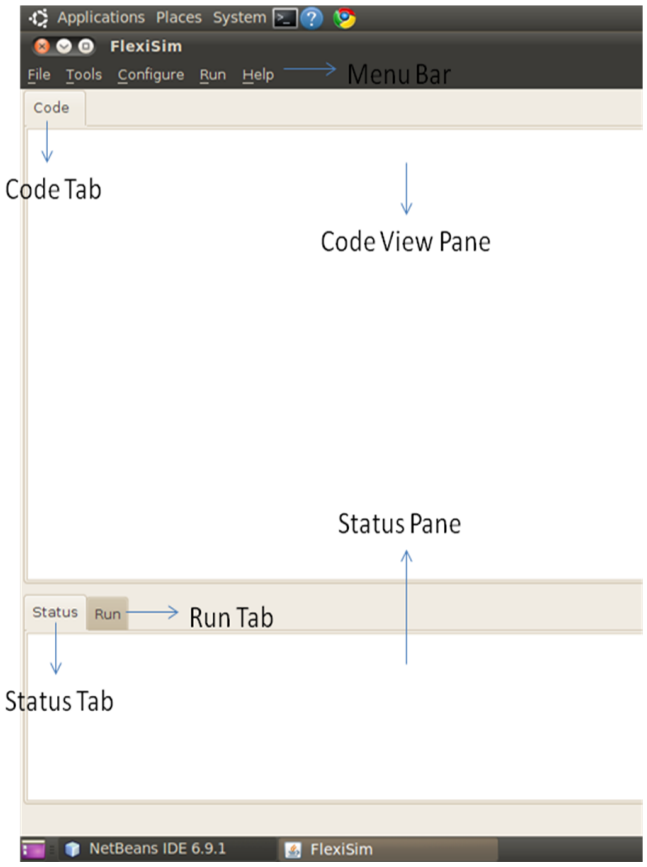

Figure 1. FlexiSim screenshot

2 Motivation

Before starting the development of FlexiSim we analyzed various available network simulators as mentioned above. Among them NS2 is the most commonly used and trusted simulator. Compared to its contemporaries, NS2 provides simulation for almost all types of wired and wireless networks [12]. But there is a major usability issue with NS2 as it involves the use of two languages one is C++ for modelling the behaviour of simulation nodes and other one is oTcl script that controls the simulation and create network topologies.

FlexiSim was designed keeping in view the difficulty in employing NS2 for simulations. Firstly, before using NS2, user has to comprehend its working and he should have an overview of its architecture. Secondly, he has to learn TCL for creating network topologies and C++ for creating protocols [1,5,6].

In spite of these issues, NS2 is widely used for simulations because of its accuracy in simulating wireless networks. Therefore FlexiSim was designed to use NS2 at the back end. At the front end it provides user with a rich GUI where even a naïve user can perform simulations and get the same results as he would get by using NS2 [1].

3 Development of FlexiSim

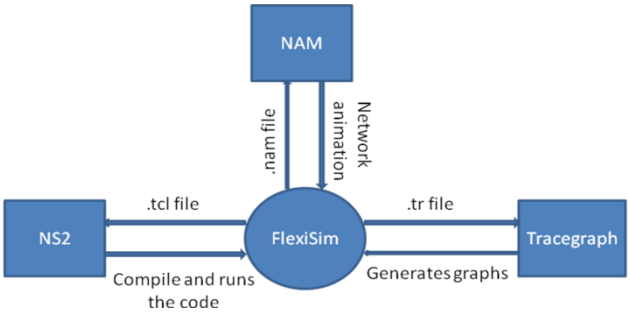

Figure 2. Interactions of FlexiSim with other programs

As mentioned above FlexiSim is developed with extensibility and scalability in mind. It was developed using Java [11] on Ubuntu 10.04 [7]. The design choices and the programming structure were also kept flexible and extensible to make the simulator easy to use. The simulator has been designed keeping in mind the problems the end users might face while writing the TCL code.

3.1 Platform Overview

The simulator was developed using Java on Ubuntu 10.04, but it is targeted for all the operating systems. The main reason behind choosing a Linux based platform is because of the NS2.34, NAM, and various trace analyzers, all of these are compatible with Linux [1,6,7,9].

An object oriented approach has been followed to develop this software. The cause behind using Java is it being completely object oriented and platform independence. Because of the platform independence offered by Java later versions of FlexiSim can be made compatible with windows too. The whole project was developed on NetBeans IDE 6.9.1. NetBeans provides a simplified development of the Java Swing Desktop application. It also has a Java Swing GUI builder (formerly known as "Project Matisse") which provides an interactive and rapid way of creating GUIs [8].

3.2 Code Organization

For the development of FlexiSim through Netbeans IDE [8] the code is distributed into various packages. The packages were formed on the basis of the functionality that

they provide. Each of them holds a set of classes pertinent to it. For Example, there is a package named FlexiSim.node. This package contains all the classes related to nodes. Likewise there are packages for other entities too.

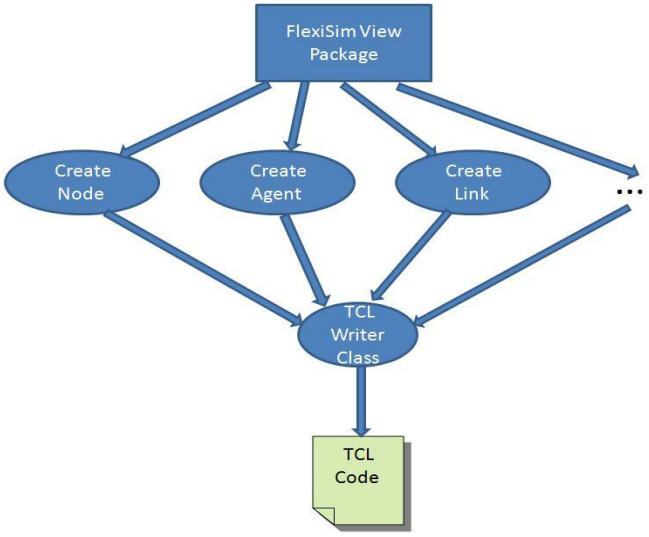

Figure 3. FlexiSim Package

As shown in Figure 2 there is a package called the FlexiSim View Package. This package contains the classes of all the entities that can be either created or configured through FlexiSim. For example, there is a class for creating nodes, creating agents and many more. All these classes call the object of TCL Writer class for writing the respective TCL code. In the TCL Writer class there are various methods for creation or configuration of each entity. As the user creates or configures any property of an entity the corresponding method in the TCL Writer class is called for generating the TCL code [5].

Suppose the user is creating a node. When the user clicks the Create Node button, an object of the Create Node class in FlexiSim View package would be called. This object would take in the parameters associated with node and then it would call the Node Write method in the TCL Writer class. This method would write the TCL code with the provided parameters into the TCL file. Similarly it works for other entities too [5].

4 Simulation Development through FlexiSim

It is very easy to develop a network simulation through FlexiSim. In order to maintain the brevity of the paper a simple wired simulation is explained. In general, to create a wired topology the following steps are to be followed.

1) Create a new project by specifying the project name and the project location. Also select the type of simulation to be created as wired.

2) After the project has been created nodes are created. While creating nodes there are two options, either the user can create a single node at a time or multiple nodes can be created at one go. While creating nodes the user needs to specify the node names.

3) Then links are created between the nodes that were made. The link type can be simplex or duplex link. Also properties of the link like bandwidth, delay and queue type need to be specified.

4) Agents can be created at different nodes. While creating agents the agent name, type and the node on which it is to be created is specified.

As an agent is created a new dialog automatically pops up regarding the application that is to be created. For application the name, type and its start and finish time are specified. Certain properties (if any) of application like packet size and bandwidth in case of CBR (Constant Bit Rate) are also specified.

5) After the agents and the application are created, the agents can be connected. Whatever agents that were created are shown in the drop down box, and the user can choose the ones which are to be connected.

6) Subsequently the events are scheduled. In event scheduling there are options for specifying the up-link, down-link time of the links as well as the detach time of the agents.

7) The various properties of the nodes such as routing protocol can be configured according to the requirements of the topology.

8) The above steps successfully create a topology. The next step is to run the simulation. In the run dialog the running time of the simulation needs to be specified.

As the simulation is run, FlexiSim automatically calls the NS2 which in turn generates the trace (.tr) file and the .nam file in the project folder which was created in Step 1. FlexiSim then calls NAM which is used for the visualisation of the simulation. Also the tracegraph can be called to analyse the performance of the simulation [6][9].

5 Case Study

In this section, we show how FlexiSim was used to simulate a routing protocol such as AOR-GLU [2]. Intentionally many implementation details were provided, while maintaining the conciseness of the paper, so that the readers could use these ideas from the simulation study and simulate experiments using FlexiSim for solving their own problems.

A typical addition of any new protocol in NS2, through FlexiSim, starts with modifying few base files in the NS2 installation folder.

5.1 Modification of NS2 Files

AOR-GLU [2] is an AODV [10] based protocol. To implement AOR-GLU [2] protocol in NS2 the existing AODV [10] protocol files are to be modified.

Apart from modifying basic protocol files to implement any protocol in NS2 [1] we need to modify NS2 [1] base files. The selection of the files and the modifications to be done in them are totally dependent upon the protocol being implemented. There is no standard as such suffice to which we can make the changes in the files. AOR-GLU [2] is a routing protocol for MANET and generally for implementing ad hoc based routing protocols some of the NS2 [1] files are required to be changed.

Once the protocol files and the above mentioned files are modified, NS is rebuilt to incur all the changes. To simulate the protocol a topology is created so that its performance can be analyzed. This analysis can be further used to compare the newly implemented protocol with the existing ones.

In the simulation 100 nodes were randomly distributed in an area of 500m X 400m. The node starting positions, destinations, and travel speeds were set random. Simulation duration was set to 150 seconds.

5.2 Simulation Sequence of Protocol Through Flexisim

Use The implementation of AOR-GLU [2] protocol through FlexiSim is explained below in steps.

1) For creating any new simulation or creating or modifying a protocol, through FlexiSim, firstly it's required to create a new project. While creating a new project the user is prompted to fill the project name, choose its location and select the project type (i.e. wireless or wired). In this case we chose the project type as wireless.

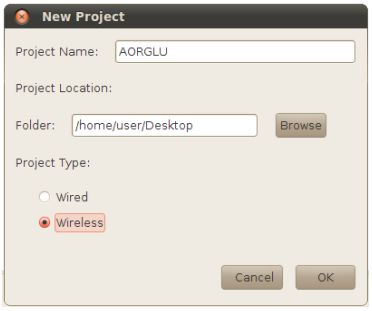

Figure 4. New Project

2) After the project was successfully created, the Modify Existing Protocol (because AOR-GLU [2] is implemented by modifying the AODV [10] protocol) option, present in the Tools menu, was selected. This launched a directory chooser through which the location of the protocol to be modified was selected. By default the path of the directory chooser is set to that of the NS2 installation folder.

After the protocol directory was chosen (in this case AODV) FlexiSim opened all the .cc and .h files, present inside the protocol folder, in different tabs. Along with these files the base NS files, which were to be modified as mentioned above, were also opened.

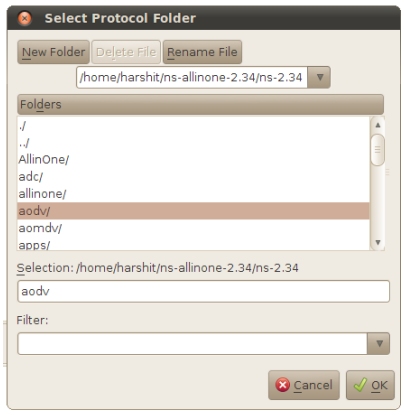

Figure 5. Select Protocol Folder

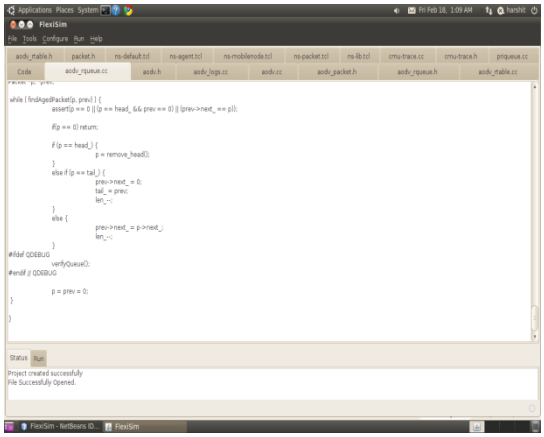

Figure 6. Opened Tabs

3) The files opened by FlexiSim in various editable tabs were modified as per the requirements of the AOR-GLU [2] protocol.

4) The project was saved so that the modifications are written in the protocol files. All the changes done in these files were performed in the copy of the original protocol which was present in the project folder.

5) The MakeNS option was selected from the Run menu. This option cleans and then rebuilds NS in order to incorporate the modifications in NS2. During this procedure a backup of the existing NS is also created in the project folder and the original protocol files and NS base files are replaced by the modified ones [1].

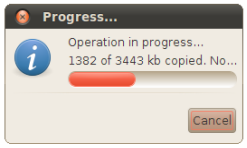

Figure 7. Backup creation in progress

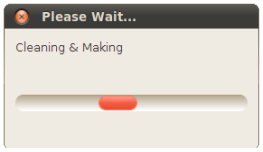

Figure 8. Clean and make operation in progress

6) A topology with 100 nodes was created by selecting the New Node(s) option from Tools menu.

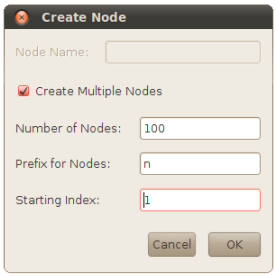

Figure 9. Multiple node creation

7) Configuration of the node was set using Node option from Configure menu. The following node configuration was set.

```
Channel/WirelessChannel     ;# channel type
Propagation/TwoRayGround    ;# radio-propagation
model
Phy/WirelessPhy             ;# network interface type
Mac/802_11                  ;# MAC type
Queue/DropTail/PriQueue     ;# interface queue type
LL                          ;# link layer type
Antenna/OmniAntenna         ;# antenna model
50                          ;# max packet in ifq
100                         ;# number of mobilenodes
AODV                        ;# routing protocol
ON                          ;# agentTrace
ON                          ;# routerTrace
OFF                         ;# macTrace
ON                          ;# movementTrace
```

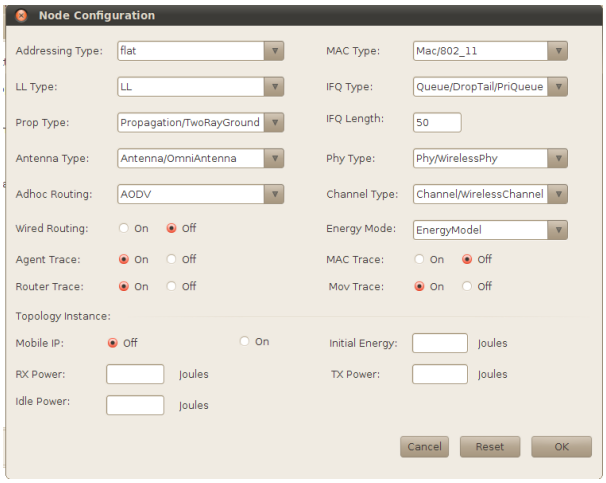

Figure 10. Node configuration

Configuration of the node was set using Node option from Configure menu. The following node configuration was set.

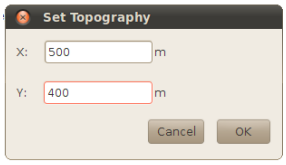

Figure 11. Set topography

8) The position of the nodes was set randomly so as to distribute the nodes throughout the topography.

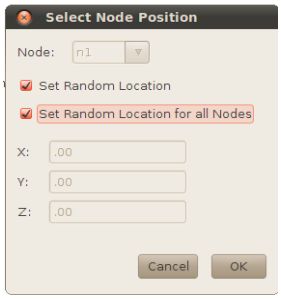

Figure 12. Node position

9) The movement of the nodes was set with random speed and destination.

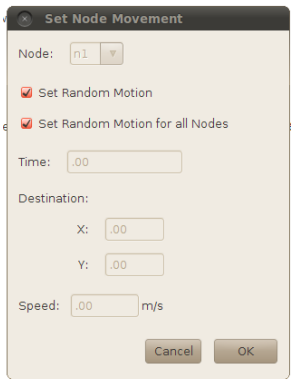

Figure 13. Node movement

10) Two agents, TCP and TCPSink, were created on two different nodes and a connection was established between the two. A FTP application was defined on the node with TCP agent where the start time was set as 10s. Later these two agents were connected.

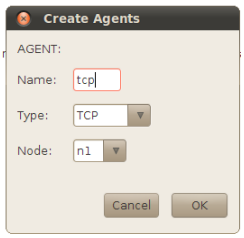

Figure 14. Creation of TCP agent

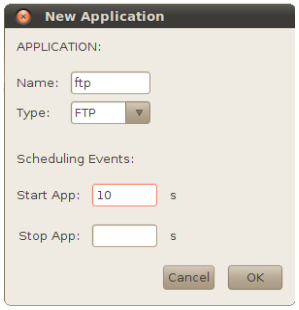

Figure 15. Creation of new FTP application

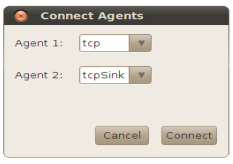

Figure 16. Connection of TCP and TCPSink agents

11) Once the creation of topology was complete, the simulation was run by selecting the Run option from the Run menu. The finish time was specified as 150s.

Figure 17. Run

12) To further analyze the protocol the open trgraph was selected from the Run menu. The trace file for the above simulation was selected and subsequently the tracegraph [9] was launched.

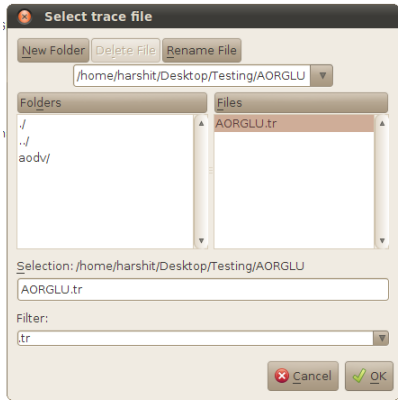

Figure 18. Selection of trace file for tracegraph

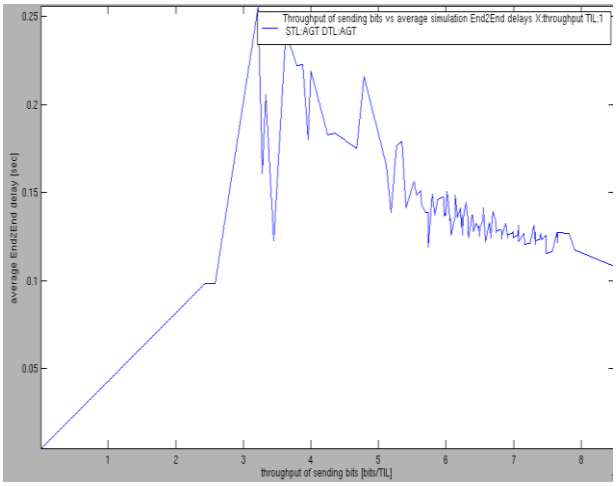

Figure 19. Tracegraph of the above simulation between throughput and average End2End delay

6 Conclusions

This paper gives a detailed explanation on design and implementation of a new NS2 based simulator, FlexiSim. FlexiSim is one of its kinds as it offers the user a flexible interface for creating topologies, creating new protocols in NS2 as well as modifying the existing protocols. FlexiSim not only makes easier for the user to run simulations on NS2 but also provides them with a faster and effective way of doing this.

In the future we plan to automate the modifications that are needed in creating a new protocol and also make the TCL file code pane editable. For the better evaluation of protocols we plan to include the feature of comparison between the different protocols with the help of suitable trace analyzer FlexiSim is available on request on http://www.flexisim.org.

This website provides FlexiSim along with its manual and tutorials.

7 References

[1] The Network Simulator – ns-2, 2002. http://www.isi.edu/nsnam/ns/.

[2] Chia-Chang Hsu; Chin-Laung Lei; "A Geographic Scheme with Location Update for Ad Hoc Routing" in Systems and Networks Communications, 2009. ICSNC '09. Fourth International Conference on vol., no., pp. 43-48, 20-25 sept,2009.

[3] GloMoSim. http://pcl.cs.ucla.edu/projects/glomosim/obtaining_glomosim.html

[4] OMNeT++, http://www.omnetpp.org/

[5] TCL code, http://www.tcl.tk/

[6] NAM: Network Animator. http://www.isi.edu/nsnam/nam/

[7] Ubuntu. http://www.ubuntu.com/

[8] Netbeans IDE http://netbeans.org/

[9] Tracegraph. http://www.angelfire.com/al4/esorkor/

[10] AA C. E. Perkins and E. M. Royer. "The ad hoc on-demand distance vector protocol," in Ad Hoc Networking, C. E. Perkins, ed. Addison-Wesley, 2001, pp. 173-219.

[11] Java, http://www.java.com/

[12] AA Mekni, M, Moulin, B., "A survey on Sensor Webs Simulation Tools", Sensor Technologies and Application, 2008, SENSORCOMM '08. Second International Conference on , vol, no, pp. 574-579, 25-31 Aug 2008.