UNIVERSITY OF MAURITIUS RESEARCH JOURNAL – Volume 15 – 2009
University of Mauritius, Réduit, Mauritius

Research Week 2007

Resource-Hungry Ad Hoc Wireless Network simulators

S Kissoondyal*
Email*: sdk_r56@yahoo.com*

C Boodhun
Email*: boodhunchundun@sbm.intnet.mu*

N Narayanen
Email*: gnn_nicolas@yahoo.com*

P Appavoo
Faculty of Engineering,
University of Mauritius
Email*: p.appavoo@uom.ac.mu*

Paper Accepted on 05 March 2009

Abstract

With the increasing spread of wireless telecommunications, mobile ad hoc networks (manets) are threading their way towards developing countries. Before deploying wireless applications/protocols in such countries or in places of limited resources, they are tested in a simulated environment. Presently, network designers/researchers/technologists use network simulators, which may not be optimal in the context used. Common simulators are resource-hungry and are either inappropriate for large scale testing in the resource-limited setting that prevails in developing countries, or simply too time consuming to have concluding results. This paper shows a classification of ad hoc wireless network simulators mainly based on the several characteristics like required platform (hardware and OS), application domains, scalability issues, mode of processing, availability of graphical user interface and mobility models. Each simulator's general and exceptional supports available will be critically reviewed. This will simplify the timely identification of the optimal wireless network simulator needed by researchers and/or practitioners, with respect to what they have to accomplish with the limited available resources.

Keywords: simulators, ad hoc network, programming languages, mobility models

**For correspondences and reprints*

1. INTRODUCTION

Research and development in wireless networking are still in the course of expansion in both developed and to-be developed regions of the world. Emerging theories and prototyped applications' behaviours in this field should be proved and analysed respectively before large-scale development. The required resources, mainly computing equipment and physical space, are not always available to perform these real-time experiments. For this purpose, a number of network simulators have been developed. The supports provided, for example the variation of certain actual parameters, are dependent on the context upon which leaded to its development. Under certain conditions, custom-based simulators are developed mainly in the absence of the appropriate simulation tool. In other cases, a time-consuming task is carried out comparing and analysing the different capabilities of the wide range of simulators available. This paper critically reviews the main features and capabilities of the commonly used simulators which are NS2, GloMoSim, PDNS (extension of NS2), OPNET, OMNET, J-SIM, ATEMU, QUALNET and JiST/SWANS. Furthermore, the table of simulators included in Section 3 allows researchers to effortlessly pick out the most appropriate simulator.

2. SIMULATORS

In this section, we give an overview of the most commonly used simulators, namely NS2, GloMoSim, PDNS, OPNET, OMNET, J-SIM, ATEMU, QUALNET and JiST/SWANS, after highlighting the main features that need to be looked into before choosing the right simulator.

2.1 Features classifying simulators

- *Licensing/cost*
  Cost of using the simulator is one of the main issues that affect its usage despite being suitable for a given situation. While some of them are available freely and even open source, especially for educational purposes, others are deployed on a commercial basis.

- *Operating system (OS)*
  Most of the simulators were developed on the Linux/Unix OS. The main programming languages used are C, C++ and Java. If Java is used, then the simulator is platform-independent, otherwise if C/C++ are used, Visual C++ may be used to run the simulator in Windows. In some cases, OS-specific applications are developed to support the simulator.

- *Scalability and mode of processing*
  The number of nodes simulated can scale up from a few hundreds with some simulators to millions with others when coupled with distributed processing.

- *Graphical output*
  Although some of the simulators do not have a Graphical User Interface to create network simulation's scenarios, they do have graphical output which proves to be very effective in visualising nodes movement and path established between communication parties.

- *Network type and routing protocols*
  The network type supported is mainly packet-switched with static network topology. The range of available routing protocols for ad hoc wireless network is many and is still growing but each simulator may support one only or a few.

- *Mobility models supported*
  Depending on the simulating events, nodes' movement follows certain paths that can be defined under mobility models. Some of the most commonly-used models: (1) *Random Waypoint Model*, where a mobile node starts from its original location, randomly chooses a location in the simulated area as the destination, and moves in a straight line to that destination location at the

speed that is uniformly distributed between 0 and a configurable maximum speed. When the mobile node reaches the destination location, it chooses a next destination location and repeats the procedure again; (2) the *RPGM Model* where each group has a logical center (group leader) that determines the group's motion behavior; (3) the *Freeway Mobility (FW) Model,* that emulates the motion behavior of mobile nodes on a freeway, it is expected to have spatial and high temporal dependence and imposes strict geographic restrictions on the node movement by not allowing a node to change its lane; (4) the *Manhattan Mobility (MH) Model,* which has been introduced to emulate the movement pattern of mobile nodes on streets defined by maps and is expected to have the same spatial and temporal dependence as well as geographic restrictions characteristics as *FW Model*. However, it differs from the *FW Model* in giving a node some freedom to change its direction; (5) the *Trajectory-Based Mobility Model*, that uses a trajectory array provided by the user is used to needs to specify how target and sensor nodes move. A mobile node moves from one point in the trajectory to the next at a constant speed.

2.2 Network Simulator Version 2 (NS2)

NS2 is a variant of the *Real Network Simulator* which is 'a network simulator originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks' [Ibrahim Haddad & David Gordon, 2002]. NS2 is free software; anyone 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.

Being written in C++, while having an OTcl interpreter as a front-end, NS2 is intended to simulate and research computer networks, using a discrete event simulation technique. But to add components and/or modifying existing one, several software modules need to be written and several parameters and multi-step data flows must be taken into account which makes NS2 a rather complex software.

Moreover, NS is currently being developed by the Defense Advanced Research Projects Agency), through the SAMAN project. Thus protocols and network operations are analyzed under extreme situations as the latter makes those network protocols and operations more susceptible to failure. In addition, NS run best under various UNIX systems such as FreeBSD, Linux, SunOS, and Solaris, since it was developed on them but it should run on a Posix-like computer where some tweaking may be needed. Moreover, NS is not compatible under Windows 9x/2000/XP unless Cygwin is used to provide a Linux-like environment.

Originally, this simulator has no built-in GUI; in fact it uses an external visualization Tool where the most commonly used one is called *Nam.* The lack of graphical tools could greatly help code development; unfortunately, the provided documentation does not help from this point of view. In fact, it is often limited and out of date with the current release of the simulator. Most problems must be solved by consulting the highly dynamic newsgroups and browsing the source code. Hence 'Nam is a Tcl/Tk based animation tool that is used to visualize the NS simulations and real world packet trace data' [Nam Trace, 2006]. Its ability to read large animation data sets come at the expense of a minimum amount of data is to be kept in memory and can also be extensible enough so that it could be used indifferent network visualization situations. Under this constraint NAM was designed to read simple animation event commands from a large trace file.

NS2 supports several Mobility Models like the Random Waypoint model which is provided by the "setdest" tool in the standard NS2 distribution but it is not provided in the release; the RPGM Model, the Freeway Mobility (FW), the Manhattan Mobility (MH) Model, FW Model. In addition to the mobility models, NS2 also contain several routing protocols like AODV, DSR, DSDV and OLSR.

Furthermore, the scale of operation may involve up to a few thousand nodes at the cost of a large memory footprint. Hence only small sized, but at the same time fine-grained simulations can be undertaken. Simple scenarios should run on any reasonable machine, but very large scenarios benefit from large amounts of memory. However, the above mentioned limitation is almost present in all network simulators present today. There is also an ongoing project to develop NS version 3.

2.3 GloMoSim

Global Mobile Information System Simulator (GloMoSim) is a 'scalable simulation environment for large wireless and wired communication networks' [UCLA Parallel Computing Laboratory, 2001]. It uses a parallel discrete-event simulation capability provided by Parsec and benefits from its ability to run on shared-memory symmetric processor (SMP) computers. Hence, the latest Parsec compiler is needed to run the simulator, fortunately, it now comes with the GloMoSim distribution. Moreover, some knowledge in Parsec is required to develop protocols and most protocol developers will be writing purely C code with some Parsec functions for time management.

Actually, the hardware requirements do not really impose a limitation. On the other side, the environment within which GloMoSim may operate may require additional software to run. For instance, in UNIX (Solaris with SPARC architecture) or Linux (Red Hat, Mandrake, many others) environment GloMoSim run smoothly while in Windows environment it is essential to have the Microsoft VC++ version 6.0 /SP4 and, to support the Visualization Tool, JAVA JRE version 1.2 or JAVA SDK version 1.2 will be needed. By the way, GloMoSim has a Visualization Tool that is platform independent as it is coded in Java.

Concerning its availability, GloMoSim is not public domain, but it is freely available without fee for education, or research, or to non-profit agencies. However, the documentation is limited as well as the set of tools accessible to hurry up the topologies creation, to supervise system performance and to generally plan the characteristics of and interact with the entire simulation environment. Being likely the second most used simulator, after NS2, in the research community, it is therefore widely accepted from a scientific point of view. Also, its highly modular design is such that it appears quite straightforward to modify and/or extend the basic functionalities and it can handle both parallel processing and sequential processing.

An important strength of GloMoSim is that it simulates networks with up to thousand nodes linked by a *heterogeneous communications capability* that includes multicast, asymmetric communications using direct satellite broadcasts, multi-hop wireless communications using ad hoc networking, and traditional Internet protocols. Also, the *node aggregation technique* is presented into this simulator to give considerable benefits to the simulation performance; in this way, the number of nodes in the system can be increased while maintaining the same number of entities in the simulation.

Besides, GloMoSim offers implementations of many different wireless layers (802.11, MACA, CSMA) and ad hoc routing protocols (DSR, Fisheye, IP with AODV) and supports various mobility models as Random Waypoint Mobility Model (RWPM), Reference Point Group, Pathloss Matrix and Random-Drunken. The Transport Layer provides two protocols either Transmission Control Protocol TCP or User Datagram Protocol (UDP). The choice of protocol is hard-coded into the simulator i.e. traffic always uses the UDP while all other traffic (HTTP, FTP) uses TCP.

QualNet, a commercial product from Scalable Network Technologies, derived from GloMoSim alleviates most of the GloMoSim flaws. However, it has loads of additional features with respect to GloMoSim.

2.4 PDNS

PDNS or Parallel/Distributed Network Simulator, developed by PADS Research group of Georgia Tech University is an upgraded version of NS. It is an extension to the NS2 simulator for parallel and distributed simulation of wired networks and similarly, PDNS is open-source. 'The Parallel/Distributed Network Simulator aims at overcoming the limitation of NS2 regarding its scalability. PDNS boosts NS2 processes by distributed the simulation over a network of closely coupled workstations [Hogie, Bouvry & Guinand, 2006].

Thus, this simulator uses a federated simulation approach where separate instantiations of NS2 modeling different sub-networks execute on different processors allowing the simulation of networks consisting of up to hundreds of thousands nodes. Such an approach ease the breakdown of a network into nodes, where each node can be seen as an active component of a network, and all nodes are working together with each other over real hardware platform. The load is shared between a set of interconnected CPU but does not require deep modification of the NS code to support roll-backs. 'The current version of PDNS has been tested on as many as 136 processors simulating a 600,000+ node network topology' [College of Computing, Georgia Institute of Technology, 2004].

To build and run a network simulator on PDNS, *libSynk* can be used. It is a compact and portable library for adding fast communication and synchronization to distributed applications. PDNS can operate in the following environment: Intel Linux, Intel Solaris, Sparc Solaris, SGI Irix, HP UX, Macintosh, OS X Microsoft Windows systems. PDNS uses TCL Scripts.

Moreover, shortcomings of using PDNS is that a user willing to use the distributed capability would have to figure out all the various possible independent virtual nodes in his NS script file. Furthermore, he would have to implement them as independent nodes using the new syntax. Though the new PDNS syntax is fairly simple and easy to use, defining a node point for a particular node of the cluster would be a tricky task assuming that all the nodes in the cluster are never equally loaded even with an active load balancer.

2.5 OPNET

OPNET (Optimized Network Engineering Tools) is an engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis created by MIL3, Inc. It includes graphical editors and animation and is used by large companies as it is a well-established and highly professional product. OPNET can operate in both Linux and windows environment and does not have any pre-install requirements. Furthermore, it is available for *free* for North-American universities (although this free license does not include any technical support). It can simulate all kinds of wired and several wireless networks. An IEEE 802.11 compliant MAC layer implementation is also provided.

Although OPNET is mostly intended for businesses to analyze or change their network, it is possible to implement any customized algorithm by reusing a lot of existing components. Most part of the deployment is made through a *hierarchical graphic user interface*. Network with several *hundreds of nodes* can be managed. However, the simulation requires a lot of processing power and can be very time consuming particularly for network with a large number of transmitter and receivers. This is mainly due to the detailed radio pipeline stage that OPNET uses since every packet transmitted is required to go through these stages. The simulation time can be reduced by using parallel processors.

OPNET is written in C++ hence, all the components are modeled in an object-oriented approach which gives intuitive easy mapping to your real systems. Besides, the mobility models supported are mainly arbitrary trajectories and Random Waypoint. Moreover, It supports various device models as traffic generators (workstations, servers, and stations), network devices (hubs, bridges, switches, routers/gateways), links (SONET, PPP, FDDI, 10BaseT, ISDN, xDSL) and vendor device models (Cisco Systems, 3Com, Nortel, Lucent, HP, etc.)

The graphical interface of the simulator simplifies most of the routine operations, while the development of new models always requires the definition of a finite state machine. If this fact can be seen as a difficulty at the beginning, it is also true that finite state machines are undoubtedly an expressive and effective tool for modeling. In OPNET, differently from NS2 and GloMoSim, it results quite easy to describe an application that bypasses part of the protocol stack. This aspect can be quite important to speed up and/or to reduce the level of unnecessary detail during the simulations. The performance seem to scale quite well, but there are not enough data in the current literature to make a more precise statement in this sense in the context of mobile ad hoc networks.

2.6 OMNET++

Objective Modular Network Testbed is a discrete event simulation tool designed to simulate computer networks, multi-processors and other distributed systems. Its application can be used for modeling other systems as well. It has proved to be a popular network simulation tool in the scientific community as well as in industry over the years.

In the commercial category, OPNET is widely held to be the state of the art in network simulation. OMNET++ is targeted at roughly the same segment of network simulation as OPNET. Non-commercial simulation tools cannot compete with some commercial ones (especially OPNET), which have a large selection of ready-made protocol models. OMNET++ is no exception. Network simulation tools naturally share the property that a model (network) consists of blocks, entities, modules, etc. connected by channels, connections. OMNET++ probably uses the most flexible method: it has a human-readable textual topology description format (the NED language) which is easy to create with any text-processing tool (perl, awk, etc.), and the same format is used by the graphical editor. It is also possible to create a driver entity to build a network at run-time by program. OMNET++ also supports submodule nesting.

OMNET++ links the GUI library with the debugging/tracing capability into the simulation executable. It is an architecture that enables the GUI to be very powerful where every user-created object is visible and modifiable in the GUI via inspector windows and the user has control over the execution. Hence, OMNET++ provides a unique debugging/tracing capability among the C++ based simulation tools. Performance is a particularly interesting issue with OMNET++ since the GUI debugging/tracing support involves some extra overhead in the simulation library.

OMNET++ is a non-commercial tool having a simulation library which is open-source. However, the most serious drawbacks is the lack of available protocol models, but since this is mostly due to the fact that it is relatively new, with the help of the OMNET++ user community, the situation is likely to become much better in the future.

2.7 J-Sim

J-Sim is implemented on top of a component-based software architecture called the autonomous component architecture (ACA). Its main application area is ***queueing network*** simulation, however, the range of its use can be very wide – almost any system where object states change discretely can be modeled using J-Sim. J-Sim is a Simula-like simulation environment written in Java.

J-Sim has been developed entirely in Java. This, coupled with the autonomous component architecture, makes J-Sim a truly platform-neutral, extensible, and reusable environment. J-Sim also provides a script interface to allow integration with different script languages such as Perl, Tcl, or Python. So, J-Sim is a dual-language simulation environment in which classes are written in Java (for NS2, in C++) and "glued" together using Tcl/Java.

J-Sim is built up upon some very well known principles inherited from the Simula language, such as: life of processes, simulation time, process state manipulation routines and queue facilities. J-Sim supports two different mobility models: trajectory-based and random waypoint.

2.8 ATEMU

Being a software emulator for AVR processor based systems, 'ATEMU is intended to bridge the gap between actual sensor network deployments and sensor network simulations' [Manish Karir, 2004]. The ATEMU distribution consists of two components: the atemu emulator core, and the xatdb graphical debugger. Moreover, the ATEMU can also be used in an educational but controlled environment to perform high fidelity large-scale sensor network emulation studies, without the purchase of expensive sensor node hardware. Furthermore, ATEMU has the sole ability to simulate a *heterogeneous* sensor network**,** besides, it is able to perform extremely low-level emulation of the sensor node hardware which make it different most simulators.

In addition, to the support for the AVR processor, this simulator supports other peripheral devices on the MICA2 sensor node platform such as the radio and as it is binary compatible with the aforementioned MICA2 hardware, developers of TinyOS related software could use it directly. However, though the current release only includes support for MICA2 hardware, it also provides extension for the introduction of other sensor node platforms. It allows for the use of heterogeneous sensor nodes in the same sensor network.

ATEMU provides support for multiple platforms including Solaris and several distributions of Linux and also offers a solution around the logistical problems of carrying out testing and research with large numbers of physical devices.
While allowing a precise and faithful simulation of the micro controller behaviour and radio-waveform propagation over air, the ATEMU simulator simply cannot scale to a large network because of the naïve implementation of global time, where each node is advanced a single clock cycle at a time in one large loop hence parallel simulation is not possible. Besides, ATEMU also does not present a useful program representation for analysis tools, but treats most of the program as raw binary.

Scenarios that require extremely high-fidelity results even at the expense of longer simulation times can benefit immensely from ATEMU. However, ATEMU is not as portable and has trouble scaling to large networks.

2.9 QualNet

'QualNet is a commercial product which is derived from GloMoSim' [Gianni A. Di Caro, 2003]. Its large-scale simulation capability is being recognized since it is reported as "scalable up to 10's of thousands of nodes" and besides, the simulator provides supports for parallel execution. Moreover, Qualnet has already made provision for an extensive protocol model library for both wired and wireless networks (local, ad hoc, satellite and cellular) and algorithms. It extends the GloMoSim offer by bringing support, a more completed documentation, a complete set of user-friendly tools for building scenarios and analyzing simulation output. But, still, QualNet is not very popular due to its commercial nature.

Along with a satisfactory level of usability, modifiability and expandability QualNet comes with a rather good, highly modular software design with possibility of parallel and/or distributed implementations. Moreover, it has been accepted from the scientific community and possesses advanced graphical and mathematical tools for experiment building, monitoring and post-processing. Adequate documentation and technical support is offered and continuously updated while realistic 3D model of the environment has been specified and implemented.

As such, QualNet seems to be the most complete network simulator, in terms of available protocols, models and tools for what concerns mobile ad hoc networks. QualNet also provides support for Java and Visual C++ codes.

2.10 JiST/SWANS

SWANS, Scalable Wireless Ad hoc Network Simulator, was developed on top of JiST (Just in Simulation Time) a discrete event simulation engine, that runs over the Java Virtual achine. SWANS, with capabilities of NS and GloMoSim, can simulate networks of orders of magnitude larger than it would have been otherwise possible with any other simulators with the same amount of memory and processing power. This is mainly because (1) message garbage collection, (2) reduced lines of codes, components of JiST are half of similar components of GloMoSim which in turn are much smaller than that of NS, (3) objects are passed by reference and therefore eliminating serialisation or copy or context-switching, and (4) broadcasted packet can be shared among receiving nodes with its feature of timeless object. Moreover, SWANS has the unique feature of running regular java applications over the simulated network.

3. TABLE OF SIMULATORS

The following table provides a quick reference to compare and pick out the most appropriate simulator to fulfil a required need.

Simulator	***Licence***	***Operating System***	***Scalability***	***Mode of Processing***	***Graphical Output***	***Network Type***	***Mobility Model***
NS2	Free software, Can be redistributed/ modified under the terms of the General Public License	Run efficiently under UNIX systems but under Windows, Cygwin is needed to provide a Linux-like environment.	Few thousand nodes	Distributed	Have no built-in GUI. Use external visualization tool, the most common one is NAM	Static n/w topology with Packet-switched data networks	Random Waypoint; Freeway Mobility; Manhattan Mobility; *RPGM.*
GloMoSim	Not public domain, but is freely available for education, or research, or to non-profit agencies	UNIX or Linux. But Windows needs Microsoft VC++ version 6.0 /SP4	Few thousand	Symmetric Multi-processor	In-built Visualization tool where JAVA JRE version 1.2 is needed.	Static (Parsec library) n/w topology with Packet-switched data networks	Random Waypoint; Reference Point Group; Pathloss Matrix
PDNS (extension of NS2)	open-source	Intel Linux, Intel Solaris, Sparc Solaris, SGI Irix, HP UX, Macintosh OS X, Windows.	Hundred of thousands	Parallel and Distributed processing		Static n/w topology with Packet-switched data networks	
OPNET	Commercially deployed user-network systems but freely available without technical support	Operate smoothly in both Linux and windows environment.	Few hundred	Parallel and distributed	Included in distribution	Static + node mobility n/w topology with packet-switched data networks	Arbitrary trajectories, Random Waypoint,
OMNET	Free for Academic and Educational use	Operate smoothly in both Linux and windows environment.	Very large scale. Limit to virtual memory capacity of computer	Muti processor/ distributed	Tk-based graphical, windowing user interface and command-line user interface	Including top-level network, node, LAN	Random Waypoint
J-SIM	Open Source	Solaris 8, OS/2 Warp 4, Windows NT 4, Windows XP, and Linux		Parallel	*GEditor,* a java package	Wireless / wired ad hoc networks	Sensor Mobility Model
ATEMU				Distributed	Included in distribution	Sensor Networks	Arbitrary trajectories, Random Waypoint,
QUALNET	Commercially deployed	Operate smoothly in both Linux and windows environment. Windows needs VC++ environment	Tens of thousands	Parallel and sequential	In-built Visualization tool where JAVA JRE version 1.2 is needed. 3D output	Static (Parsec library) n/w topology with Packet-switched data networks	Random Waypoint; Reference Point Group; Pathloss Matrix
JiST/SWANS	Open source, Free non-commercial academic use	Any OS, JVM required. Java	Million	Sequential	No graphical output, Customised java-based GUI possible	Wireless ad hoc networks	Random walk, Random Waypoint.

Table 1: Main features of highlighted Simulators

4. CONCLUSION

The most popular, most commonly used, network simulators are (1) NS2 using C++, (2) GloMoSim using C, (3) QualNet, commercial, using Java and Visual C++, and (4) SWANS using Java. NS and GlomoSim has a bounded capacity of a few thousand nodes [Riley and Ammar, 2002]. PDNS (Parallel/Distributed NS) is an extension of NS due to excessive memory and CPU requirements of the latter, the former allows the distribution of processing across a network of workstations. QualNet, a commercial product, is the parallel version of GloMoSim, sequential version. SWANS, built atop of JiST with capabilities of NS and GloMoSim, can simulate networks of orders of magnitude larger than it would have been otherwise possible with any other simulators with the same amount of memory and processing power.

5. REFERENCES

COLLEGE OF COMPUTING, GEORGIA INSTITUTE OF TECHNOLOGY, 2004, Retrieved November 20, 2006, from http://www.cc.gatech.edu/computing/compass/pdns/

FAN BAI, NARAYANAN SADAGOPAN & AHMED HELMY, 2004, User Manual for IMPORTANT Mobility Tool Generators in NS2 Simulator, Retrieved November 20, 2006, from http://nile.usc.edu/important/

GIANNI A. DI CARO, 2003, Analysis of simulation environments for mobile ad hoc networks, Technical report no. IDSIA-24-03, pp. 18-21, Retrieved November 24, 2006, from IDSIA / USI-SUPSI Database, http://www.idsia.ch/idsiareport/IDSIA-24-03.pdf

G. RILEY AND M. AMMAR, 2002, Simulating large networks: How big is big enough?, *in proceedings First International Conference on Grand Challenges for Modeling and Simulation*.

IBRAHIM HADDAD & DAVID GORDON, 2002, Network Simulator 2: A Simulation Tool For Linux, retrieved november 20, 2006, from http://www.linuxjournal.com/article/5929

LUC HOGIE, PASCAL BOUVRY & FRÉDÉRIC GUINAND, 2006, An Overview of MANETs Simulation, Retrieved November 26, 2006, from http://agamemnon.uni.lu/~lhogie/me/publications/pdf/

MANISH KARIR, 2004, ATEMU: A Fine-grained Sensor Network Simulator, Retrieve on 22 November 2006 from, www.merit.edu/networkresearch/papers/pdf/2004/karir-secon2004.pdf

NAM TRACE, 2006, Retrieved November 20, 2006, from http://www.isi.edu/nsnam/ns/doc/node590.html.

UCLA Parallel Computing Laboratory, 2001, Retrieved November 20, 2006, from http://pcl.cs.ucla.edu/projects/glomosim/

Simulator	***Licence***	***Operating System***	***Scalability***	***Mode of Processing***	***Graphical Output***	***Network Type***	***Mobility Model***
NS2	Free software, Can be redistributed/ modified under the terms of the General Public License	Run efficiently under UNIX systems but under Windows, Cygwin is needed to provide a Linux-like environment.	Few thousand nodes	Distributed	Have no built-in GUI. Use external visualization tool, the most common one is NAM	Static n/w topology with Packet-switched data networks	Random Waypoint; Freeway Mobility; Manhattan Mobility; *RPGM*.
GloMoSim	Not public domain, but is freely available for education, or research, or to non-profit agencies	UNIX or Linux. But Windows needs Microsoft VC++ version 6.0 /SP4	Few thousand	Symmetric Multi-processor	In-built Visualization tool where JAVA JRE version 1.2 is needed.	Static (Parsec library) n/w topology with Packet-switched data networks	Random Waypoint; Reference Point Group, Pathloss Matrix
PDNS (extension of NS2)	open-source	Intel Linux, Intel Solaris, Sparc Solaris, SGI Irix, HP UX, Macintosh OS X, Windows.	Hundred of thousands	Parallel and Distributed processing		Static n/w topology with Packet-switched data networks	

OPNET	Commercially deployed user-network systems but freely available without technical support	Operate smoothly in both Linux and windows environment.	Few hundred	Parallel and distributed	Included in distribution	Static + node mobility n/w topology with packet-switched data networks	Arbitrary trajectories, Random Waypoint,
OMNET	Free for Academic and Educational use	Operate smoothly in both Linux and windows environment.	N/A	Muti processor/ distributed	Tkenv: an interactive execution environment, allowing progress of simulation & change parameters.	Wireless / wired ad hoc networks	Random Waypoint
J-SIM	Open Source	Solaris 8, OS/2 Warp 4, Windows NT 4, Windows XP, and Linux	N/A	Parallel	*GEditor*, a java package	Wireless / wired ad hoc networks	SensorMobi lityModel
ATEMU	Open Source	AVR processor based systems	N/A	Distributed	Included in distribution	Sensor Networks	Arbitrary trajectories, Random Waypoint,

QUALNET	Commercially deployed	Operate smoothly in both Linux and windows environment. Windows needs VC++ environment	Tens of thousands	Parallel and sequential	In-built Visualization tool where JAVA JRE version 1.2 is needed. 3D output	Static (Parsec library) n/w topology with Packet-switched data networks	Random Waypoint; Reference Point Group, Pathloss Matrix
JiST/SWANS	Open source, Free non-commercial academic use	Any OS, JVM required. Java	Million	Sequential	No graphical output, Customised java-based GUI possible	Wireless ad hoc networks	Random walk, Random Waypoint.

Table 1: Main features of highlighted Simulators