Survey on Simulation Tools for Wireless Networks

K.Lakshmanarao
AsstProfessor
IT-Dept,GMRIT

Ch.R. VinodKumar
AsstProfessor
IT-Dept,GMRIT

K.Kanakavardhini
Asstprofessor
IT-Dept,GMRIT

Abstract

The establishment of a real time network scenario is very difficult in the wireless network research area and in the performance analysis of wireless network protocols. So the physical arrangement of wireless network is not easy task and very costly. Now days the usage of simulators is increased to test wireless networks. This paper presents advantages, disadvantages and features of different simulators. The usage of simulators reduces time and cost of testing, analysis of wireless networks. We hope this survey is useful for those who feel difficult to select appropriate simulator to carry their research work in wireless networks.

1. Introduction

The simulation is used to provide abstract view of network model and its functions. The simulator used to design the imaginary view of a real life network objects on a computer. The developed imaginary view is used to study the behaviour of a wireless network model, to calculate mathematical formulas, evaluate performance of the model through some criteria like packet delay, loss of transmission and so on. The network simulator provides an integrated , versatile, flexible GUI-based network developer tool to develop and simulate a network for an appropriate protocol stack , routing algorithms and to test Quality of service in particular criteria and so on[1]. In wireless networks mobility of a network device creates so many problems. The simulator design is very useful to test various mobility models [8] in wireless networks as stated below:

1. Random Walk Mobility Model (including its many derivatives): A simple mobility model based on random directions and speeds.
2. Random Waypoint Mobility Model: A model that includes pause times between changes in destination and speed.
3. Random Direction Mobility Model: A model that forces MNs to travel to the edge of the simulation area before changing direction and speed.
4. A Boundless Simulation Area Mobility Model: A model that converts a 2D rectangular simulation area into a torus-shaped simulation area.
5. Gauss-Markov Mobility Model: A model that uses one tuning parameter to vary the degree of randomness in the mobility pattern.
6. A Probabilistic Version of the Random Walk Mobility Model: A model that utilizes a set of probabilities to determine the next position of wireless network.
7. City Section Mobility Model: A simulation area that represents streets within a city.

[5]The objective of any simulator is to accurately model and predict the behaviour of a real world environment. Developers are provided with information on feasibility and reflectivity crucial to the implementation of the system prior to investing significant time and money. Simulation-based testing can help to indicate whether or not these time and monetary investments are wise.

2. Simulator selection criteria

The best simulator selection process is measured based on Simulator Architecture, Usability, Scalability Statistics, Underlying Network Simulation and System Limitations [7].

1. Simulator Architecture [7]: This criteria relating to the design and working style of the simulator, the supported features and their implementation. These criteria also include whether it supports structured or unstructured overlay simulation, or both; whether the simulator works on discrete event simulation basis, that is whether it uses a scheduler which synchronises message transfer between nodes, adding delay as necessary; how it implements remote procedure calls; how the identifiers

are chosen; whether it allows for distributed simulation, i.e. can simulations be run across a number of machines to allow greater scaling or faster simulation runtime? These criteria also include aspects of how node behaviour is simulated. Whether both iterative and recursive routing can be simulated.

2. Usability: [7]This criteria focus on the flexibility of the simulator to learn and use, i.e. whether the simulator API allows code to be easily understood, conversion of source code is compatible with other simulators. Documentation support and online support in bug fixing exist or not.
3. Scalability: [7] It concentrates on how the simulation model supports to thousands of nodes or more.
4. Statistics: [7] This criteria concentrates on flexibility in produced results, graphs and statistical analysis of a simulation model.

3. SIMULATION TOOLS ANALYSIS

This section succinctly introduces various wireless network simulators. The reviews mainly based on published or publicly available information about the simulation tools.

NS2: NS2 [9, 10] is an open-source, Object-oriented discrete event-driven simulator designed specifically for research in computer communication networks. Since its inception in 1989, NS2 has continuously gained tremendous interest from industry, academia, and government. [2,5] NS-2 Simulations are based on a combination of C++ and OTcl. In general, C++ is used for implementing protocols and extending the ns-2 library. OTcl is used to create and control the simulation environment itself, including the selection of output data. Simulation is run at the packet level, allowing for detailed results. The fig.1 describes usage of OTcl language in NS-2.

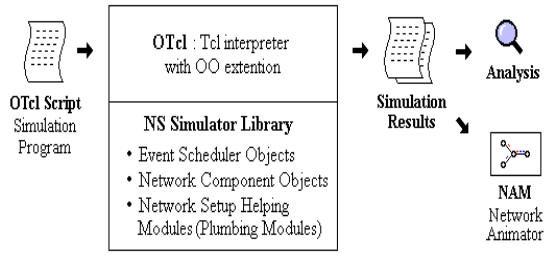

Fig.1. OTcl usage in NS-2

Simulations can be observed graphically by Network Animator (NAM) and xgraph is used for simulation results. After compiling the simulation source to executable and running it to generate trace files, simulation results can be observed graphically by using Network Animator (NAM). Ns-2 does not scale well for sensor networks [2, 5]. This is in part due to its object-oriented design. Another drawback to ns-2 is the lack of customization available. Packet formats, energy models, MAC protocols, and the sensing hardware models all differ from those found in most sensors. One last drawback for ns-2 is the lack of an application model. In many network environments this is not a problem, but sensor networks often contain interactions between the application level and the network protocol level.

NS3: The NS-3[11,12] simulator is a discrete-event network simulator targeted primarily for research and educational use. The ns-3 project, started in 2006, is an open-source project. The Open source licensing based on GNU GPLv2 compatibility. Ns-3[11,12] is not an extension of ns-2; it is a new simulator. The two simulators are both written in C++ but ns-3 does not support the ns-2 APIs [12]. It provides documentation Coverage of the C++ APIs using Doxygen as well as Documentation of the Bake integration tool.

It supports simulation of wireless networks MAC Layer protocols, routing protocols and Qos parameters. NS-3 supports key abstractions like node, application, channel, Net Devices, Topology Helper. The script can be written in either C++ or python. The modules supported by NS-3 are given in below fig.2.

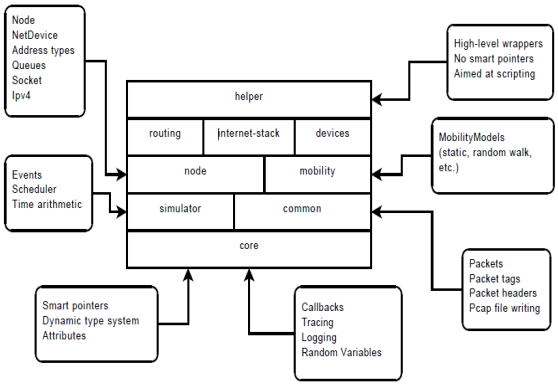

Fig.2. the different modules of NS-3

OMNeT++: OMNeT++[14] is an open source object-oriented modular discrete event network simulation framework. It has a generic architecture, so it has been) used in various problem domains such as modeling of wired and wireless communication

networks, protocol and queueing networks modeling, modeling of multiprocessors and other distributed hardware systems, it is used to perform validating of hardware architectures, evaluating performance aspects of complex software systems and in general, modeling and simulation of any system where the discrete event approach is suitable, and can be conveniently mapped into entities communicating by exchanging messages.

OMNeT++[14] simulations can be run under various user interfaces. Graphical, animating user interfaces are highly useful for demonstration and debugging purposes, and command-line user interfaces are best for batch execution. The simulator as well as user interfaces and tools are highly portable. They are tested on the most common operating systems like Linux, Mac OS/X, Windows, and they can be compiled out of the box or after trivial modifications on most Unix-like operating systems.

In contrast to ns-2 and ns-3, OMNeT++[2,6,14] is not a network simulator by definition, but a general purpose public source, component-based discrete event network simulator discrete event based simulation framework[15]. OMNeT++[14] also supports parallel distributed simulation. it can use several mechanisms for communication between partitions of a parallel distributed simulation, for example MPI or named pipes. The parallel simulation algorithm can easily be extended, or new ones can be plugged in. The main simulation frameworks[15] for OMNeT++ 4.x are INET framework , OverSim,veins, INETMANET, MIXIM and castalla.

The INET framework is a standard protocol model library of OMNeT++[15]. INET contains models for the Internet protocol stack (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.) for wired and wireless link layer protocols (Ethernet, PPP, IEEE 802.11, etc), support for mobility, MANET protocols, DiffServ, MPLS with LDP and RSVP-TE signalling, several application models, and many other protocols and components.

There are several INET-based model frameworks, maintained by independent research groups some of those models are listed below: OverSim[15] is an open-source overlay and peer-to-peer network simulation framework for the OMNeT++ simulation environment. The simulator contains several models for structured (e.g. Chord, Kademlia, Pastry) and unstructured (e.g. GIA) P2P systems and overlay protocols. Veins[15] is an open source Inter-Vehicular Communication (IVC) simulation framework composed of an event-based network simulator and a road traffic micro-simulation model. INETMANET[15] is a fork of the INET Framework, maintained by Alfonso Ariza Quintana. It is kept up-to-date with INET, and adds a number of experimental features and protocols, mainly for mobile ad-hoc networks, many them written by Alfonso Ariza. MiXiM[15] is an OMNeT++ modeling framework created for mobile and fixed wireless networks like wireless sensor networks, body area networks, ad-hoc networks, vehicular networks, etc.

Castalia[15] is a simulator for Wireless Sensor Networks (WSN), Body Area Networks (BAN) and generally networks of low-power embedded devices. It is used by researchers and developers to test their distributed algorithms and/or protocols in realistic wireless channel and radio models, with a realistic node behaviour especially relating to access of the radio. Castalia's salient features include: model for temporal variation of path loss, fine-grain interference and RSSI calculation, physical process modeling, node clock drift, and several popular MAC protocols implemented. Castalia is highly parametric. It provides tools to help run large parametric simulation studies , process and visualize the results.

OMNeT++ is capable of running most TinyOS simulations by NesCT application that converts TinyOS source to simulator compatible C++ code [13]. NesC simulation code interchange between sensor platforms is possible but only in a restricted sense, because the protocol and hardware implementation in the simulator is simplified, and not all hardware is supported.

TINYOS: [17] is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal area networks, smart buildings, and smart meters. A worldwide community from academia and industry use, develop, and support the operating system as well as its associated tools

J-SIM: JSim has been developed by a team at the Distributed Real-time Computing Laboratory (DRCL). The project has been sponsored by the National Science Foundation (NSF), DARPA's Information Technology Office, Air Force Office of Scientific Research's Multidisciplinary University Research Initiative, the Ohio State University and the University of Illinois at Urbana-Champaign. J-Sim is free and available with source code. JSim[18] is a Java-based simulation system for building quantitative numeric models and analyzing them with respect to experimental reference data. JSim's primary focus is in physiology and biomedicine, however its computational engine is quite general and applicable to a wide range of scientific domains. JSim models may intermix ODEs, PDEs, implicit equations, integrals, summations, discrete events and procedural code as appropriate. JSim's model compiler can automatically insert conversion factors for compatible physical units as well as detect and reject unit unbalanced equations.

JSim[19] model calculations are specified in JSim's own Mathematical Modeling Language(MML) an easy-to-read text-based language. MML models are most often expressed in terms of mathematical equations (for example, ordinary and partial differential equations, implicit equations), but formulation via discrete events and function calls to Java, C and Fortran are also available. MML is constructed so that model writers may intermix mathematics, events and procedural code as needed. The detailed architecture of Jsim available at [19] gives light on different supported modules.

Its script uses JAVA and TCL, the TCL Scripting is an essential part of J-Sim, use it to "glue" all the components and define how the system operates. It makes it possible to manipulate Java objects in the Tcl environment, such as creating an object from a Java class, invoking a method of a Java object, or accessing a field variable of a Java object. The architecture is shown below.

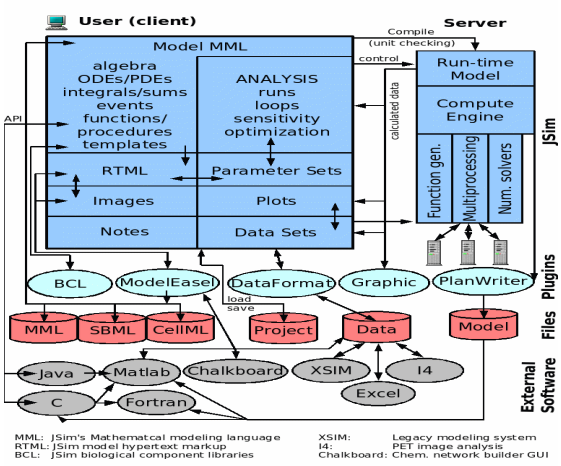

Fig.3. The architecture of JSim

OPNET: This simulator is developed by OPNET technologies. OPNET had been originally developed at the Massachusetts Institute of Technology (MIT) and since 1987 has become commercial software. It provides a
comprehensive development environment supporting the modelling of communication networks and distributed systems. Both behaviour and performance of modelled systems can be analysed by performing discrete event simulations.
The main programming language in OPNET is C (recent releases support C++ development). The initial configuration(topology setup, parameter setting) is usually achieved using Graphical User Interface (GUI), a set of XML files or through C library calls.

NetSim: It is used to create platform independent software that could be used in simple, consumer electronic products. Java designed for simple, efficient, platform-independent program for creating WWW-based programs. Using Java one can create small programs called applets that are embedded into an HTML document and viewable on any Java-compatible browser. Java applets are compiled into a set of byte-codes, or machine-independent processing instructions.

SimPy: It is a process-oriented discrete-event simulator. Unlike the other simulators, no public available network models exist for SimPy. Instead, it is a bare simulation API written in Python. In SimPy, the basic simulation entities are processes. They are executed in parallel and may exchange Python objects among each other. Most processes include an infinite loop in which the main actions of the process are performed. Besides abstractions for processes and the related exchange of objects, SimPy provides

instructions for the synchronization of simulation processes and commands for the monitoring of simulation data.

QualNet: It is a commercial network simulator from Scalable Network Technologies, in 2000-2001. It is ultra-high-fidelity network simulation software that predicts wireless, wired and mixed-platform network and networking device performance. For implementing new protocols, Qualnet uses C/C++ and follows a procedural paradigm. Uses the parallel simulation environment for complex systems.

4. Conclusions

The objective of this paper have been to provide survey on various simulators to design abstract model for wireless network and discussed existing pros and cons of each simulator. With this knowledge network model developer choose apt simulator to carry out the work. It allows users to select the most suitable simulator to test their test bid performance analysis and results.

5. References

[1] Mrs. Saba Siraj, Mr. Ajay Kumar Gupta, Mrs Rinku-Badgujar, "Network Simulation Tools Survey," International Journal of Advanced Research in Computer and Communication Engineering. ,Vol. 1, Issue 4, June 2012.

[2] Marko Korkalainen, Mikko Sallinen, Niilo Kärkkäinen, Pirkka Tukeva, "Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications", Fifth International Conference on Networking and Services 2009.

[3] Harsh Sundani, Haoyue Li, Vijay K. Devabhaktuni, Mansoor Alam, & Prabir Bhattacharya, "Wireless Sensor Network Simulators A Survey and Comparisons", International Journal Of Computer Networks (IJCN), Volume (2) : Issue (5).

[4] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, P. Pavón-Mariño, J. García-Haro, "Simulation Tools for Wireless Sensor Networks", Summer Simulation Multiconference - SPECTS 2005.

[5] David Curren University of Binghamton, "A Survey of Simulation in Sensor Networks".

[6] Elias Weing¨ artner, Hendrik vom Lehn and Klaus Wehrle, "A performance comparison of recent network simulators", IEEE ICC 2009.

[7] Stephen Naicken, Anirban Basu, Barnaby Livingston and Sethalat Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators", PGNet 2006.

[8] T . Camp, J. Boleng, and V . Davies, "Survey of Mobility Models", Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.

[9] NS-Guide.pdf Text book from Springer

[10] The Network Simulator-ns2. http://www.isi.edu/nsnam/ns.

[11] NS-3 Tutorial http://www.nsnam.org/ns-3-18/documentation/

[12] NS-3 Documentation http://www.nsnam.org/ns-3-18/documentation/

[13] NS3 http://www.nsnam.org

[14] OMNet ++ discrete event simulation system, Available from: http://www.omnetpp.org.

[15] OMNet++ available at:http://www.omnetpp.org/doc/omnetpp/manual/usman.html

[16] OMNet++ frameworks available at: http://www.omnetpp.org/models

[17] TinyOS available at: http://www.tinyos.net/

[18] JSim available at http://bioeng.washington.edu/jsim/

[19] JSim documentation available at http://bioeng.washington.edu/jsim/docs/overview.html.