A Guideline for Selecting an Open-source WSN simulator

Siniša Nikolić, Valentin Penca
{sinisa_nikolic, valentin_penca}@uns.ac.rs
Fakultet tehničkih nauka u Novom Sadu

Abstract – *Simulation is now used for various aspects of WSN networks research and development. Use of simulators overcomes the problems which user must oppose in real world evaluation. In this paper we define a, set of criteria for comparing WSN network simulators and, based on proposed criteria, analyze three open source, event based simulators NS-2/MannaSim, OMNeT++/Castalia and SWANS/SIDnet that have larger impact in WSN domain. Finally, we suggest simple, practical guideline for simulator selection.*

1. INTRODUCTION

Wireless Sensor Network (WSN) is a network of spatially distributed autonomous sensors that commonly includes hundreds of thousands low cost, battery-powered and reduced size devices with limited processing, sensing and wireless communication capabilities.

The specifics and complexity of WSN often lead to analytical methods to be unsuitable or inaccurate for simulation. Additionally, the proportion of algorithms that are analyzed through practical evaluation is comparatively low, possibly due to the relative infancy, deployment cost, time required, broad diversity and application dependence of WSN.

WSN simulators are usually developed as an extension of some existing network simulators. In a contrast to network simulators, WSN simulators must provide more complex implementation of radio channel, physical environment, sensing and energy models.

In this paper we are presenting the criteria for evaluation and a survey of open-source simulators for WSN simulation purposes.

The rest of this paper is organized as follows. Section 2 introduces the problem and highlights the main problems of WSN evaluation under real-world scenario assumption. Section 3 briefly presents related work in the filed of WSN simulators and proposes the evaluation criteria for comparing WSN simulators. Section 4 contains evaluation results for selected open source WSN simulators. Section 5 contains an argumented guideline for selecting open-source simulators aimed at WSN simulation.

2. PROBLEMS OF REAL-WORLD EVALUATION

Evaluation is an unavoidable step for studying, development and improvement of WSN. Due to the constraints and problems existing when a real-world evaluation scenario is deployed, the use of simulators is essential and sometimes the only possible solution to test new applications, algorithms and protocols for WSN. Problems that have the greatest impact are [2] :

- Cost of hardware devices for the real-world scenarios is often too high.
- Deployment and relocation of sensors in the real world environment is time consuming, difficult and potentially dangerous task.
- Debugging of WSN in a distributed environment which has a large number of sensors. Potential result of debugging is redeployment of WSN.
- Creation and exploitation of a controlled (specific) environment likewise influence of obstacles on the propagation of signals is almost impossible.
- Repeating the identical scenario with different parameters in the system is almost impossible in the real-world.

Simulators enable experiments in controlled conditions in which it is possible to check the individual aspects of sensor networks. They overcome all problems of evaluation in the real-world scenario, but they do not provide a guarantee that the simulated system will behave identically in the real-world. The usage of the simulators is only one possible phase in evaluation and analysis of network. In [1] the results of simulation and real-world testbed are presented.

Simulation results are relying on the particular scenario that is being studied (environment), hardware and physical layer assumptions which may not lead to a complete demonstration of all that is happening, so the credibility of results may be taken into a question. Therefore, after successful simulation, as the final phase of experiments it is recommended to test the WSN in real conditions.

3. EVALUATION METODOLOGY

Nowadays, there are many network simulation frameworks in which sensor networks can be tested, where various simulators have different capabilities and goals. There are generic purpose simulators, usually extension of network simulators used to simulate the network traffic, algorithms and to research different aspects of communication protocols. Other simulators are more specialized and optimized to simulate execution of a specific simulation platform. The problem of choosing the simulator can be a difficult task, particularly for the beginners in the WSN simulation.

The papers [2], [3] and [4] present surveys of WSN simulation tools. They offer slightly diverse comaprison criteria and briefly describe realtively large number of tools. Unfortunatelly, the suggested criteria are highly aggregated and do not allow deeper insight in the simulation process. In our approach, a smaller number of simulators will be described in more details,

and they will be compared by set of criteria which is of relatively fine granulation.

All features that will be used as criteria for evaluation and comparison are divided in two groups: *functional* and *non-functional*. This classification implies strict separation of features into those which describe simulator as software and those which are functional features essential for representing the WSN. Similar categorization is proposed by [5]. The list and description of the criteria used in this paper is given below.

3.1. Functional features

- *Possibility of usage* – in some references this simulator criteria is also referred to as *complexity* [1] or *level of details* [4]. It represents the possibility to simulate WSN on different levels (communication, application, hardware) of use. There are two types of simulators. First type, the *generic simulators* focus on communication, sensing and data processing aspects. These types of simulators are useful for observing the WSN in general, where hardware/software infrastructure of node are not being focused on. These simulators can be used to develop, compare and learn algorithms and protocols in WSN. The second type, *specific simulators* are specialized for emulating hardware and software platforms. *Specific simulators* are often called emulators and can be further classified into: *application* (code) and *instruction* (firmware). The purpose of a*pplication* simulators is to simulate a specific software environment (operating system), where the same application can be run on simulator and on real node without changes. *Instruction* simulators support emulation of the sensor nodes hardware..
- *Time dimension* – how simulation is executed over a time. This criterion classifies the simulators into *discrete-event* and *time-driven*. The idea of a *discrete-event* simulator is to jump from one event to the next. The events are recorded as event notices in the future event list (FEL). Each event occurs at a point in time and marks a change of state in the system. A *discrete-event* simulator uses event scheduler to simulate events. In a *time-driven* simulation we have a various recording of time, which is follows fixed steps.
- *Energy consumption models* – the models of power consumption in relation to the operation of sensor nodes.
- *Available protocol models* – Which OSI protocols are available in a simulator.
- *Scalability* – how simulator reacts when a large scale sensor network is simulated. The scalability depends on how the simulator performs regarding memory usage and programming language in which the simulator is implemented (low level languages tend to perform better then high level languages).
- *Deployment models* – introduce the initial placement of sensor nodes in virtual field. Simulators implement the random (from plane, topology patterns) and manual placement of nodes.
- *Mobility models* – sensor nodes can change position in simulation. Their position can be randomly or manually changed (user can define moving paths).
- *Interaction with real nodes* – possibility of the simulator to include real, physical nodes into simulation.
- *Sensing models* – how is sensing process modeled on nodes (random generation of sensing information or advanced sensing models which include sensing environment).
- *Radio channel* – available models of signal propagation (signal fading, terrain modeling).
- *WSN platforms models* –the hardware and software sensor platforms which can be simulated.

3.2. Non-Functional features

- *License* –availablility to the end users (terms of usage and redistribution).
- *Popularity* –how intensively the simulator is being used by the community. The level of acceptance of simulator is defined by the number of science papers, users and books. The most popular simulators are those that are open source, in phase of active development, portable, reusable, with GUI support and rich with protocol and node models.
- *Platform independence* – possibility to use the simulator in different operating systems.
- *GUI tools* – support of graphical user interfaces. The desirable functionalities of GUI tools are: trace support (monitoring of simulation), debug support (interacting in simulation – inspection of modules and variables, possibility to add queries in real time) and modeling (visual modeling and composition of simulation scenario).
- *Programming language* – language in which simulator is written. Users' level of knowledge of some programming language may be crucial in the selection of a simulator.
- *Active development* – criterion that answers the question "is the simulator still being developed?"
- *Documentation* – availabaility and quality of user and developer manuals.
- *Parallel execution* – the goal is to get higher speed of simulation execution. Only a few simulators implement this feature.
- *Extensibility* – reusability and change of available models.

4. SIMULATORS OVERVIEW

In this section we are presenting three open source simulators that have more significant impact in WSN domain. The selection is based on

4.1. NS-2/MannaSim

NS-2 [6], [7], [8] began as NS (Network Simulator) in 1989 with the purpose of general network simulation for studying the dynamic nature of communication networks. It is the most popular simulation tool for sensor networks. NS-2 is an object-oriented discrete event simulator which allows for straightforward creation and use of new protocols. NS-2 is the paradigm of reusability; its modular approach for protocol development has effectively made NS-2 extensible.

Simulations are based on a combination of C++ and OTcl. In general, C++ is used for implementation of 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. NAM is a simple GUI tool which can be used for animating the simulation results in NS-2. NS2 is available for Linux (Ubuntu, Fedora etc.), FreeBSD, SunOS/Solaris, HP/SGI, and Windows 95/98/NT/ME/2000/XP.

NS-2 uses a very simple energy model, where energy level is implemented as a node attribute. Every node has initial amount of energy (at the beginning of the simulation) that is decreased for every packet it transmits and receives (identical for every size of packet). Mobility and deployment of nodes may be set explicitly in the configuration or can be randomly set by the simulator. Direction and seed can be set for every node movement.

NS-2 does not scale well for sensor networks, it is not easy to scale the NS2 beyond several hundred simulated nodes (limit may be thousand nodes). Every node is its own object and can interact with every other node in the simulation. The previous said creates a large number of dependencies to be checked at every simulation interval (n^2 connection may be possible). Object-oriented architecture is the problem [9].

The Mannasim Framework [10] is an extension for WSN simulation based on NS-2. Mannasim extends NS-2 introducing new modules for design, development and analysis of different WSN applications. Main feature that manasim introduces is WSN sensing model, data processing, improved energy model and hieratical organization of nodes. Sensing model in MannaSim is very simple and it represents a random generator for creating sensing values (standard deviation of average value). Improved energy model includes additional sensing and data processing consumption (*constant_device_consumption* * *time_of_operation*). Mannasim provides JAVA tool for generating simulation scenarios.

4.2. OMNeT++/Castalia

The OMNeT++ is a C++ discrete event simulation environment [11] publicly available since 1997. OMNeT++ was designed to be as general as possible and it is not a WSN simulator. OMNeT++ provides a basic machinery and tools to write simulations, but itself it does not provide any components aimed at computer network simulations, queuing network simulations, system architecture simulations or any other area. Because of its generic and flexible architecture, various applications can be supported by designing additional simulation models and frameworks such as INET/INETMANET, MiXiM or Castalia.

The simulator as well as user interfaces and tools are highly portable. They are tested on the most common operating systems (Linux, Mac OS/X, Windows), and they can be compiled out of the box or after trivial modifications on most Unix-like operating systems.

OMNeT++ was designed from the beginning to support network simulation at a large scale, keeping in mind that simulation models need to be hierarchical and simulation software itself should be modular, customizable and should provide for embedding simulations into larger applications. OMNeT++ contains an Integrated Development Environment (IDE) that provides rich environment for editing, debugging simulation and analysis, visualization of simulation results. Tkenv, the GUI user interface of OMNeT++, is a simulation GUI tool that supports interactive simulation execution, animation, tracing and debugging.

The major drawback of OMNeT++ was the lack of available protocols compared with other simulators. However, in recent years OMNET++ is becoming a popular tool and its lack of protocol models is being cut down by recent contributions.

Castalia [12] is one of many simulators built on the top of OMNeT++. It is specifically designed for WSN, encompassing all important aspects of the system and providing the most accurate modeling available in the research community, starting with the communication models (i.e., wireless channel and radio models) [13]. This simulator enables support for Wireless Sensor Networks (WSN), Body Area Networks (BAN) and generally networks for low-power embedded devices. Purpose of this generic C++ simulator is to test user's algorithms and protocols.

Adopting the principle of OMNeT++, sensor nodes are built as compound modules, whose sub-modules represent the wireless sensor node stack. Scalability in Castalia is not an issue, modular approach enables high scalability. Memory usage increase and simulator speed decrease tend to show linear dependencies as networks become bigger.

Nodes do not connect to each other directly but through the wireless channel module. This module, upon receiving a packet, decides which nodes should receive the packet. The whole model of sensing is the most realistic in Castalia. For sensing purpose nodes are connected to adequate physical process module (one instance of module for each physical process), to get their sensor readings. Nodes can acquire multiple sensing devices (multiple sensing modalities). Sensors are implemented with a high level of details (sampling rate, devices Bias, sampling Noise of device, resolution of measurement etc.). Complex Sensing model is defined in time and space on sensing phenomena. Deployment of nodes can be set manually or

automatically (uniform or randomize grid distribution in 2D or 3D area). After initial deployment, mobility manager (module) specifies how the nodes will move through the space (mobility pattern). Mobility of node must be set manually in configuration files, only one mobility pattern is available (line pattern) [14].
Energy model in Castalia includes power consumption depending on the state of the sensor node (active, shallow sleep, deep sleep etc). These differences in consumption influence the radio module, where power is used for sending and receiving packets. Also energy consumption is calculated for the transition processes between states and for the sensing process of the sensor node. Sensor power usage is modeled for every type of sensor as consumption per single sample of phenomena.
In previous years, because of small number of implemented protocols, this simulator was not so popular (initially routing was seen as a less important feature so no module was introduced). Nowadays, when the set of protocols is extended, Castalia stands for one of the simulators with most growing popularity.

4.3. SWANS/ SIDnet

SWANS [15] is a discrete event, wireless network simulator build on the top of JiST platform. JiST is basically a simulation kernel which enables Java based simulation environment. JVM is modified to run programs in simulation time instead of real time. Models in SWANS are written and compiled in Java. Afterwards, compiled code is rewritten by embedding simulation engine, in order to execute simulator on standard JVM. This approach of virtual-machine simulation enables high portability and merging high-level languages with simulation semantics. Aforesaid allows the execution of already written JAVA applications on the simulator.
SWANS has a modular architecture. Components are defined for different layers of *node stack*. Communication among layers is achieved by message exchange (simulated network packets). Message is a Java object that encapsulates a higher level message to mimic the chain of packet headers. In SWANS there is no unnecessary overhead in the intercommunication model among layers of node stack. Message is a shared object that is passed by reference to avoid copying.
Deployment field is partitioned into grid of node bins. When some node is transmitting signal, the Field component uses transmitting node position in the grid, and information about signal strength to determine the neighboring nodes that are capable of receiving the signal. That way, the number of communication dependencies to be checked at every simulation interval is significantly lower then n^2. The radios of nodes are parameterized with frequency, reception threshold, bandwidth, error models, transmission power and antenna gain. Node mobility is automatically supported. Nodes can move to random position, in random direction or by "teleport" model.
The latest version of SWANS has been released in March 2005. Lack of active development and small number of supported protocols have caused SWANS to be less popular then other simulators.
Despite the fact that SWANS is written in Java (Java performance tends to be lower than C++ in tasks of memory handling), this is the highly scalable simulator [16]. SWANS can handle network of 10^6 wireless nodes. High network scalability is achieved mainly by SWANS architecture and cross-layer optimization. In [9] it has been presented that this simulator outperforms the NS-2 in execution time of simulation and memory consumption.
SIDnet-SWANS (Simulator and Integrated Development Platform for Sensor Networks Applications) [17], [18] is a Java general purpose WSN simulator built on the top of the architecture of JiST/SWANS. This extension enables battery, sensing, GPS and other WSN component. SIDnet provides the GUI interface that allows run-time interaction with simulation on various levels. Through the GUI user can observe phenomena fluctuation, network topology, energy level of network, network statistic, change phenomena dynamic and simulation speed, inspect and influence on individual nodes and query the network. This flexible graphical user interface provides visual feedback of almost all important aspect of the sensor network in real time. In SIDnet user can attach virtual terminal and query the phenomena of interest, this is done by implementing high-level language structures that is similar to the syntax of TinySQL.
Deployment of the nodes can be manual (single node deployment), random automatic (uniformly distributed, bi variate distributed) or automatic fly-by distributed (nodes are being deployed along given trajectory) [19].
Energy consumption model is highly realistic. It takes into account *operational energy consumption* and *battery drain*. Also there is possibility to recharge the battery. Operational energy is used for radio (active and sleeping consumption), processing (active and idle consumption of CPU) and sensing purposes. Battery consumption, in all mentioned cases, is realized as a dependency of elapsed time and different levels of consumption.
One large drawback of SIDnet is extremely poor documentation.
Table 1 shows briefly the characteristics of the analyzed simulators following the proposed methodology and criteria.

	Features		NS-2/MannaSim		OMNeT++/Castalia	SWANS/SIDnet	
Non-Functional	License		GNU GPL		ACADEMIC PUBLIC	ACADEMIC PUBLIC	
	Popularity		most popular		with rising popularity	not so popular	
	Platform		Linux, FreeBSD, SunOS, Solaris, Windows (using Cygwin)		Linux, Mac OS/X, Windows (using Cygvin)	JVM required	
	GUI tools		animation	simulation modeling	IDE and tools for: animation, debugging (code, interaction with simulation), simulation modeling	animation, debugging (interacting with simulation, query SN)	
	Prog. language		C++, Otcl		C++	Java	
	Active development		yes		yes	no	
	Documentation		good		good	bad	
	Parallel execution		no		no	no	
	Extensibility		high		high	high	
Functional	Possibility of usage		generic		generic	generic	
	Time dimension		discrete event		discrete event	discrete event	
	Scalability		low		high	highest	
	Intercation with SN		no		no	no	
	WSN platforms	Models	no		no	no	
	Deployment		manual, random		manually, random (uniform, grid distribution)	manual, random (uniform, bi variate, from plane)	
	Mobility		manual, random		manual	random (position, direction, teleport)	
	Radio channel		free space, two-ray ground reflection, shadowing		empirically measured path losses , map of path loss, lognormal shadowing , temporal variation, unit disk	zero (none), releigh and rician sigal fading, free space, two-ray and table driven path loss	
	Physical layer		Lucent WaveLan DSSS	+Crossbow Mica2	Texas Instruments CC2420 and CC1000	no data	
	Data Link Layer		IEEE 802.11(several implementations), Preamble based single hop TDMA protocol		Tunable MAC, TMAC, SMAC, IEEE 802.15.4, IEEE 802.15.6	IEEE 802.11b	+IEEE 802.15.4
	Network Layer		DSDV,DSR,TORA, AODV,PUMA	+LEACH, +Directed Diffusion	Simple Tree, Multipath Rings	ZRP, DSR, AODV, Single route, Tree, Multipath routing	
	Transport Layer		TCP, UDP		not supported	UDP,TCP	
	Energy consumption		simple, txPower, rxPower	good, Tx,Rx, sensing, processing	good, Tx, Rx, sensing (various), SN modes	recharge, battery drains, Tx, Rx, processing, sensing, SN modes	
	Sensing		with MannaSim only, simple, random numbers, no sensors and environment		complex (sampling rate, dev. bias, noise, resolution), multiple sensors, various phenomena fields	complex, multiple sensors, various phenomena fields	

Table 1 – Analyzed characteristics of the simulators

5. CONCLUSIONS AND GUIDELINES FOR SIMULATOR SELECTION

Almost all available simulators are network simulators with or without wireless capabilities. Therefore, they are extended in order to support sensing, radio and energy models, which are basic WSN simulation requirements, resulting in Mannasim, Castalia, and SIDnet extensions.

NS-2/Mannasim, OMNeT++/Castalia and SWANS/ SIDnet are generic purpose simulators which can be used to develop, compare and learn communication protocols/algorithms.

Regarding the energy models, SIDnet has the most comprehensive model of battery consumption, including battery drains.

Since the radio is the biggest energy consumer, simulators must predict consumption for different radio mode (shallow sleep, deep sleep, active…) which is supported in Castalia and SIDnet.

Generally speaking, the number of implemented protocols is highly related to the popularity of the software: the more models implemented, the less time for WSN development is needed. NS-2 is a simulator with best performances regradning protocol models implemented, while is the worst one.

In general, it is expected tthat simulators implemented in C/C++ tend to show better scalability than those that are implemented in Java, but there are exceptions like SWANS which scales surprisingly well [3]. Simulators with component-based architecture scale better than object-oriented architectures due to the possibility of parallel execution.

When the aim of simulation is not only to to test a protocol, the sensing models of higher level of details are needed. Mannasim does not simulate sensing phenomena and its sensing model is very simple (generator of random values), while SIDnet both have complex sensing models, where multiple sensors on a single node collect data from environment. GUI support is preferable, but not a necessary condition when choosing a simulator. NS-2/Mannasim has simple GUI environment (simulation animation and modeling). OMNeT++ and SIDnet provide powerful GUI which supports interactive simulation execution, animation, tracing and debugging. SIDnet allows

queries through the network to be sent in order to get sensing information.

Due to specifics of the particular WSN application and,consequently, the aspects that should be studied, it is not possible to give an unambiguous guideline for selection of WSN simulator in general. However, the presented analysis can provide some useful recommendations which are as follows. Firstly, the selection criteria can be classified in two groups according to their importance.
The first group, which is essential and must fulfill all user requests, comprises following simulator features: communication protocol models, energy consumption models, sensing models and scalability. The second group comprises the features that are preferred but not essential to selection of simulator. These are: popularity of the simulator, user's familiarity with implemented programming languages, GUI availability, proper documentation and active development .

Based on the essential features, we suggest following recommendation for the open source simulators that have been analyzed in this paper.
NS-2 simulator is the most suitable solution regarding the communication protocols criterion.
Castalia has the most realistic sensing model.
SIDnet simulator handles complex aspect of energy consumption and copes best with scalability of sensor field.

REFERENCES

[1] K. Wehrle, M. Günes, and J. Gross, *Modeling and Tools for Network Simulation*, Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2010.

[2] E.M. Shakshuki, H. Malik, and T.R. Sheltami, "A comparative study on simulation vs. real time deployment in wireless sensor networks," *Journal of Systems and Software*, vol. 84, 2011, pp. 45-54.

[3] M. Mekni and B. Moulin, "A Survey on Sensor Webs Simulation Tools," *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, Cap Esterel, France: 2008, pp. 574-579.

[4] E. Egea-Lopez, J. Vales-Alonso, A.S. Martinez-Sala, P. Pavon-Marino, and J. García-Haro, "Simulation tools for wireless sensor networks," *Proc. Int'l. Symp. Perf. Eval. of Comp. and Telecommun. Sys*, pp. 559–66.

[5] M. Jevtić, N. Zogović, and G. Dimić, "Evaluation of Wireless Sensor Network Simulators," *Proceedings of the 17th Telecommunications Forum (TELFOR 2009), Belgrade, Serbia*, Belgrade,Serbia: 2009.

[6] I. Khemapech, A. Miller, and I. Duncan, *Simulating Wireless Sensor Networks*, Technical report, School of Computer Science, University of St Andrews, 2005.

[7] "The Network Simulator - ns-2," *http://www.isi.edu/nsnam/ns/*.

[8] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*, Boston, MA: Springer US, 2009.

[9] I.T. Downard, *Simulating sensor networks in ns-2*, Citeseer, 2004.

[10] R. Barr, "Swans-scalable wireless ad hoc network simulator, User Guide," Mar. 2004.

[11] "MannaSim Framework," *http://www.mannasim.dcc.ufmg.br*.

[12] "OMNeT++ Network Simulation Framework," *http://www.omnetpp.org/*.

[13] "Castalia," *http://castalia.npc.nicta.com.au/*.

[14] D. Pediaditakis, Y. Tselishchev, and A. Boulis, "Performance and scalability evaluation of the Castalia Wireless Sensor Network simulator," *3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools2010)*, Malaga/Spain: 2010, pp. 271–275.

[15] A. Boulis, "Castalia A simulator for Wireless Sensor Networks and Body Area Networks, User's Manual," Aug. 2010.

[16] A.H. Network, "Scalable Wireless Ad Hoc Network Simulation," *Handbook on theoretical and algorithmic aspect of sensor, ad hoc wireless, and peer-to-peer networks*, 2006, p. 297.

[17] E. Schoch, M. Feiri, F. Kargl, and M. Weber, "Simulation of ad hoc networks: ns-2 compared to JiST/SWANS," *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, pp. 1–8.

[18] "Projects - SIDnet-SWANS," *http://users.eecs.northwestern.edu/~ocg474/SIDnet.html*.

[19] O.C. Ghica, "SIDnet-SWANS, User Manual," Mar. 2010.

[20] O. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischoff, and N. Valtchanov, *SIDnet-SWANS: A Simulator and Integrated Development platform for Sensor Networks Applications*, 2006.

Acknowledgments

Results presented in this paper are part of the research conducted within the Grant No. III-43007, Ministry of Science and Technological Development of the Republic of Serbia.