G-JSIM – A GUI TOOL FOR WIRELESS SENSOR NETWORKS SIMULATIONS UNDER J-SIM

[1]Paulo A. C. S. Neves, [2]Iúri D. C. Veiga, [3]Joel J. P. C. Rodrigues

[1]Superior School of Technology, Politecnic Institute of Castelo Branco, Avenida do Empresário, Castelo Branco, Portugal
[1,2,3]Department of Informatics, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
[1,3]Institute of Telecommunications - Networks and Multimedia Group, Portugal
[1]pneves@est.ipcb.pt, [2]iuri.d.veiga@gmail.com, [3]joel@ubi.pt

ABSTRACT

A Wireless Sensor Network is composed of up to thousands of smart sensing nodes with processing unit and memory, sensing unit and wireless communication capabilities. Wireless Sensor Networks application spans from the military applications into almost every field we can think of. Several simulation tools are readily available and, among them, the J-Sim is a java-based simulator with growing interest by research and network developers alike. In this paper, we propose to enhance J-Sim functionality with a Guided User Interface for Wireless Sensor Networks that dramatically increases the user-friendliness of the simulator. Also, we provide a free download web page for everyone to benefit.

Index Terms— Wireless Sensor Networks, J-Sim simulator, Guided User Interface, XML-based Simulation Tools, Tcl

1. INTRODUCTION

A Wireless Sensor Network (WSN) [1] is a network of interconnected nodes with sensing capability, self-powered, with a processing core and wireless communication capabilities. The intelligent sensors scan the environment and provide useful information transmitted over to a special node: the sink node. Although there are some designs that are sinkless, the sink node is present in the large majority of deployments.

WSNs are expanding its applications from military into almost every application we can think of, like environment applications, habitat monitoring, industry and business application and smart home, smart office, smart system. Recently a great interest is draw to medical applications [2], namely personal monitoring and biofeedback [3].

Simulation is a relatively fast means to obtain an estimate of network performance and tuning. Three of the most used simulators for WSNs are Network Simulator 2 (ns-2) [4], Java Simulator (J-Sim) [5] and Sensor Network Emulator and Simulator (SENSE) [6]. A comparative study about these simulators was presented in [7].

The paper elaborates on the development of a Guided User Interface (GUI) for the specification of WSNs simulation under J-Sim, called G-JSim. Our approach provides the specification of a network, through pre-defined guided steps, simulation parameters and network topology storage and retrieval, automatic Tcl file creation and issues J-Sim to perform the simulation itself. We consider the user friendliness of the solution, by providing a structured GUI, based on the different node types and grouping similar simulation parameters.

The remainder of the paper is structured as follows. Section 2 surfaces the J-Sim simulator, while section 3 presents our solution in the user perspective. Section 4 elaborates on G-JSIM internals and, finally, section 5 concludes the paper.

2. THE J-SIM SIMULATOR

J-Sim is developed in the Ohio State University and it is an open source, component-based compositional network simulation environment that is developed entirely in Java. The simulator provides many advantages over the "classic" ns-2 approach, namely with the use of the Autonomous Component Architecture (ACA) [8] that enables component independence.

The ACA uses a model that resembles integrated circuit assembly, since components have ports to communicate between each other and can be constructed independently. J-Sim uses Java to implement all the cases and Tool Command Language (Tcl) is used as a linking script language that enables construction, configuration and/or network simulation at run-time.

In [8], the authors state that J-Sim is much more scalable than ns-2 (with emphasis on memory usage). J-Sim is also a dual-language simulator like ns-2 (J-Sim uses Java and Tcl,

while ns-2 uses C++ and Tcl), the classes/methods/fields in Java need not be explicitly exported to be accessed in the Tcl environment.

J-Sim uses the simulation model depicted in Figure 1. The target node sends stimulus information for the WSN to capture: the sensor nodes must cooperate in order to retrieve meaningful data and send it to the sink node. With this model, it is possible to simulate several WSN scenarios.

Target nodes have only one communication channel, the sensor channel, since they only send stimulus to the sensor network, the sensor nodes communicate in two ways, sensor and wireless channel, and finally the sink nodes only communicate in the wireless channel.

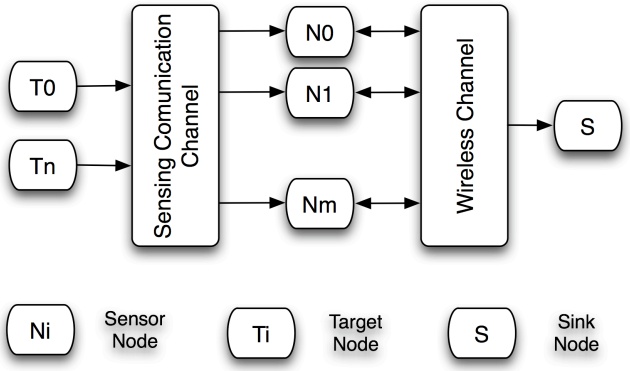

Figure 1. J-Sim simulation model for WSNs.

To the best of our knowledge there are no dedicated applications for guided creation of simulation scenarios, namely an application that can capture WSN parameters graphically, build the corresponding Tcl file and automatically launch the simulation.

3. G-JSIM IN USE

The graphical interface for Java Simulator is built in Java. Since J-Sim already needs a java Runtime environment, there is no need to install additional frameworks. We provide a downloadable zip archive of the application at http://e-projects.di.ubi.pt.

Figure 2 shows the main screen interface of G-JSim. In this screen, the user can configure the top level parameters of the Wireless Sensor Network, such as the number of sensor and target nodes (sink nodes are hardwired to 1 at the moment), the method to define the network topology, and the simulation time. This first screen also requests what the propagation model will be and the network grid size. An asterisk close to a parameter or button indicates that changes aren't saved yet.

After initial network setup, individual network components can be set up. The sensor node parameters input is divided into three main areas: Power Model, Sensor Mobility Model and Sensor Function model. Power Model has three different areas: the CPU model, the Radio model, and the Battery model as may be seen in Figure 3.

Figure 4 presents the target node's simulation parameters. These are divided into Topology (that defines sensor movement boundaries), the initial position of the node and the Destination Mode (the model that is used to calculate node movement), the Target Agent Layer and Sensor Physical Layer.

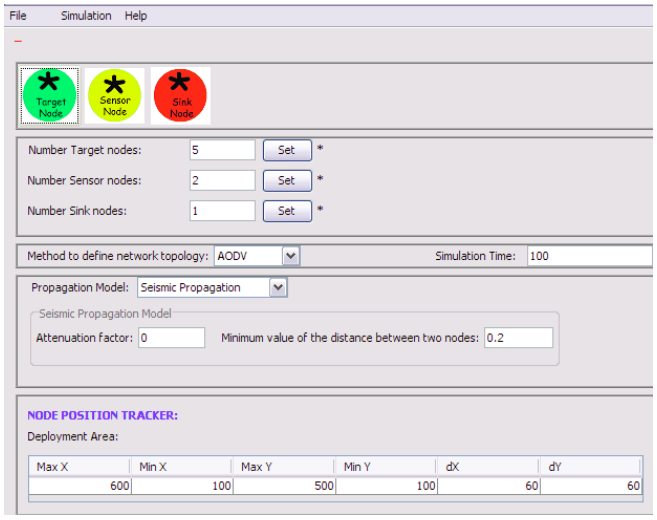

Figure 2. G-JSim main screen interface.

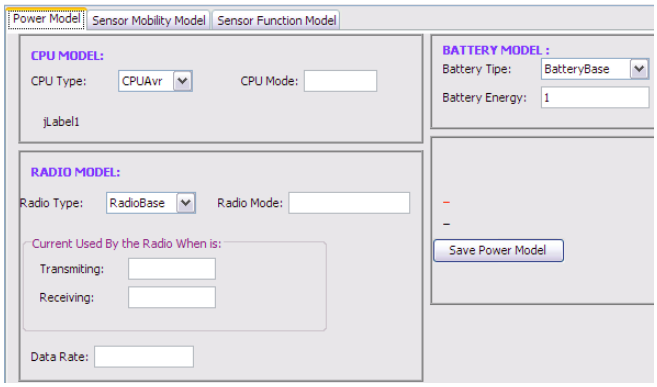

Figure 3. Sensor node's simulation parameters input.

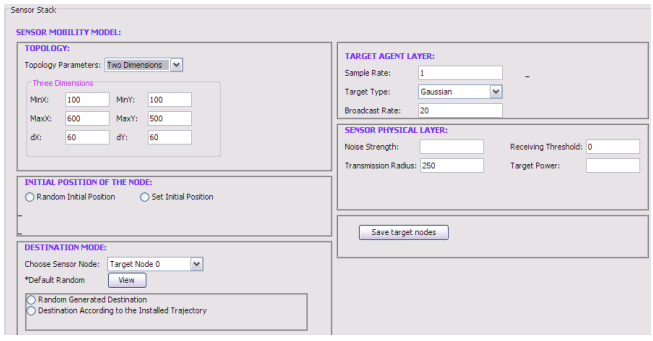

Figure 4. Target node's simulation parameters input.

Finally, the sink node's simulation parameters are shown in Figure 5. The sensor application part deals with Coherent

Threshold, and node position, while Queue allows queue parameter's specification. Finally Wireless Phy sets physical layer parameters and RTS threshold of Mac 802.11 can also be set. If more than one node type is selected the interface allows personalization of individual nodes, bounded by J-Sim capabilities. For instance, the user can define the individual location of each node, but sink nodes are limited to one per simulation at this time.

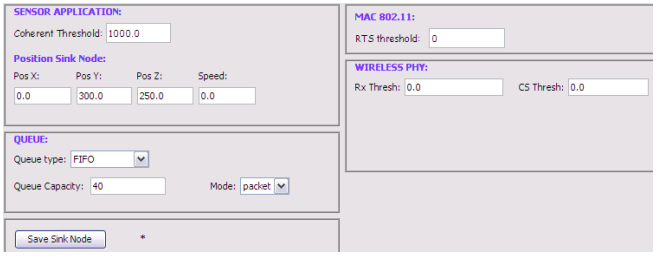

Figure 5. Sink node's simulation parameters input.

After simulation parameters input, the application is ready for the next step. The user can save the current simulation parameters in an eXtensible Markup Language (XML) file. Figure 6 shows the beginning of one XML file. This file can latter be loaded, avoiding the burden of input previous simulation parameters used and allowing fine adjustments after application shutdown.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Data_Simulation>
        <netWorkTopologyType>AODV</netWorkTopologyType>
        <propagationType>Seismic Propagation</propagationType>
        <simulationTime>600.0</simulationTime>
        <d0>0.2</d0>
        <atnFactor>0.0</atnFactor>
        <flagD0>true</flagD0>
        <flagAtnFactor>true</flagAtnFactor>
        <sensorNodeMaxX>0.0</sensorNodeMaxX>
        <sensorNodeMinX>0.0</sensorNodeMinX>
        <sensorNodeMaxY>0.0</sensorNodeMaxY>
        <sensorNodeMinY>0.0</sensorNodeMinY>
        <capacityWirelessChannel>0</capacityWirelessChannel>
        <nGrids>0</nGrids>
        <nodePosMaxX>600.0</nodePosMaxX>
        <nodePosMaxY>500.0</nodePosMaxY>
        <nodePosMaxZ>0.0</nodePosMaxZ>
        <nodePosMinX>100.0</nodePosMinX>
        <nodePosMinY>100.0</nodePosMinY>
        <nodePosMinZ>0.0</nodePosMinZ>
        <nodePosDX>60.0</nodePosDX>
        <nodePosDY>60.0</nodePosDY>
```

Figure 6. Illustration of XML file that stores simulation parameters.

All simulation options are according to J-Sim simulation engine, November 2007 version. After the XML file creation the user can issue a simulation. J-Sim accepts a Tcl file for simulation input. We generate the Tcl file based on a Tcl WSN simulation file by Ahmed Sobeih [10].

Figure 7 shows an extract of the Tcl file generated on one of our performed simulations. The file starts by informing J-Sim that a WSN with 4 sensor nodes and a sink node must be simulated, and 2 target nodes are used to provide stimulus for this network.

```
#Number of Target Nodes
set num_target_nodes 2

#Number of Sensor Nodes
set num_sensor_nodes 4

#Number of Sink Nodes
set num_sink_nodes 1

set sink_id 0

#Number of Nodes (sensor + target + sink)
set num_nodes 7

#Create the sensor Channel
mkdir drcl.inet.sensorsim.SensorChannel sensorChannel

# Capacity of the sensor channel is total number of nodes (sensors + targets)
! sensorChannel setCapacity $num_nodes

#Create the propagation model
mkdir drcl.inet.sensorsim.SeismicProp seismicProp

#Sets the minimum value of the distance between two nodes
! seismicProp setD0 0.2
```

Figure 7. Illustration of Tcl file used for simulation.

Finally the simulation takes place. J-Sim is automatically invoked and a third file is created with log information, as Figure 8 shows. We named this file with extension `.sim`.

```
TCL0> create sink 0
create sensor 1
create sensor 2
create sensor 3
create sensor 4
create target 5
create target 6
simulation begins...
Target 5: generating signal at time 0.05 loc: 550.0, 250.0, 0.0
Target 6: generating signal at time 0.06 loc: 400.0, 450.0, 0.0
Target 5: generating signal at time 20.05 loc: 186.55257801708888,
330.79967725171025, 0.0
Target 6: generating signal at time 20.06 loc: 510.8940036922546,
265.5468162323655. 0.0
```

Figure 8. Illustration of the simulation log file.

When simulation ends, J-Sim shows the results in a graphical form, in signal-to-noise (SNR) ratio charts, like Figure 9 shows. Each graph corresponds to a target node.

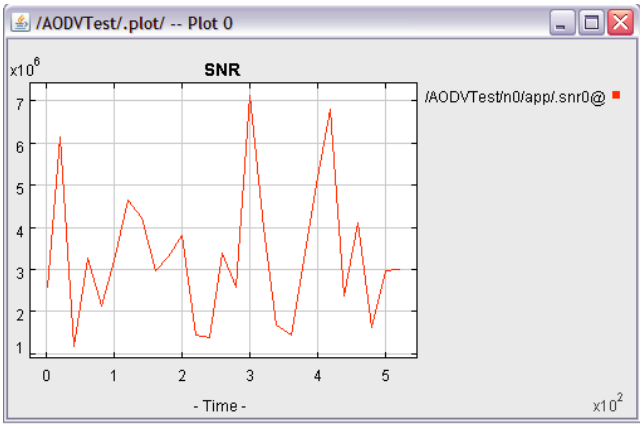

Figure 9. SNR ratio illustration.

4. G-JSIM ARCHITECTURE

G-JSIM is entirely built over the Java development platform, JDK 6 Release 5 and we used the Netbeans 6.0.1 IDE. Figure 10 shows the G-JSIM class diagram. The classes *SinkNodes*, *SensorNodes* and *TargetNodes* have the attributes and methods of their respective node type. Class *AllVariables* groups all of them, with other simulation parameters not directly related to the nodes.

Classes *TargetNodeGUI*, *SensorNodeGUI* and *SinkNodeGUI* handle the Guided User Interface of their respective node types, while *MainInterface* is the core of the graphical interface.

In terms of files, there are classes that open existing Tcl (*OpenTcl*) and XML (*OpenXml*) files, and classes that create and save the current Tcl (*GenerateTcl*) and XML (*GenerateXml*) data.

Finally, the class *FileTypeFilter* allows restriction of file extension when browsing for a file, and the class *CreateTables* manage the tables created for the sensor's mobility.

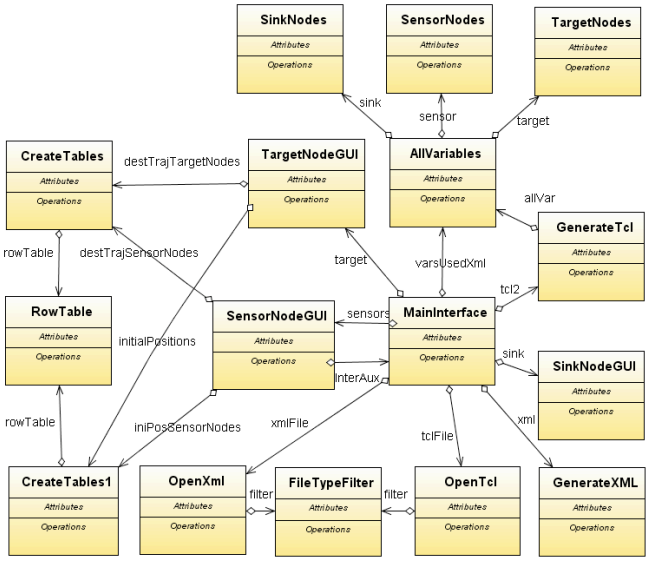

Figure 10. G-JSIM classes diagram.

5. CONCLUSIONS AND FUTURE WORK

We presented a GUI application, called G-JSim, for WSN simulation tool with direct and automatic interface with the J-Sim simulation engine. The tool is freely available for download, and grabs user input parameters, storing them into a XML file and automatically launches J-Sim to perform simulations.

With the growing use of sensors in consumer electronics, this tool might help further developments on this area, providing reliable and easy-to-get information before an actual deployment of the sensor network.

As a future work, in the next G-JSIM version, a new user-friendlier interface should be created, namely, with a graphical introduction of the nodes and take advantage of possible J-Sim updates at the time.

ACKNOWLEDGEMENTS

Part of this work has been supported by the Group of Networks and Multimedia of the Institute of Telecommunications – Covilhã Lab, Portugal, and by the Euro-NF Network of Excellence of Seven Framework Programme of EC.

REFERENCES

[1] I. F. Akyldiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: a Survey," *Computer Networks (Elsevier)*, vol. 38, pp. 393-422, 2002.

[2] The promise of Wireless Sensor Networks for Medicine, http://www.intel.com/research/exploratory/wireless_promise.htm, Accessed in October 2007.

[3] S. Kroc and V. Delic, "Personal Wireless Sensor Network for Mobile Health Care Monitoring", presented at 6th international Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service, Serbia and Montenegro, 2003.

[4] NS-2 Networking Simulator, http://www.isi.edu/nsnam/ns/, Accessed in January 2007.

[5] The J-SIM Simulator, http://www.j-sim.org, Accessed in January 2007.

[6] SENSE - Sensor Network Simulator and Emulator, http://www.ita.cs.rpi.edu/sense/index.html, Accessed in March 2007.

[7] Paulo A. C. S. Neves, Joel F. P. Fonseca, and Joel J. P. C. Rodrigues, "Simulation Tools for Wireless Sensor Networks in Medicine: a Comparative Study", in Proceedings of BIOSTEC 2008 - International Joint Conference on Biomedical Engineering Systems and Technologies, Funchal, Madeira-Portugal, January 28-31, 2008.

[8] The Autonomous Component Architecture, http://www.j-sim.org/whitepapers/aca.html, Accessed in April 2007.

[9] A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, "J-Sim: A Simulation Environment for Wireless Sensor Networks", presented at 38th Annual Simulation Symposium (ANSS '05), San Diego, CA, USA, 2005.

[10] Ahmed Sobeih Home Page, https://netfiles.uiuc.edu/sobeih/www/, Accessed in September 2007.