2009 First International Conference on Advances in System Simulation

# G-SENSE – A GRAPHICAL INTERFACE FOR SENSE SIMULATOR

Pedro M. P. Rosa
Department of Informatics
Univ. of Beira Interior
Covilhã, Portugal
pedropinto@ubi.pt

Paulo A. C. S. Neves
Instituto de Telecomunicações
Univ. of Beira Interior
Pol. Inst. Castelo Branco
Castelo Branco, Portugal
pneves@est.ipcb.pt

Binod Vaidya
Instituto de Telecomunicações
Covilhã, Portugal
bnvaidya@co.it.pt

Joel J. P. C. Rodrigues
Instituto de Telecomunicações
Department of informatics
Univ. of Beira Interior
Covilhã, Portugal
joel@ubi.pt

*Abstract*— **Wireless sensor networks greatly benefit from simulation before deployment, since some of these networks may contain thousands of nodes. The new challenges compared to traditional computer networks led to several approaches for network simulation, namely SENSE – Sensor Network Emulator and Simulator. However this approach presents a limited user interface, namely based on text, forcing users to have knowledge on C++ programming language. This paper presents a tool, called G-Sense, that greatly improves SENSE user friendliness, with graphical input of simulation parameters, save and load simulation features, and simulation results management with plot view. This new tool uses SENSE simulation engine in a transparent way, so the user may be focused on the simulation itself, not in the underlying simulation tool. We present G-Sense architecture, usability and extensive experiments for its validation. We believe that this tool will contribute for SENSE adoption for wireless sensor network simulation, clearly improving on its ease of use.**

***Keywords- Wireless sensor networks, SENSE simulator, Guided User Interface, Visualization of networks***

## I. INTRODUCTION

A Wireless Sensor Network (WSN) [1] is a network of interconnected nodes with sensing capability, self-powered, with a processing core, memory, 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.

Network simulation is a relatively fast way to obtain an estimate of network performance and tuning. Three of the most used simulators for WSNs are Network Simulator 2 (ns-2) [2], Java Simulator (J-Sim) [3], and Sensor Network Emulator and Simulator (SENSE) [4, 5]. A comparative study about these simulators is presented in [6], where the authors conclude that SENSE presents a simpler approach, although not as flexible as the other two.

Since WSN can reach thousands of nodes, information can be overwhelming. To better understand the output, there are some problems that may occur while developing protocols and algorithms for networks, which may not be identifiable without a way to actually see what is happening in the network. Without the help of visualization, the designer would often need to analyze pages of simulation printouts, to draw a picture of what happened in the simulation, and where relies the opportunities for improvement.

Protocols for wireless networks are especially sensitive to environment perturbation and temporary failures. Even a small change in protocol design may result in a huge change in network performance. The adequate visualization of simulation results helps understand, and afterwards control the implication of simulation parameters and protocol changes.

Equally important is the help that visualization can provide in understanding and teaching how the sensor network operates and how its protocols and algorithms really work. Specially in wireless networks, routing and distributed computing is usually a complex and dynamic process which is difficult to conceptualize and, therefore, difficult to teach [7].

SENSE is a statistical simulator, so simulation results may not fit some user requirements. However, the major concern with SENSE is user-friendliness. The simulator relies on command line parameters, C++ compiler and makefiles to run. Moreover, simulation results are displayed directly on the terminal. This is the main reason to develop a tool for SENSE, improving user interface, featuring simulation management, and improving overall user experience.

We focus on wireless network simulation guided user interface since SENSE is specific for such networks. From the developer team it is not expected an extension to other kinds of networks in the near future. As a result we are focused on this paper on wireless sensor networks simulation.

G-Sense application is a tool developed to assist in simulation scenario creation, management, results visualization, and their comparison. When used with SENSE, G-Sense provides a user-friendly simulation scenario creation, using a graphical view and input of simulation parameters, interacting with SENSE without any further configuration. Moreover, G-Sense gives native save and load features with persistent storage, and Adobe® Acrobat® Portable Document Format (PDF) for simulation results files. With G-Sense the user rapidly identifies if the SENSE simulator corresponds to its needs without the need to dig on code, use command-line parameters, or even have knowledge of any programming language.

A good example of a similar approach but for J-SIM simulator, is presented in [8]. The authors developed a GUI tool for WSN simulation over J-SIM, another powerful simulation tool. However, such approach does not feature simulation results file storage for latter analysis. Due to the different simulation parameters and output of SENSE and J-SIM, we were unable to perform code reuse, even at the GUI level. However, much has been learned from the development of this tool. Moreover, since G-JSim and J-SIM are both Java-based, code reuse under .NET Compact Framework, namely with suppression of J# on the 2008 version, proves to be not trivial.

The remainder of the paper is organized as follows. Section II briefly discusses related works whereas Section III describes SENSE simulator, pointing its major drawbacks in simulation management and visualization. Section IV presents G-Sense architecture, while Section V draws on G-Sense user interface, operation and features. Section VI shows application testing and validation and finally, Section VII concludes the paper.

## II. Related Work

In this section we briefly discuss some previous applications for input of simulation parameters, result visualization and management. However, our approach is unique in nature, since SENSE itself is very different from other simulators, from data presentation (summary in nature, due to the statistical nature) to simulation parameters.

Several GUI supporting tools such as nsBench [9], gEditor [10], and G-JSim [8] have been developed. nsBench is a drag-and-drop GUI tool for the world wide known network simulator 2 (ns-2). The tool enables creating, analyzing and visualizing ns-2 simulation scripts and traces automatically, providing a complete GUI.

The Java Simulator (J-Sim) gEditor is a graphical editor for J-Sim that uses the J-Sim simulation engine to actually run the simulation. It also enables graphical creation and edition of a simulation model, enabling control over the simulation while it is running (a simulator feature), while also featuring management.

Finally G-JSim [8] was also developed over the J-Sim simulator. This tool was designed specifically for wireless sensor networks simulation, featuring simulation parameters management and guided input of simulation parameters.

Since SENSE is a statistical simulator with command-line based interaction, it is desirable to develop a GUI tool for it. However, above-mentioned GUI tools are not suitable for SENSE as simulation engines are different in nature. Thus we intend to develop a GUI tool for SENSE that enables creating, analyzing and visualizing SENSE simulation. SENSE is focused on wireless networks simulation, so our tool is limited to SENSE functionality at the moment. However if the user wishes so, introducing new protocols can augment SENSE and G-Sense can also cope with it.

Although on a different area, authors of [11] present a MATLAB/SIMULINK-based GUI for simulation of circuits. Simple, but also powerful the GUI is also based on another tool like G-Sense, serving the purpose of improved user experience.

## III. SENSE Simulator

SENSE is a simulator created (from scratch) for simulation of WSN, providing scalability and extensibility. This simulator uses component-port model to represent parts of a network, thus featuring extension capabilities [12]. In this section we briefly present some background content on the SENSE simulator.

### A. SENSE Simulation Architecture

SENSE simulation tool uses components that exchange information through ports. Two types of ports are considered – the *inports* that are functional in nature, implementing certain functionality, similar to a function; and the *outports* that acts like a function pointer, defines the functionality that it expects from others (similar to an interface). The *inports* and *outports* of the sensor node in SENSE are directly connected to the corresponding *outports* and *inports* of internal components.

SENSE uses a simulation model based on the sensor node model depicted in Figure 1, which can be extended. With this model, it is possible to simulate several WSN scenarios.

In SENSE, a sensor node is a composite component that features a number of smaller primitive components. Usually, a sensor has some layered network protocol components (PHY, MAC, NET and APP), a power and battery components related to power management, and others, such as MOBILITY and SENSOR [5].

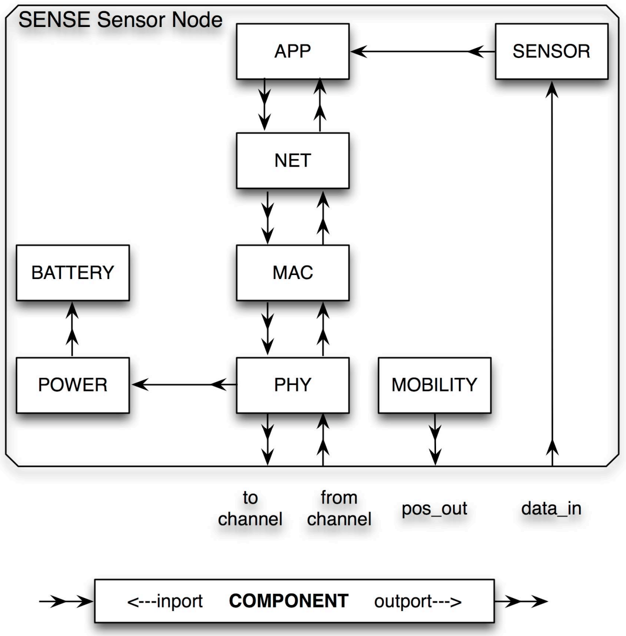


Figure 1. The internal structure of a typical sensor node model for WSNs in SENSE [5].

The parameters to build the simulation are chosen at the SENSOR component level. The MOBILITY component gives the localization of the sensor in the sensor grid.

### B. *Visualization with SENSE*

SENSE was developed as a statistical analysis tool, providing results directly on the terminal interface (SHELL environment), without any native file output for permanent storage and latter analysis. Authors of [13] propose a possible solution for visualization based on *iNSpect* that presents graphical representation of simulation results. However, it does not cope with limitations of the simulation creation process.

Figure 2 presents an example of SENSE execution output. Although the basic levels of information are useful and necessary to understand what happens during network simulation, it is far from ideal. Moreover, simulation management is a very cumbersome task for the user, redirecting terminal output, and selecting appropriate file names. In addition, to compare operations between several simulations' results, as well as obtain previous results, requires new simulations.

```
StopTime: 60, Number of Nodes: 5, Terrain: 30 by 30
Number of Sources: 2, Packet Size: 20, Interval: 5.000000
X: 30.000000, Y: 30.000000, gridsize: 1106.166755
Grid disabled
@ 0.000000 0 6.067256 0.436219
@ 0.000000 3 11.829650 5.393423
@ 0.000000 4 14.538638 12.263121
@ 0.000000 2 26.196943 6.898405
@ 0.000000 1 22.536475 15.502456
APP -- packets sent: 52, received: 52, success rate: 1.000, delay: 0.001
52.000000 52
NET -- packets sent: 57, received: 69, average hop: 1.000
MAC -- packets sent: 110, received: 440
-----------------------------------------------------------------------------
CostSimEng with HeapQueue, stopped at 60.000000
 1058 events processed in 0.004 seconds, event processing rate: 264500
```

Figure 2. An example of the simulation results from SENSE simulator.

To the best of our knowledge there are no dedicated applications for guided creation of simulation scenarios over SENSE, namely an application that can capture WSN parameters, manage simulation scenarios and results presentation and management. These shortcomings clearly justify the development of G-Sense.

## IV. G-SENSE TOOL

In this section we describe the G-Sense tool in terms of requirements that led to its development, namely functional architecture and SENSE interaction. We also present G-Sense foundation technology, with the respective fundamentals.

### A. *G-Sense Requirements*

The requirements analysis phase always precedes the construction of an application and G-Sense is no exception. Beginning with the main goal to provide simulation management and creation support, we defined several requirements that G-Sense should cope with.

Functional requirements present the expected features the application must contain. In this regard, the application must offer simulation creation, presenting a graphical interface for parameter introduction for simulation environment definition, issue automatic simulation on behalf of the user, and provide visualization of simulation results. Moreover, save and load of simulation scenarios and PDF export of results are also part of the functional requirements.

Architectural requirements focus on the relationship between SENSE and G-Sense. The architecture must provide a seamless user experience, without the need for user expertise on C++ and *makefiles*. As a result SENSE must be invisible to the user.

Finally, G-Sense should also enable facilitated comparison between protocols, by a single click on a button, and take advantage of SENSE node positioning output present on some protocols. Node positioning must be graphically represented, if possible as a chart.

### B. *G-Sense Architecture*

The functional architecture of G-Sense mainly focuses on the interaction with the SENSE simulator. Figure 3 depicts G-Sense architecture block diagram. The GUI (Guided User Interface) component is responsible for simulation parameters input and validation, and also displays simulation results. File management is responsible for load and save operations, as well as PDF file creation. Finally, parameter processing sends simulation parameters to the SENSE simulation engine for processing, and graphical representation is responsible for node position representation.

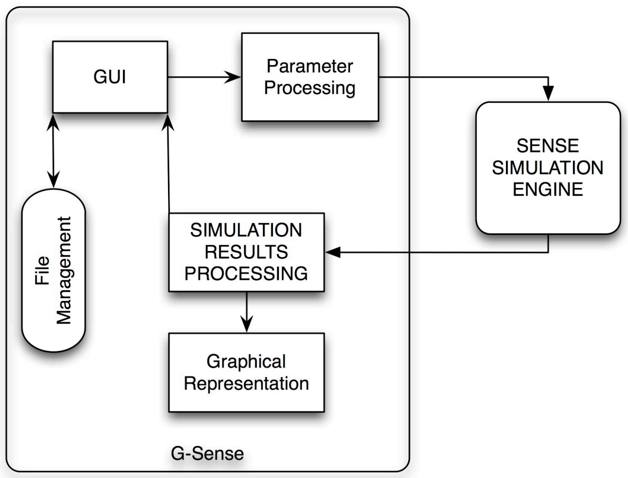


Figure 3. Block diagram of G-Sense architecture.

During the system construction, we gave special attention to the connection between G-Sense and SENSE, in order to find a suitable solution for it. SENSE uses a specific compiler, CompC++, a component-oriented extension to C++ [14]. This complicated the integration process, even using *.NET* Framework [15]. A feasible solution is to create executable files from SENSE, one for each supported protocol. Under *.NET* Framework we created a process with the executable file and pass parameters to it – the simulation parameters. This approach is fundamental for SENSE integration on G-Sense, and is possible through the (argc, argv) variables on the main function.

G-Sense allows operation with other user-created protocols on SENSE. Given the software connection nature between G-Sense and SENSE, the user must create the executable file corresponding to its new developed protocol. Such file must also use the same interface as other SENSE protocols, that is, accept the same parameters and by the same order, and output must also be in a similar form.

In terms of GUI, we considered an emerging technology for GUI specification, XUL (XML User interface Language), developed by the Mozilla project, is a XML user interface markup language [16]. However, it can only currently be used over Mozilla applications, such as Firefox, and web applications transferred over HTTP, which clearly is not the case with G-Sense.

### *C. G-Sense Technologies*

The construction of G-Sense was based on the Microsoft approach instead of Java. The *.NET* Framework supports multiple programming languages, enabling programming language freedom. The *.NET* Framework provides execution environment, simplified development and integration between several programming languages, such as C++ used by SENSE.

G-Sense is entirely built in C#, taking advantage of a powerful language on the *.NET* Framework, featuring advanced standard services that can be integrated on a variety of computer systems running Windows®.

## V. USING G-SENSE

Our tool presents a very simple yet powerful interface, which only asks for the needed parameters to perform a simulation. On the main screen, the Menu dominates the functionality. The File Menu allows load and save of simulation scenarios, and application exit. The protocols menu allows selection of the given protocol to be used, opening the respective simulation screen. Two protocol layers can be selected: the Network layer and MAC layer. Finally, the help menu presents the *about* screen and general application help.

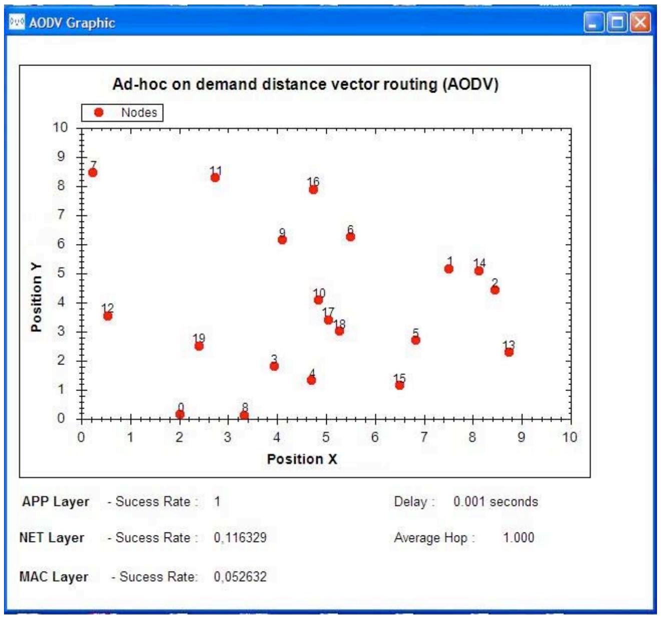
G-Sense guided user interface enables configuration of top-level parameters of the sensor network. As can be seen, from top to bottom, we have the following simulation parameters: *Stop Time* (expressed in seconds), *Number of Nodes*, *Terrain Size* (expressed in meters), *Number of Source Nodes*, *Packet Size* (expressed in Kbytes), and *Interval* (expressed in seconds).

*Stop Time* is the total simulation time, while *Number of Nodes* expresses the number of network nodes present in the WSN. The terrain is squared by default, with *Terrain Size*, defined in meters, equal to each side. *Source Nodes* are used to provide stimulus to the network, while *Packet Size* is the nominal size of transmitted packets between nodes. Finally, *Interval* defines the periodicity of packet transmission.

After simulation parameters input, the user can issue a simulation by hitting *Run* button. From this step everything goes automatically, G-Sense calls the corresponding SENSE process, and afterwards presenting simulation results when finished.

When a simulation is concluded, the application is ready for the next step. The user can save the current simulation parameters on a file with extension similar to the initials of the simulated protocol. In this case (figure 4), the file extension will have the "aodv" extension. This file can be loaded when the user wants, enabling parameters retrieval in a single operation. The user can change parameters after load, and simulate with the new parameters. The simulation parameters can also be saved onto a PDF file for easy sharing.

The user can access the respective graphic of the simulation through the *Graphic* button, where it can see the location of the nodes on the terrain. An example is depicted on Figure 4. We used ZedGraph [17], a class library, user control and web control for .NET, also written in C# to create the graphical simulation output from SENSE raw data. This enables node placement visualization inside the created terrain. ZedGraph creates a right-click context menu that enables graph copy into another application, as well as zoom in and check current values.



APP Layer - Sucess Rate : 1 Delay : 0.001 seconds

NET Layer - Sucess Rate : 0,116329 Average Hop : 1.000

MAC Layer - Sucess Rate: 0,052632

Figure 4. Graphical output of G-Sense showing node's position.

With *Compare* feature, the user can select another protocol to simulate with the same parameters. G-Sense opens another window, conveniently allowing in-screen compare of simulation results, with the ability to change simulation parameters on both windows and perform different comparisons.

An extra *Input* button allows the user to simulate an extra protocol with G-Sense, given the conditions presented in subsection III-B. In this case the saved file will have the extension of "eprot".

## VI. Tests and Validation

Every application needs testing on its development phase to validate the proposed solution. In this section we demonstrate application functionality through testing, which leads to the corresponding validation.

### A. *Application Testing*

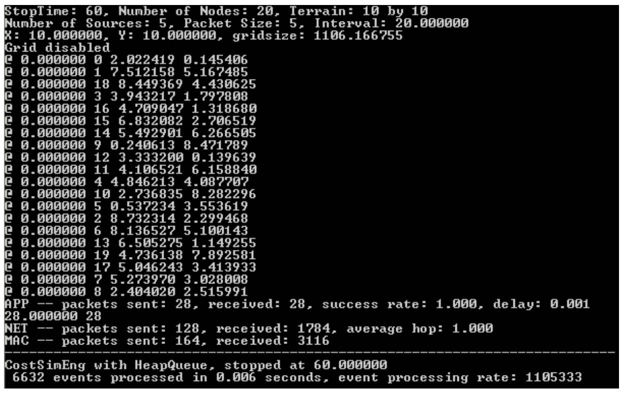
We performed several application features testing along with the development cycle. This approach enables early bug tracking and greatly contributes for application evaluation. We divided testing into two main areas, testing the features and testing the results against SENSE output with the same parameters and protocol.

Testing the features involves testing every G-Sense feature under all possible scenarios. As a result it involves parameter input validation, file operation validation (file overwrite, file load, among others), compare and node position visualization feature. All features were tested and validated.

The comparison of outputs from SENSE and G-Sense is to us more than just a testing, it involves validation of our approach. If output does not match, it does not matter whether G-Sense functionalities are bug free. As a result, the following sub-section presents an example of such results, demonstrating the difference between SENSE output and G-Sense operation in a simulation scenario.

### B. *Application Validation*

We issued several different simulation scenarios for output compare between G-Sense proposal and SENSE. In Figure 5 we show the obtained simulation output from SENSE, using AODV.



```
StopTime: 60, Number of Nodes: 20, Terrain: 10 by 10
Number of Sources: 5, Packet Size: 5, Interval: 20.000000
X: 10.000000, Y: 10.000000, gridsize: 1106.166755
Grid disabled
@ 0.000000 0 2.022419 0.145406
@ 0.000000 1 7.512158 5.167485
@ 0.000000 18 8.449369 4.430625
@ 0.000000 3 3.943217 1.797808
@ 0.000000 16 4.709047 1.318680
@ 0.000000 15 6.832082 2.706519
@ 0.000000 14 5.492901 6.266505
@ 0.000000 9 0.240613 8.471789
@ 0.000000 12 3.333200 0.139639
@ 0.000000 11 4.106521 6.158840
@ 0.000000 4 4.846213 4.087707
@ 0.000000 10 2.736835 8.282296
@ 0.000000 5 0.537234 3.553619
@ 0.000000 2 8.732314 2.299468
@ 0.000000 6 8.136527 5.100143
@ 0.000000 13 6.505275 1.149255
@ 0.000000 19 4.736138 7.892581
@ 0.000000 17 5.046243 3.413933
@ 0.000000 7 5.273970 3.028008
@ 0.000000 8 2.404020 2.515991
APP -- packets sent: 28, received: 28, success rate: 1.000, delay: 0.001
28.000000 28
NET -- packets sent: 128, received: 1784, average hop: 1.000
MAC -- packets sent: 164, received: 3116
------------------------------------------------------------------------
CostSimEng with HeapQueue, stopped at 60.000000
 6632 events processed in 0.006 seconds, event processing rate: 1105333
```

Figure 5. SENSE simulation output using SSR routing.

We used the following simulation parameters: stop time equals 60 seconds, 20 nodes, 10 by 10 terrain size, 5 source nodes, 10KBytes per packet and 5 seconds interval time.

As expected, since SENSE provides a command-line-based operation, the simulation results output is done directly over the terminal.

Figure 6 presents the same simulation, but using G-Sense. As one can confirm, the results perfectly match. We issued several of this tests to validate our approach, namely because of the adopted relationship between Sense and G-Sense.

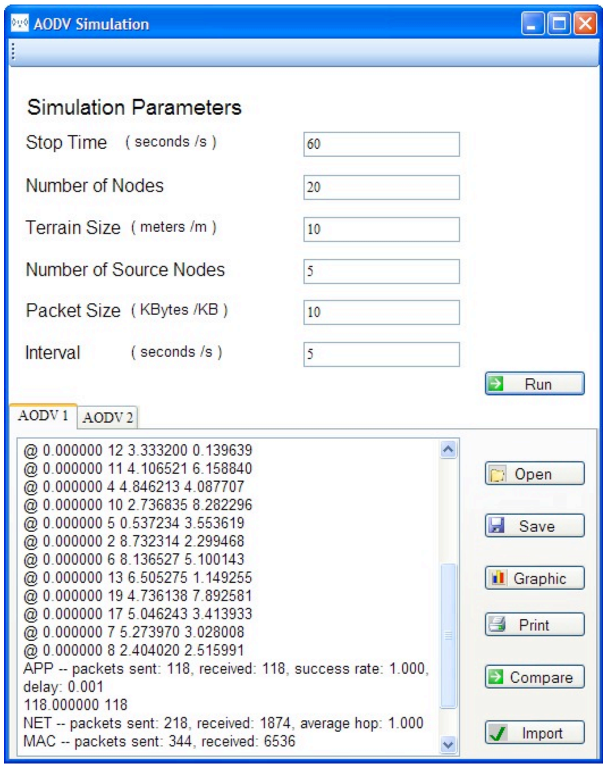
## VII. Conclusion and Future Work

This paper presented G-Sense, a user-friendly GUI tool that enables extensive simulation scenarios management and provides several enhancements over SENSE. However, G-Sense is bounded by SENSE capabilities, namely in terms of simulation results.

G-Sense abstracts the user from SENSE internals, namely, programming languages and command-line parameters. Additional features such as simulation parameters load and save on file, simulation results comparison with different protocols, and PDF output are also considered. Finally the (possible) graphical representation of nodes is also present. Such timesaving features greatly improve user efficiency, as being able to focus on the simulation and not on underlying technologies.

With this downloadable tool, available at *http://netgna.it.ubi.pt/gsense.html*, we expect to contribute for the adoption of SENSE simulator, while also offering a simple, yet powerful, guided user interface tool for wireless sensor networks simulation.

As future work we hope to introduce more flexibility on the extra protocol support, namely with permanent inclusion of external protocols onto the G-Sense environment.



Simulation Parameters

Stop Time ( seconds /s ) 60

Number of Nodes 20

Terrain Size ( meters /m ) 10

Number of Source Nodes 5

Packet Size ( KBytes /KB ) 10

Interval ( seconds /s ) 5

```
@ 0.000000 12 3.333200 0.139639
@ 0.000000 11 4.106521 6.158840
@ 0.000000 4 4.846213 4.087707
@ 0.000000 10 2.736835 8.282296
@ 0.000000 5 0.537234 3.553619
@ 0.000000 2 8.732314 2.299468
@ 0.000000 6 8.136527 5.100143
@ 0.000000 13 6.505275 1.149255
@ 0.000000 19 4.736138 7.892581
@ 0.000000 17 5.046243 3.413933
@ 0.000000 7 5.273970 3.028008
@ 0.000000 8 2.404020 2.515991
APP -- packets sent: 118, received: 118, success rate: 1.000,
delay: 0.001
118.000000 118
NET -- packets sent: 218, received: 1874, average hop: 1.000
MAC -- packets sent: 344, received: 6536
```

Figure 6. G-Sense simulation output using SSR routing.

## Acknowledgments

Part of this work has been supported by the Portuguese Science and Technology Foundation (*Fundação para a Ciência e Tecnologia*), under contract PTDC/EIA/70830/2006, the *Instituto de Telecomunicações*, Next Generation Networks and Applications (NetGNA), Portugal, and by the Euro-NF Network of Excellence from the Seventh Framework Programme of EU.

## References

[1] I. Khemapech, I. Duncan, and A. Miller, "A Survey of Wireless Sensor Networks Technology", in *6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool, UK, June, 2005.

[2] ns-2 Networking Simulator, URL: http://nsnam.isi.edu/nsnam/index.php/User_Information, Accessed in February 2009.

[3] The J-SIM Official Webpage, URL: http://sites.google.com/site/jsimofficial/, Accessed in February 2009.

[4] SENSE - Sensor Network Simulator and Emulator, URL: http://www.ita.cs.rpi.edu/sense/index.html, Accessed in January 2009.

[5] D. Chen and P. K. Varshney, "QoS Support in Wireless Sensor Networks: A Survey", in *International Conference on Wireless Networks (ICWN 2004)*, Las Vegas, Nevada, USA, June 21-24, 2004.

[6] P. Neves, J. Fonseca, and J. Rodrigues, "Simulation Tools for Wireless Sensor Networks in Medicine - A Comparative Study", in *Proceedings of the First International Conference on Biomedical Electronics and Devices, BIODEVICES 2008*, Funchal, Madeira, Portugal, January 28-31, 2008, pp. 111-114.

[7] C. Goldstein, S. Leisten, K. Stark, and A. Tickle, "Using a Network Simulation Tool to Engage Students in Active Learning Enhances their Understanding of Complex Data Communications Concepts", in *Proceedings of the 7th Australasian conference on Computing education*. vol. 42 Newcastle, New South Wales, Australia: Australian Computer Society, Inc., 2005, pp. 223-228.

[8] P. Neves, I. Veiga, and J. Rodrigues, "G-JSim - A GUI Tool for Wireless Sensor Networks Simulations Under J-SIM", in *IEEE International Symposium on Consumer Electronics (ISCE 2008)*, ISBN: 978-1-4244-2422-1, Vilamoura. Algarve, Portugal, April 14-16, 2008, pp. 1-4.

[9] NS-2 Workbench Project (nsBench), URL: http://www.mnlab.cs.depaul.edu/projects/nsbench/main.htm, Accessed in May 2009.

[10] gEditor, the J-Sim Graphical Editor, URL http://www.j-sim.org/geditor/v0.5/, Accessed in May 2009.

[11] S. Doolla, S. S. Bhat, T. S. Bhatti, and V. M., "A GUI Based Simulation of Power Electronic Converters and Reactive Power Compensators Using MATLAB/SIMULINK", in *International Conference on Power System Technology, PowerCon 2004*, November 21-24, 2004, pp. 1710-1715.

[12] G. Chen and B. K. Szymanski, "COST: A Component-Oriented Discrete Event Simulator", in *Proceedings of the Winter Simulation Conference*, ISBN: 0-7803-7614-5, San Diego, California, USA, December 8-11, 2002, pp. 776-782.

[13] C. Morrell, T. Babbitt, and B. K. Szymanski, "Visualization in Sensor Network Simulator, SENSE and Its Use in Protocol Verification", Rensselaer Computer Science 2008.

[14] CompC++: A Component-Oriented Extension to C++, URL: http://www.cs.rpi.edu/~cheng3/compc++/, Accessed in January 2009.

[15] M. A. Stoecker, S. J. Stein, and T. Northrup, *Microsoft .NET FRAMEWORK 2.0 Windows-Based Client Development*: Microsoft Press, 2007.

[16] XUL - XML User interface language by Mozilla project, URL: http://www.mozilla.org/projects/xul/, Accessed in June 2009.

[17] ZedGraph - A class library, user control and web control for .NET, URL: http://sourceforge.net/project/showfiles.php?group_id=114675, Accessed in April 2009.