

SenNetSim: A GUI-Based Simulator for Sensor Networks

Shrinivasa Kini and Srinivasan Ramasubramanian
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ 85721
{skini, srini}@ece.arizona.edu

ABSTRACT

Sensor Networks have been receiving increasing attention from the research and development communities for their wide range of applications in areas such as security, surveillance, environmental monitoring and health care. To evaluate the solution to a problem in this area, it is often infeasible to have a practical deployment of a sensor network. In such cases, simulations in the form of computer software programs are employed.

SenNetSim is a novel Graphical User Interface (GUI) based simulator which provides a framework to model a sensor network for coverage applications. SenNetSim is composed of four components - the main application window, the control panel, the sensor library, and the graph window. The main application window, shown in Figure 1, allows the user to add and remove sensor nodes and define the region to be covered. As the application is document-based, there may be several main windows open at the same time. However, the user can work only on one window at a time, the current working window is referred to as the *current document*. The control panel, shown in Figure 2 is a floating palette which displays the attributes corresponding to the objects in the current document. It has four sections that provide information on the current document, selected node(s), grid settings, and covers. A toggle button allows the user to close a particular section when it is not required. The sensor library window, shown in Figure 3, provides an interface to manage the different kinds of sensors that may be deployed at a node in the network. The simulation supports a node to be equipped with multiple sensing modalities that share a common battery. The user is allowed to add/delete sensors and edit attributes such as name, coverage type, coverage area, rate of energy consumption, etc. of a sensor.

The application currently supports only rectangular regions, but can be easily extended to support arbitrary regions. There are two modes under which the graph is constructed: *Add Node* (AN) and *Edit Info* (EI). As the names suggest, in the AN mode, the user is allowed to add nodes to the graph and in the EI mode, the user is allowed to modify the attributes of a node or a set of nodes. The user may switch between these modes at any time. Nodes can be added anywhere in the window - inside or outside the region defined. The simulator also provides the capability to deploy nodes with random coordinates in the

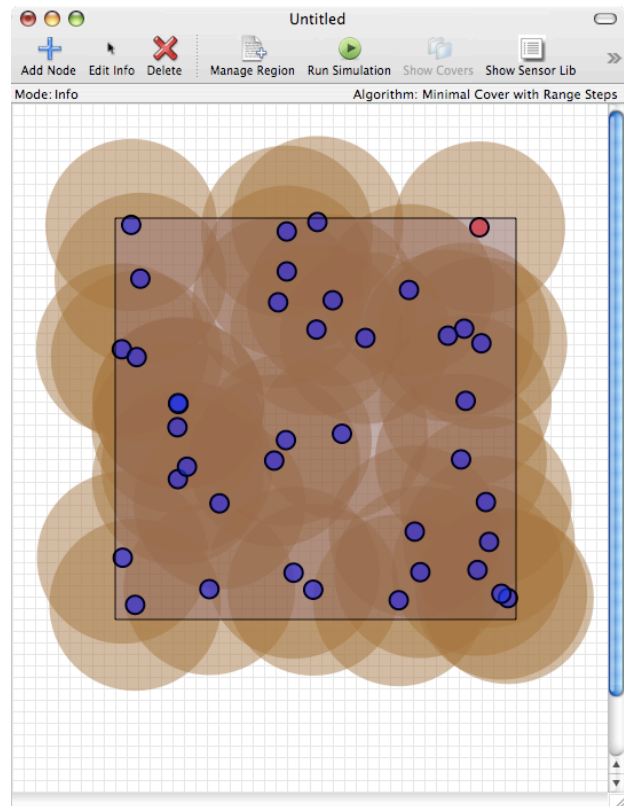


Fig. 1. The main application window.

graph window. Once the nodes are deployed, the user may select an algorithm from the application menu for simulation on the constructed graph. The controls to start a simulation and view the results are provided in the main document window.

Any new algorithm to be included into the application will have to be coded in the project and included in the build. The simulator is also capable of showing graph plots of the results from within the application. An example graph plot is shown in Figure 4. The graph plot is extended from the SM2DGraphView framework from Snowmint Creative Solutions [1]. The application allows the user to save a graph and open previously saved graphs. The entire graph is stored into one single file with a ".sns" extension with the data being stored in XML format. This allows the user to edit the graph independent of the application using a simple text editor.

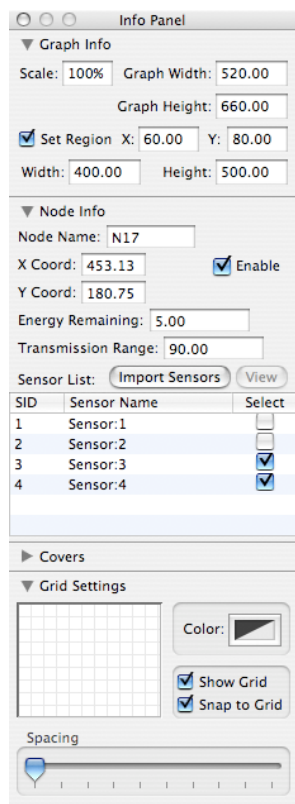


Fig. 2. The control panel.

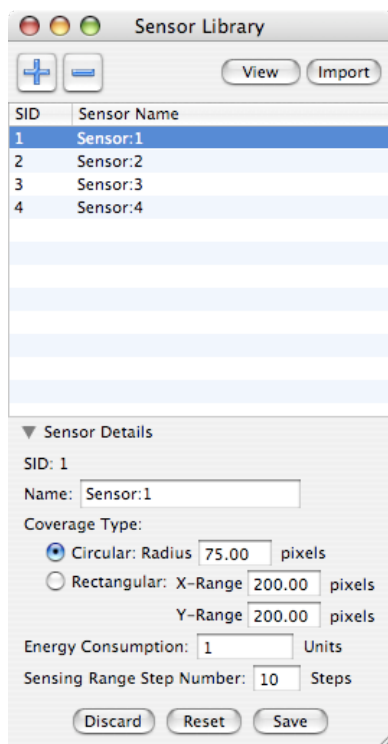


Fig. 3. The sensor library window.

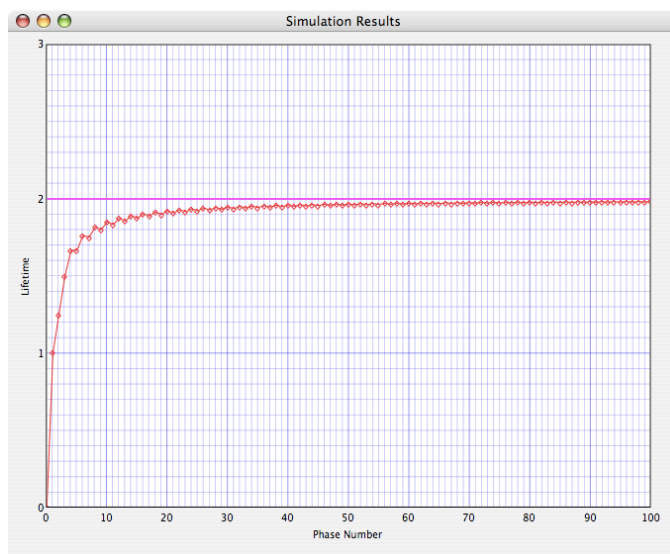


Fig. 4. The graph window.

As most algorithms select a subset of covers, the simulator provides controls for viewing the sequence of covers selected once the simulation is complete.

The simulator is developed using the Objective-C (ObjC) programming language with the Cocoa framework on the Mac OS X platform. With its grounding in ANSI C, ObjC provides the benefits of an object-oriented programming (OOP) language and support for advanced features such as dynamic typing, binding, and loading. The Cocoa framework, written in ObjC, has an event-driven architecture that provides useful and efficient set of classes needed to build the GUI components of the simulator.

ACKNOWLEDGMENT

The research developed in this paper is supported by National Science Foundation under grants CNS-0325979 and CNS-0435490.

REFERENCES

- [1] Snowmint Creative Solutions LLC, "Sm2dgraphview framework," <http://developer.snowmintcs.com/frameworks/sm2dgraphview/>, 2006.