NetTopo: A Framework of Simulation and Visualization for Wireless Sensor Networks

Lei Shu, Chun Wu, Manfred Hauswirth

***Abstract*—Network simulators are necessary for testing algorithms of large scale wireless sensor networks (WSNs), but lack the accuracy of real-world deployments. Deploying real WSN testbed provides a more realistic test environment, and allows users to get more accurate test results. However, deploying real testbed is highly constrained by the available budget when the test needs a large scale WSN environment. By leveraging the advantages of both network simulator and real testbed, an approach that integrates simulation environment and testbed can effectively solve both scalability and accuracy issues. Hence, the simulation of virtual WSN, the visualization of real testbed, and the interaction between simulated WSN and testbed emerge as three key challenges. In this paper, we present an integrated framework called NetTopo for providing both simulation and visualization functions to assist the investigation of algorithms in WSNs. NetTopo is provides a common virtual WSN for the purpose of interaction between sensor devices and simulated virtual nodes. Two case studies are described to prove the effectiveness of NetTopo.**

***Index Terms*—NetTopo, Research Framework, Wireless Testbed, Simulation, Visualization, Wireless Sensor Networks.**

I. Introduction

THE increasing market requirements of wireless sensor networks (WSNs) applications drive the fast development of research in various aspects, e.g., sensor hardware design, sensor operating system development. Among all kinds of research issues in WSNs, various algorithms, e.g., routing algorithm, topology control algorithm, are highly concentrated. Designing and validating algorithms pertaining to WSNs are among the most fundamental focuses of researchers. Network simulators are widely used for the purpose of analysis in these tasks due to the fast prototyping and the capability of tackling large scale systems. However, even the best simulators are still not able to provide real condition simulation environments in terms of completeness, complexity and accuracy, e.g., theoretical model based simulation may trade accuracy for simulation performance [1]. Taking this drawback of simulators into account, using real testbed to evaluate algorithms of WSNs is essentially necessary before applying them into commercial applications. Testbeds allow for rigorous and replicable testing. Nevertheless, there are two serious limitations on this approach in the following two conditions: 1) Large scale. Till today, it is still very expensive to buy a large number of sensor devices for a large scale testbed, e.g., the price of a Crossbow professional kit with eight sensor nodes is around 2800 Euros. Especially, for most academic researches the cost for building a large scale testbed is not acceptable. 2) Not replicable environment. For some specific applications, e.g., monitoring an erupting volcano [2], rescue in a sudden earthquake and monitoring hazardous situations [3], deploying a testbed is unwanted since the devices are exposed to dangerous conditions which can cause serious damage.

L. Shu is with the Digital Enterprise Research Institute Ireland, National University of Ireland, Galway, Ireland e-mail: lei.shu@deri.org (see http://lei.shu.deri.googlepages.com/).

C. Wu and M. Hauswirth are with the Digital Enterprise Research Institute Ireland, National University of Ireland, Galway, Ireland.

Even though simulation models are usually not able to represent the real environments with the levels of completeness and accuracy, simulators are still very important tools, because that simulators enable rapid prototyping and early evaluation of sensor network applications and algorithms. Compared to the cost and time involved in setting up an entire testbed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. Due to the complementary properties of simulators and testbeds, a better solution could be the integration of simulation environment and physical testbed. Using this technology, applications can run partially in a simulation environment and partially in a physical WSN testbed and interact with each other to create an environment where scalability issues, heterogeneous environments, etc. can be better studied.

Thus, the subject of this research work is about building a new software framework, which integrates simulation and visualization functions to assist the investigation of various algorithms in WSNs. This integration is specially motivated by the following two concrete scenarios:

- Researchers want to compare the performance of running a same algorithm in both simulator and real testbed. The comparison can guide researchers to improve the algorithm design and incorporate more realistic conditions. A good example is the applying of face routing algorithm in GPSR [4], which is proved to be loop free in theory but actually is not loop free in realist situations, due to the irregular radio coverage [5].
- A budget limitation prevents researchers from buying enough real sensor nodes but the research work has to base on a large scale WSN. For example, to evaluate the performance of sensor middleware [6], a large scale sensor network is needed. Researchers can actually do the research work by integrating a small number of real sensor nodes and a large number of virtual sensor nodes generated from the simulator.

The integration of simulation environment and physical testbed brings three major challenges:

- Sensor node simulation. Normally, a number of heterogeneous sensor devices can be used for building a WSN testbed. The integrated platform should not simulate only a specific sensor device, which means that the heterogeneous problem requires the integrated platform to be flexible enough to simulate any new sensor device.

- Testbed visualization. Sensor nodes are small in size and do not have user interfaces as displays or keyboards, which is difficult to track the testbed communication status. On the other hand, the communication topology in testbed is invisible, but researchers usually need to see the topology to analyze their algorithms. For example, when implementing a routing algorithm in the testbed, the actual routing path is expected to be visible.
- Interaction between the simulated WSN and testbed. The simulated WSN and the real testbed need to exchange information, e.g., routing packet. Their horizontal interconnection, communication, interaction, and collaboration are all emerging difficult problems that need to be addressed.

In this paper, we present an extensible integrated framework of simulation and visualization called NetTopo to assist investigation of algorithms in WSNs. With respect to the simulation module, users can easily define a large number of on-demand initial parameters for sensor nodes, e.g., residential energy, transmission bandwidth, radio radius. Users also can define and extend the internal processing behaviour of sensor nodes, e.g., energy consumption, bandwidth management. NetTopo allows users to simulate an extremely large scale heterogeneous WSN. For the visualization module, it works as a plug-in component to visualize testbed's connection status, topology, sensed data, etc. These two modules paint the virtual sensor nodes and links on the same canvas which is an integration point for centralized visualization. Since the node attributes and internal operations are user definable, it guarantees the simulated virtual nodes to have the same properties with those of real nodes. The sensed data captured from the real sensor nodes can drive the simulation in a pre-deployed virtual WSN. Topology layouts and algorithms of virtual WSN are customizable and work as user defined plug-ins, both of which can easily match the corresponding topology and algorithms of real WSN testbed. As a major contribution of this research work, NetTopo is released as open source software on the SourceForge. Currently, it has more than eighty java classes and 11,000 Java lines source codes. Users can freely download the latest version of NetTopo by accessing the NetTopo website [7].

The rest of the paper is organized as follows: Section 2 presents an overview of the related work in terms of simulators and WSN visualization and positions our work with respect to the related literature. Section 3 illustrates design issues of the framework including architecture, mechanism of modular components, module interfaces and interaction between these interfaces. Section 4 describes the features of NetTopo and analyzes the implementation of the framework. It is focused on logic, programming techniques and interesting code of some core features. Section 5 presents two case studies provided in NetTopo as examples for testing the effectiveness of the framework. Section 6 concludes the thesis and prospects the future work.

II. Related Work

Since NetTopo is an integrated framework of simulation and visualization for WSNs, network simulators and WSN visualization are two main aspects of related work. In this section, we have an overview of literature on both of them respectively.

A. Related Work on WSNs Simulators

So far, a large number of WSN simulators have been proposed by researchers. These simulators can be classified into three major categories based on the level of complexity:

- *Algorithm level.* Simulators [8-10] focus on the logic, data structure and presentation of algorithms. These simulators do not consider detailed communication models and, most commonly, they rely on some form of a graph data structure to illustrate the communication between nodes. AlgoSensim [8] analyzes specific algorithms in WSNs, e.g., localization, distributed routing, flooding. Shawn [9] is targeted to simulate the effect caused by a phenomenon, improve scalability and support free choice of the implementation model. Sinalgo [10] offers a message passing view of the network, which captures well the view of actual network devices. It was designed, but is not limited to simulate wireless networks. These approaches do not reproduce the ISO network stack and with no or very simplistic MAC layer protocol, they can simulate extremely large networks.
- *Packet level.* Simulators [11-14] implement the data link and physical layers in a typical OSI network stack. Hence, it is common for these type of simulators to find implementations of 802.11b or newer MAC protocols and radio models that account for propagation, fading, collision, noise and wave diffraction. The most popular widely used simulator is ns-2 [11] which is not originally targeted to WSNs but IP networks. SensorSim [12] is an extension to ns-2. It provides battery, radio propagation and sensor channel models. J-Sim [13] adopts loosely-coupled, component-based programming model, and it supports real-time process-driven simulation. GloMoSim [14] is designed for the parallel discrete event simulation capability provided by PARSEC.
- *Instruction level.* Simulators [15-17] model the CPU execution at the level of instructions or even cycles. They are often regarded as emulators. TOSSIM [15] is the most representative but not most accurate of the existing emulators. TOSSIM simulates the TinyOS network stack at the bit level. The communication between nodes is modelled through a probabilistic graph which, as side-effect, may reduce the overall accuracy of this tool. Atemu [16] is an emulator that can run nodes with distinct applications at the same time. For this reason, Atemu is more accurate but its scalability is limited. Avrova [17] is a Java-based emulator used for programs written for the AVR microcontroller produced by Atmel and the Mica2 sensor nodes. Being written in Java, it also benefits from the added portability across various computing platforms.

All these simulators make great contributions to the development of network simulation. They have their own advantages and limitations. Based on the above classification, the simulation framework in our NetTopo is by far algorithm oriented. Additionally, it is clear that none of these simulators has considered integrating with real WSN testbed. This point clearly distinguishes NetTopo from them.

B. Related Work on WSNs Visualization

With respect to visualization of real WSN testbed, there is much less related work. Octopus [18] is an open-source dashboard to visualize and to control a WSN in the TinyOS 2.x environment. Octopus provides users with a graphical user interface (GUI) for viewing the live sensor network topology. It also allows the user to dynamically control the behavior of sensor nodes, e.g., the energy consumption, the sampling period, the radio duty cycle and formulating application queries to the nodes. The Surge Network Viewer [19] and the Mote-VIEW [20] are products of Crossbow Company to visualize WSNs. They are capable of logging wireless sensor data to a database and to analyze and plot sensor readings. The Surge Network Viewer features topology and network statistics visualization as well as logging of sensor readings and the viewing of the logged data. The Mote-VIEW, working as monitoring software, covers essentially the same topics but presents a much cleaner user interface and more features. They are designed to support only Crossbow sensor nodes, thus they are not extensible. SpyGlass [21] visualizes WSN using a flexible multi-layer mechanism that renders the information on a canvas. Data emitted by individual sensor nodes is collected by gateway software running on a machine in the sensor network. It is then passed on via TCP/IP to the visualization software on a potentially remote machine. Visualization plug-ins can register to different data types and visualize the information on a canvas. TinyViz [22] is a GUI tool of TOSSIM package of TinyOS. It visualizes sensor readings, LED states and radio links and allows direct interaction with running TOSSIM simulations. Using a simple plug-in model, users can develop new visualizations and interfaces for TinyViz. It also allows users to interact with a simulation through a GUI, but these interactions are often ad-hoc, as well as laborious and difficult to reproduce.

In short, most of existing visualization tools support only a single type of WSN and are highly coupled to the TinyOS. However, NetTopo tries to target at the visualization and control of WSN testbed where heterogeneous devices are used, e.g., wireless camera, Bluetooth based body monitoring sensor devices, and these devices are generally not TinyOS based.

III. Design of the Framework

This section illustrates in detail the process of problem-solving and planning for the whole framework solution. We first analyze the requirement specification. Then we describe the architecture of NetTopo. Based on the architecture, modules are partitioned and presented in a hierarchy with specific description respectively. After that, we present the design of module interfaces and data persistence. For the purpose of utilizing NetTopo software and extending the framework for future customizable use, this section provides related high-level software design approaches.

A. Requirement Specification

This subsection provides specific description of the behaviour of the framework to be developed. Related use cases are also included. The specification is divided into functional requirements and non-functional requirements based on project motivation, preliminaries of related work and using experience of researchers in Digital Enterprise Research Institute (DERI), National University of Ireland, Galway.

Functional requirements are listed as follows:

- The system shall provide basic symbols representing elements in a WSN including source node, sensor node, sink node, hole and link. All these symbols can be used to build users' own expected network.
- The system shall be extensible to allow users to define expected attributes of their own virtual sensor nodes.
- The system shall allow users to deploy virtual sensor nodes and holes in pre-defined network topologies including line, circle, tree, grid and random or user-defined topologies.
- The system shall allow users to kill virtual sensor nodes to make them as holes.
- The system shall provide a set of high-level APIs for users to simulate their own algorithms.
- The system shall provide functions for debugging and logging simulation process.
- The system shall be extensible to provide device-based wrappers for WSN visualization.
- The system shall allow the interaction and communication between simulation and visualization of WSNs.
- The system shall provide a graphical user interface on which network deployment, simulation and visualization are displayed.
- The system shall allow users to clear up all deployed virtual sensor nodes.
- The system shall allow users to create their own network with maximum freedom, e.g., adding, deleting, selecting, modifying, moving, searching, distributing sensor nodes and configuring their attributes, such as energy, expected lifetime, transmission radius.
- The system shall be able to record simulation results.
- The system shall support for sensor data streams (XML streams) import and export.
- The system shall provide file-related operations including file saving, creating and opening to save the deployment state for future reuse.
- The system shall support WSNs 3D representation.

Non functional requirements include:

- The system shall be programmed in Java so as to integrate with a java-based WSN middleware called Global Sensor Networks (GSN) [23], which is developed under the collaboration between DERI, Galway and EPFL, Lausanne.
- The system shall have a local operating system look and feel.