Making Wireless Sensor Network Simulators Cooperate

Qian Li
Swedish Institute of Computer Science
qian@sics.se

Fredrik Österlind
Swedish Institute of Computer Science
fros@sics.se

Thiemo Voigt
Swedish Institute of Computer Science
thiemo@sics.se

Stefan Fischer
Institute of Telematics
University of Lübeck
fischer@itm.uni-luebeck.de

Dennis Pfisterer
Institute of Telematics
University of Lübeck
pfisterer@itm.uni-luebeck.de

ABSTRACT

The development and testing of wireless sensor networks is a cumbersome and tedious task. Therefore, a number of simulators, testbeds and other tools have been developed. Unfortunately, these tools are not able to cooperate since they have not been designed with cooperation in mind. In this paper, we present an approach to make simulation tools cooperate that is based on a common input and output format. Using the recently developed WiseML format, we show that our approach is a viable solution for making simulators cooperate.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: Miscellaneous

General Terms

Performance, Measurement

Keywords

Simulators, Cooperation, Wireless Sensor Networks

1. INTRODUCTION

Simulators are indispensable tools to support the development and testing of wireless sensor networks. Simulations are commonly used for rapid prototyping which is otherwise very difficult due the restricted interaction possibilities with this type of embedded systems. Simulators are also used for the evaluation of new network protocols and algorithms and enable repeatability because they are independent of the physical world and its impact on the objects. Moreover, simulations enable nonintrusive debugging at the desired level of detail.

There exists a large number of wireless network simulators for different purposes. Some simulators are mostly used for the development and testing of algorithms. These simulators have different foci. Shawn's [10] focus is simulation speed, whereas Nettopo's [7] focus is on routing and visualization. Other simulators target prototyping of application and system software development and evaluation. This type of simulators are usually operating system specific and simulate deployable code. TOSSIM simulates TinyOS code [11] and COOJA was originally developed to simulate networks of Contiki nodes [12]. There are also emulators for specific microprocessors and sensor boards such as Avrora [14] and MSPSim [5]. Another type of simulators are what we call environment simulators. These simulators simulate the environment where sensor networks are deployed and include for example fire simulators [9] or simulators that simulate mobility patterns such as BonnMotion [1].

For successful WSN development cooperation not only between testbeds and simulators but also between simulators is required. For example, to evaluate the performance of a routing algorithm in a forest fire scenario, we would like to use both a fire simulator and a simulator that simulates the application and the routing protocol. However, simulators are usually not designed with cooperation in mind. Furthermore, they are difficult to combine since they are often written in different programming languages and use different formats for configuration and output.

In previous work, we have discussed several approaches to make wireless sensor network simulators cooperate [15]. In this paper, we detail this discussion and report on the evaluation of an approach for cooperation that we call common input and output format. We have implemented this approach by supporting the WiseML format specified by the WISEBED project [2] in the COOJA simulator and in the Shawn simulator [10]. Our experience from the implementation of WiseML support in COOJA shows that extending existing simulators to support the proposed approach is straightforward. In addition, our experimental results demonstrate that this approach indeed is a viable solution to make simulators cooperate.

There have been other approaches to combine different simulators or simulators and testbed. E.g., Sridharan et al.[13] combined a structural simulator with TOSSIM. FireSenseTB [9] integrates a fire simulator with a WSN testbed. Those approaches are similar to ours but are not based on a standard such as WiseML.

2. APPROACHES TO MAKE WSN SIMULATORS COOPERATE

To ease the development of non-trivial WSNs applications, existing WSN tools must cooperate. We envision a scenario where all existing WSN simulators, testbeds, and environment simulators can exchange information easily with little time and space overhead. To achieve this goal, effective methods must be investigated to integrate existing WSN tools.

According to the tightness of the integration, integration approaches can be classified into **tight integration** and **loose integration**. Tight integration refers to joining existing simulators' source codes tightly and locally to form a new, combined simulator. Successful examples are COOJA [12] that incorporates the emulators MSPsim [5] and Avrora [14] and MiXiM, an integration of four simulators [6]. However, tight integration can only be applied to simulators written in the same programming language, which limits the number and type of simulators that can be integrated. Moreover, this type of integration is a time-consuming effort [15].

Loose integration employs a medium to bridge the differences among different WSN tools. The most common medium is a common configuration file supported by each WSN tool. Another form of loose integration is to combine WSN tools over the Internet using e.g. sockets or Web services. Using configuration files, there are basically two approaches. The first is to use a universal format that is understood by all WSN simulators. Abdolrazaghi [4] explored this approach. We have, however, earlier argued that this is not feasible because the number of configuration parameters to be included is numerous [15]. Another reason is that simulators are specific, and in some simulators, it is not possible to change parameters that are easy to change in other simulators. For instance, in MSPSim [5] – an instruction-level emulator for Tmote Sky mote types – it is not possible to run the CPU at a speed of 100 MHz since MSP430 processors do not run at this speed. Instead, we have argued that using common input scenarios and output formats is useful [15]. In the next section, we discuss this approach in more detail.

Furthermore, we can distinguish between **online** and **offline** integration. The latter method takes output from one tool and uses it as input for the other tool. This way, at most one simulator or testbed is executed at a time. A simulator or testbed must wait until the whole data set required is generated by another tool. The online approach allows multiple WSN tools to run simultaneously and to consume the data generated by other tools while this data is being produced.

3. THE COMMON INPUT AND OUTPUT FORMAT APPROACH

In this section, we briefly present the common input and output format approach.

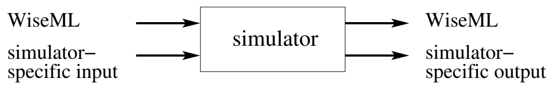

Figure 1: The common input and output approach

Figure 1 presents an high-level overview demonstrating the fact that we have some simulator-specific input and output as well as some common input and output.

In earlier work we have discussed several simulation environment parameters and simulation results that can be specified in a common format for several simulators [15]. These include topology descriptions, node movement, sensor readings, communication models, radio noise, and node failures. Obviously, the description of node movement can be regarded as an extension of the topology description. Also radio noise is a specific form of sensor reading. Regarding the example of the fire scenario, the sensor readings, the number of failed (destroyed by fire) nodes etc. provided by an environment simulator are useful input for a WSN simulation with focus on routing.

There are a number of interesting output parameters that would make simulation results more comparable include energy consumption, number of transmitted or received packets and channel utilization over both time and space. Based on these and other statistics interesting information can be inferred, e.g. preferred routing paths. Another important issue is that by having common output formats, the same tools can be used to process the simulation results.

4. WISEML: A FORMAT FOR COMMON INPUT AND OUTPUT

A format for common input and output should be easy to use, flexible and extensible, it should be supported by major programming languages and platform independent. **WiseML** [3] WIreless SEnsor network Markup Language) meets all these requirements. WiseML is an XML (eXtensible Markup Language) based language confined by WiseML schema [3] that inherits the properties of XML.

A configuration file written in WiseML is called a WiseML file. The WiseML schema defines that a WiseML file should contain both input and output data common to most of the existing WSN simulators. A WiseML file usually consists of three sections: the setup section (input), the scenario section (input), and the trace section (output). The setup section contains the static information relating to the simulation as a whole (e.g., the duration of a simulation, the coordinate type of a WSN), the node properties (e.g., position, node type), as well as link properties (e.g., encrypted, rssi). The scenario section specifies the dynamic changes of the nodes' and links' properties during the simulation. It describes, for example, at which time a node or a link fails and at which time a node moves. Unlike the first two sections, the trace section is used to store simulation results. It records e.g. sensor readings and topology changes in chronological order.

4.1 WiseML Support for the COOJA Simulator

To add WiseML support for COOJA, we divide COOJA's native CSC configuration file into a WiseML file and a COOJA file. The COOJA file contains COOJA specific information such as plugins and interfaces. The WiseML file stores the common input and output data, for example, node types and node positions. The implementation of WisemlCooja comprises three modules: the loading model that converts the WiseML format into COOJA's native format, the saving model that converts COOJA's native format into WiseML format, and the scenario generation model that converts COOJA's log information into WiseML.

5. EVALUATION

In this section we show cooperation between tools that is enabled by WiseML as common input and output format.

5.1 WSNGE and COOJA

WSNGE is a flexible and extensible WSN simulator focusing on user-friendliness, multiple protocols simulation, and online simulation reconfiguration. WSNGE released its support to WiseML 2.0 and posted an exported setup-only configuration file on its website [8].

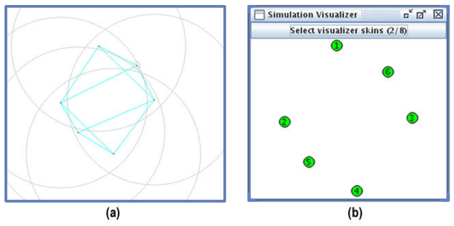

Figure 2: WiseML defined network topology being loaded into WSNGE (a) and COOJA (b)

The WiseML file contains 6 sensor nodes interconnected by 17 links. Figure 2 (a) (copied from WSNGE's website) illustrates the network's topology generated by WSNGE, (b) is the screenshot of COOJA after it successfully loads the WSNGE generated WiseML file. As we can see, the nodes are in the same positions. After assigning each sensor node a node type and an application, we can start a COOJA simulation. If we want to reload the simulation in the future, we can save it as either in COOJA's native configuration format or in the WiseML format.

5.2 BonnMotion, COOJA and Shawn

In this section, we demonstrate how Shawn loads a WiseML file exported by COOJA. This WiseML file contains node movements originally created via BonnMotion and via WiseML loaded into COOJA. BonnMotion [1] is an open-source Java software that creates and analyzes mobility scenarios. Here we have used BonnMotion to generate a mobility pattern that follows a Random-Waypoint model.

Figure 3 presents the screenshots captured from both shawn and COOJA when they are simulating with the same configuration file generated by COOJA. This configuration file contains the node movement scenario generated by BonnMotion. The network topologies at simulated second 0, 30, and 60 are shown in Figure 3 (a), (b), and (c), respectively. Since Shawn and COOJA use different coordinates to visualize network topologies, the screenshots generated by Shawn are flipped vertically for ease of comparison.

5.3 Testbed output to enhance simulations

In this section, we demonstrate how output information from a testbed can be used as input to a simulator and this way make simulations more realistic.

The first testbed consists of one sink node and three sensing nodes that sense the temperature every second. The values are logged and converted to a WiseML format temperature scenario after the completion of the experiment.

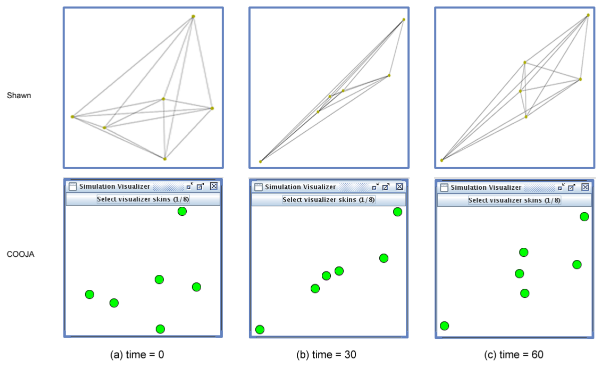

Figure 3: Network topology changes over time for Shawn (top) and COOJA (bottom). The network topology at simulated second 0 (a), after 30 s (b) and 60 s(c).

The sensing nodes broadcast packets at a rate that depends on the current temperature. If nodes sense a temperature $\leq$ 25 degrees Celcius, they transmit two packets per second and if the temperature is between 25 and 27 degrees nodes transmit four packets per second. Otherwise, they transmit eight packets per second. The nodes transmit packets at the rate until the second has past and new temperature values are sensed. The sink counts the packets it receives every second.

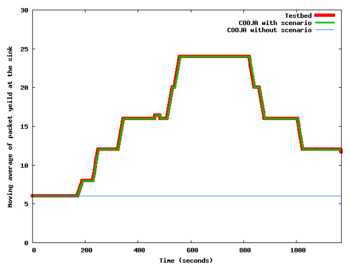

Figure 4: Output from a testbed makes simulations more realistic

Figure 4 depicts the average number of packets per second received by the sink node over time. Each value in the graph is obtained by averaging over 20 packet yields. The straight line at the bottom of the graph is generated by COOJA without temperature scenario. Here, COOJA's low default temperature that is taken and hence nodes transmit packets at the lowest rate. As a result, the sink always receives the same number of packets per second and the graph shows a straight line. The testbed curve and the COOJA with temperature scenario curve almost always overlap which shows that by supporting WiseML, testbeds can gather real life scenarios for WSN simulators to realize more realistic simulations. The setup only WiseML file corresponding to this experiment is 2.4 KB, the testbed produced temperature scenario is 8.1 KB.

A typical use-case for simulations is optimizing applications before they are deployed on hardware. Hence, a frequent task is to model a real-world environment. Apart from topology, this includes communication aspects, i.e., choosing the best communication model and parameters such that it provides a good approximation of real data transmissions. Using Shawn, a frequently used model is the so-called *stochastic communication model* which honors the fact that the probability of a successful reception diminishes with increasing distance. It defines two distances r_1 and r_2 with $r_1 < r_2$. For $0 < d < r_1$ a constant probability p_{max} is assumed while for $d > r_2$ no communication is possible. For $r_1 \leq d \leq r_2$, the packet reception probability decreases linearly from p_{max} to 0. We have derived parameterizations for this model from a set of measurements in different scenarios (e.g., indoor, outdoor, different antennae and heights above the ground).

While this model already improves the realism of Shawn simulations, it is still a substantial abstraction from reality. The real packet delivery ratio over distance may vary for each pair of nodes depending on the actual environment (e.g., walls that impede communication or different heights above ground). To better approximate reality for a given deployment, measurements from this deployment could be used to parameterize a communication model for each pair of nodes individually. Therefore, we implemented a communication model for Shawn which accepts packet delivery probabilities for each link between two nodes. To parameterize this model, we measured the packet delivery ratio for each individual link in a testbed of 21 iSense sensor nodes and logged the data to a WiseML file. From this file, packet delivery probabilities for each unidirectional link are calculated and written to a Shawn configuration file.

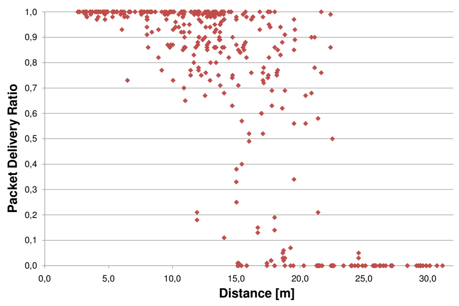

Figure 5: Packet delivery ratio measured a testbed of 21 nodes stored in WiseML and visualized

Figure 5 visualizes the measurements from the WiseML file for the 21 nodes. It is obvious that the stochastic model can only provide a very rough approximation of the prevailing packet delivery ratios in the real testbed - which is especially true in indoor deployments. Using WiseML traces to better parameterize models of simulators is hence an important means to improve the realism of simulations and to adapt the simulator to the ambient conditions of a real deployment. Obviously, this approach is not limited to Shawn but can easily be extended to other simulators.

5.4 Micro-benchmarks

We have measured the code size of the WisemlCooja implementation as the size of an application reveals, to some extent, the complexity of a system. WisemlCooja consists of seven classes thereof five Graphical User Interface handlers. The size of WisemlCooja is 137.6 KB with the main class WisemlProcessor.java 82.9 KB, accounting for approximately 62 percent of the total size. We have also measured the times it takes to load, generate and save scenarios as well as exporting scenarios to and from COOJA on a standard PC. Most of these operations are quite fast. For example, it takes around three seconds to generate a scenario for 800 simulated sensor nodes. In summary, our results show that the time and space complexity is low and would not prevent users from using our implementation.

6. CONCLUSIONS

For successful WSN development cooperation not only between testbeds and simulators but also between simulators is desirable. We have presented an approach that we call common input and output format. We have exemplified this approach with the WiseML format implemented for the COOJA and Shawn simulators. Our experiments demonstrate that this approach indeed is suitable for making simulators cooperate.

Acknowledgments This work was partially supported by the European Commission under contract numbers IST-2008-224460 (WISEBED) and FP7-2007-2-224053 (CONET).

7. REFERENCES

[1] Bonnmotion. Manual. 2009.
[2] Wisebed - wireless sensor network testbeds. `http://www.wisebed.eu`.
[3] Deliverable D4.1: First set of well-designed simulations, experiments and possible benchmarks. Technical report, The WISEBED project group, 2009. `http://www.wisebed.eu`.
[4] A. Abdolrazaghi. Unifying wireless sensor network simulators. Master's thesis, KTH Stockholm, 2009.
[5] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. Mspsim – an extensible simulator for MSP430-equipped sensor boards. In *EWSN 2007, Poster/Demo session*, Delft, The Netherlands, Jan. 2007.
[6] A. K. et al. Simulating Wireless and Mobile Networks in OMNeT++: The MiXiM Vision. In *International OMNeT++ Developers Workshop, Marseille, France*, Mar. 2008.
[7] L. S. et al. NetTopo: Beyond simulator and visualizer for wireless sensor networks. In *FGCN 2008*, Hainan, China, Dec. 2008.
[8] M. Karagiannis, I. Chatzigiannakis, and J. Rolim. WSNGE: A platform for simulating complex wireless sensor networks supporting rich network visualization and online interactivity. In *Spring Simulation Multiconference*, 2009.
[9] B. Kosucu, K. Irgan, G. Kucuk, and S. Baydere. FireSenseTB: A wireless sensor networks testbed for forest fire detection. In *IWCMC*, 2009.
[10] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. In *DASD'05*, 2005.
[11] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire tinyos applications. In *ACM SenSys*, 2003.
[12] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *IEEE SenseApp*, Tampa, USA, Nov. 2006.
[13] A. Sridharan, M. Zuniga, and B. Krishnamachari. Integrating environment simulators with network simulators. Technical report, Department of Electrical Engineering Systems, Univ. of Southern California, 2004.
[14] B. Titzer, D. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *ACM/IEEE IPSN*, Apr. 2005.
[15] T. Voigt, J. Eriksson, F. Österlind, R. Sauter, N. Aschenbruck, P. J. Marrón, V. Reynolds, L. Shu, O. Visser, A. Koubaa, and A. Köpke. Towards comparable simulations of cooperating objects and wireless sensor networks. In *WSNPerf*, 2009.