Energy-Aware Simulation for Wireless Sensor Networks

Geoff V. Merrett, Neil M. White, Nick R. Harris and Bashir M. Al-Hashimi
Electronic Systems and Devices Group, School of Electronics and Computer Science
University of Southampton
Southampton, UK
Email: gvm@ecs.soton.ac.uk

Abstract—**Energy-aware sensor nodes are usually tightly energy-constrained, execute energy-efficient algorithms, have the ability to interrogate and control the devices used for storing and consuming energy, and often feature one or more sources of energy harvesting. Due to the cost, time and expertise required to deploy a Wireless Sensor Network (WSN), simulation is currently the most widely adopted evaluation method. Network simulation is well established for mobile ad hoc networks, using simulators such as the popular ns2. However, the differing characteristics and performance criteria of WSNs introduce additional simulation requirements, and this has resulted in a number of simulators and simulator extensions developed specifically for this purpose. This paper investigates the suitability of a number of state-of-the-art simulators for evaluating energy-aware WSNs, and subsequently proposes a novel structure for simulating energy-aware WSNs. The proposed structure provides diverse, flexible and extensible hardware and environment models, and integrates a structured architecture for embedded software to enhance the design of energy-aware sensor nodes. To illustrate an implementation of the structure, details of – and observations obtained using – an in-house simulator (WSNsim) are presented.**

Keywords-wireless sensor networks, energy-aware, simulation, modeling

I. Introduction

Wireless Sensor Networks (WSNs) consist of a number of small, self-powered sensing devices (nodes) that communicate with each other using a wireless medium [3]. WSNs have application to a wide range of areas including environmental monitoring, industrial monitoring, military, healthcare, and security [4]. As nodes are often expected to operate over periods of many years, be small in volume and be self-powered, considerable research is being undertaken into the development of energy-aware hardware, algorithms and protocols.

There are three accepted techniques for evaluating and analyzing networks: analytical methods [5], computer simulation, and practical implementation. The constraints and complexity of WSNs often cause analytical methods to be unsuitable or inaccurate [6]. Additionally, the proportion of algorithms that are analyzed through practical evaluation is comparatively low, possibly due to the relative infancy, deployment cost, time required, broad diversity and application dependence of WSNs. As a result, simulation is currently the most widely adopted method of analyzing WSNs, allowing the rapid evaluation, optimization and adjustment of proposed algorithms and protocols.

This paper investigates the suitability of current network simulators to evaluate energy-aware WSNs. The findings of a thorough literature review highlight that there are a number of shortcomings in currently available simulators and, to rectify these, we propose a new simulation structure for energy-aware WSNs. The proposed structure offers the following novel features and benefits:

- The provision and flexible integration of adequate modeling for node hardware and the environment.
- The integration of a structured architecture for embedded software to enhance and consider the importance of energy-aware sensor node design.

In order to understand the requirements of a simulator, Section II provides an overview of energy-aware WSNs. Section III presents a review of existing WSN simulators, and their ability to sufficiently evaluate energy-aware WSNs. Section IV details our proposed structure for energy-aware WSN simulation and, subsequently, Section V describes how it was implemented in an in-house simulator developed as a part of this research. Finally, Section VI draws conclusions, and outlines the future directions of this research.

II. Energy-Aware Wireless Sensor Networks

Nodes in a WSN are usually highly energy-constrained and expected to operate for long periods from limited on-board energy reserves. To permit this, nodes – and the embedded software that they execute – must have energy-aware operation. Energy efficiency has been of significant importance since WSNs were first conceived but, as certain applications have emerged and evolved [7], a real need for ultra-miniaturized long-life devices has reemerged as a dominant requirement. Because of this, continued developments in energy-efficient operation are paramount, requiring major advances to be made in energy hardware, power management circuitry and energy-aware algorithms and protocols.

The energy components of a typical wireless sensor node are shown in Fig. 1. Energy is provided to the node from an *energy source*, whether this is a form of energy harvesting from sources such as solar, vibration or wind, or a resource such as the mains supply or the manual provision and replacement of primary batteries. Energy obtained from the energy source is

This work was supported in part by the Engineering and Physical Science Research Council (EPSRC) under grant number EP/D042917/1.

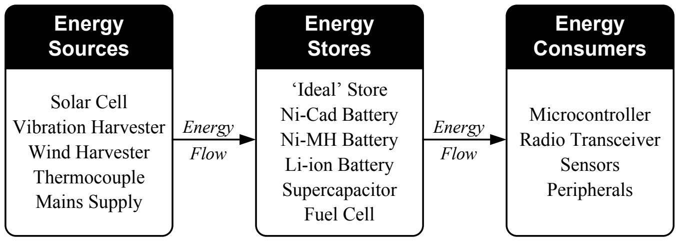

Figure 1: Energy components of a typical wireless sensor node.

buffered in an *energy store*; this is usually a battery or supercapacitor. Finally, energy is used by the node's *energy consumers*; these are hardware components such as the microcontroller, radio transceiver, sensors and peripherals.

With the increased usage of energy sources in nodes [8, 9], the need for energy stores other than batteries (many of which suffer from only offering a limited number of charging cycles) is increased. This can be seen by the body of research that is now utilizing supercapacitors (devices that are similar to standard electrolytic capacitors, but with capacities of many Farads) to store the node's energy [9, 10].

To be energy-aware, the embedded software executing on the node must be aware of the state of its energy components. This may be as advanced as monitoring the energy harvested from each source [11], inspecting the rate of consumption by different consumers [12], directing the flow of energy from and to different stores [10] and managing the charging of rechargeable stores [10]. Alternatively, this may equate to simply being able to inspect the residual energy in a single store. Therefore, the embedded software must not only be capable of interfacing with energy hardware (this is generally a requirement of power management circuitry), but also interpreting the data that are obtained – usually in the form of a sampled voltage – into a remaining lifetime, power or energy. Based upon these values, the operation of the node is adjusted accordingly, usually to maximize the lifetime of the network.

To sufficiently and effectively evaluate algorithms and protocols for energy-aware WSNs, the characteristics, properties and requirements described in this section need to be reflected and supported in simulation. The following section investigates popular WSN simulators, and assesses their ability to evaluate energy-aware WSNs.

III. Existing Simulators for Energy-Aware WSNs

While simulation is reasonably well established for Mobile Ad Hoc Networks (MANETs), the simulation of WSNs not only requires the implementation of a radio channel, but also a 'sense-able' physical environment and accurate energy models. The design aims and strategies of different simulators result in them each having different strengths and weaknesses. In order to successfully simulate and evaluate energy-aware WSNs, it is necessary to provide necessary support for the node's embedded software, wireless communication, sensing, and energy resources [13] (this is not an exhaustive list, but is the criteria under which simulators will be evaluated in this paper). This section reviews the state-of-the-art in network simulators suited to WSNs [14], and evaluates their ability to simulate energy-aware algorithms and protocols.

Simulators for use with WSNs can be classified into two predominant categories: those that have been developed as extensions to existing generic network simulators (such as the SensorSim [2] extension to ns2 [15]), and those that have been designed specifically for the simulation of WSNs (such as TOSSIM [16]).

A. *Generic Network Simulators*

The widely used ns-2 [15] is an object-oriented discrete event network simulator based upon the Real network simulator (ns) that was released in 1989. The extensibility of ns-2 has been a major contributor to its success, with protocol implementations being widely produced and developed by the research community. The use of ns-2 with WSNs is, however, limited by its scalability, the lack of an application model, and the reported difficulty in developing extensions [6]. While the latter is not such a problem for traditional networks using established protocols, it poses a number of obstacles in the simulation of WSNs. SensorSim [2] is a WSN extension for ns2, providing advanced models and the ability to interact with external hardware. As shown in Fig. 2, SensorSim's simulated embedded software contains a communication stack and a 'sensor protocol' stack. The latter samples the sensor channel (similar to a communication channel, in that it dictates spatial attenuation of physical phenomena) and computes relevant data (such as tolerances) for use by the application. SensorSim also implements a single energy provider (a battery), and multiple energy consumers (including the CPU, radio transceiver and sensors). SensorSim suffers from the same scalability problems as ns-2 and, due to support concerns, has been withdrawn from public release. More recently, the NRL Sensor Simulator [17] has been developed to provide support for WSNs in ns-2, but does not provide the level of detail for sensing and energy hardware that is presented by alternative simulators.

Like ns-2, OMNeT++ [18] is a discrete-event general purpose network simulator, using a modular structure to define the network architecture. While the well-supported INET and mobility frameworks provide support for wireless communication and mobility, a lack of sufficient sensing and energy models make them generally unsuited to WSN simulation. Many other extensions and frameworks have been developed for OMNET++, including SenSim [19] and, more

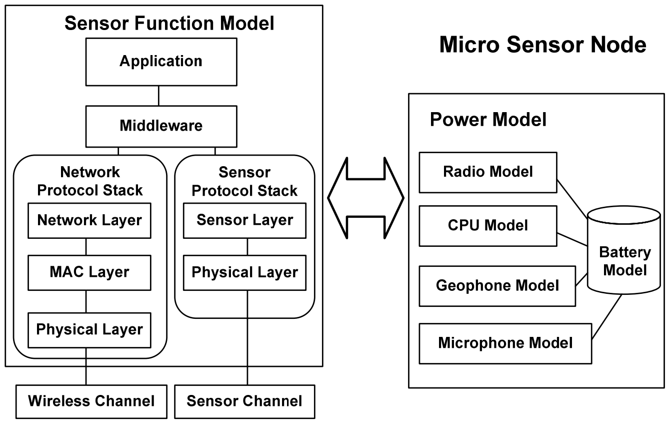

Figure 2: The structure of SensorSim (adapted from [2]).

recently, Castalia [20]. SenSim provides a sensor network extension for OMNeT++ by providing 'node-level' modules to represent each layer of the protocol stack, the node hardware (including energy models to represent a simple battery and multiple consumers, for example the CPU and radio), and a coordinator to pass messages around the node. Additional 'network-level' modules represent a sensor channel and a network channel. SenSim requires a reasonably steep learning curve, which is not usually popular with non-established simulators. The Castalia simulator is a model-centric WSN extension for OMNET++. Wireless communication is implemented using a Packet Reception Rate (PRR) based empirical model, and a 'sensing device manager' interacts with the APP to obtain sensor samples from the physical process. Sensor devices are modeled with noise and bias, and obtain data from a flexible model of physical processes. Castalia does not currently provide anything other than an 'ideal' [13] energy store.

OPNET is an object-orientated commercial network simulator, available free for academic use. However, due to scalability and extensibility issues, it is not widely used for WSN simulation. J-Sim [21] is a component-based simulator, avoiding scalability issues associated with object-orientated simulators such as ns-2. The sensor networks package for J-Sim provides a very similar node architecture to that of SensorSim [2] (as shown in Fig. 2). J-Sim is relatively complicated to use and, due to a limited established user base, is not as widely adopted as ns-2 or OMNET++.

B. *Wireless Sensor Network Simulators*

SENSE [6] improves on the efficiency of J-SIM by providing a component-orientated architecture and making improvements in the inter-communication efficiency. Energy modeling in SENSE is implemented via a battery model (accounting for dynamic battery effects), and empirical values for energy consumption in different node peripherals. However, SENSE does not model sensing with any granularity, and lacks developed modules. The SENS simulator [22] models the environment using a system of square tiles, where each tile can have different characteristics (representing different obstacles and surfaces) that affect the radio or 'sensing' signals that propagate across or through them. SENS contains only an 'ideal' battery model, and a limited architecture for the implementation of simulated embedded software.

Finally, a sub-category of WSN-specific simulators is host to those which simulate or emulate specific device hardware. TOSSIM [16] is both a simulator and an emulator for WSNs, in that it simulates TinyOS code for the Mica range of nodes and, as such, provides obvious advantages to projects that are to be implemented on the MICA nodes. ATEMU [23] increases accuracy by providing emulation of the processor on the MICA2 node, and allowing cycle-level emulation of code. This enhances simulation accuracy at the expense of scalability and speed. Avrora [24] finds a middle ground between accuracy and speed by providing cycle-level emulation, but sacrificing the continuous synchronization between nodes. Both TOSSIM and Avrora have energy consumer models developed for them (named PowerTOSSIM [25] and AEON [26] respectively). Both of these use empirical current consumption measurements (of hardware such as the radio transceiver, microcontroller and sensors) to calculate the overall power consumption. While these simulators/emulators are very suited to the evaluation of the hardware to which they are specific, they offer little benefit to the simulation of algorithms or protocols deployed on other hardware platforms (for example, deployments and evaluations using development kits such as the Texas Instruments ez430-RF2500 or CC2431EM) or that are device unspecific. Due to the device-specific nature of these simulators/emulators, they are not considered in this paper.

In Egea-Lopez *et al.* [1], the authors propose a model (shown in Fig. 3) for WSN simulation which shares many similarities with that of SensorSim (shown in Fig. 2). The model provides a physical parameter/sensor channel and a node-level physical sensor layer, and also introduces an 'energy producer' object (to represent sources of energy) in addition to the 'battery' and 'power' objects (representing energy stores and consumers respectively). While we believe that this is the model most suited to energy-aware WSN simulation that has been proposed to date, it is not without its limitations and shortcomings; these are discussed in the following section.

C. *Discussion*

The conducted review of existing simulators (summarized above) highlighted a number of limitations and shortcomings for the evaluation of energy-aware WSNs. While most simulators provide adequate support (or are at least structured to allow ease of implementation) for wireless channel modeling, energy and sensing often appear to be considered as less important functions. This is clearly not the case in many

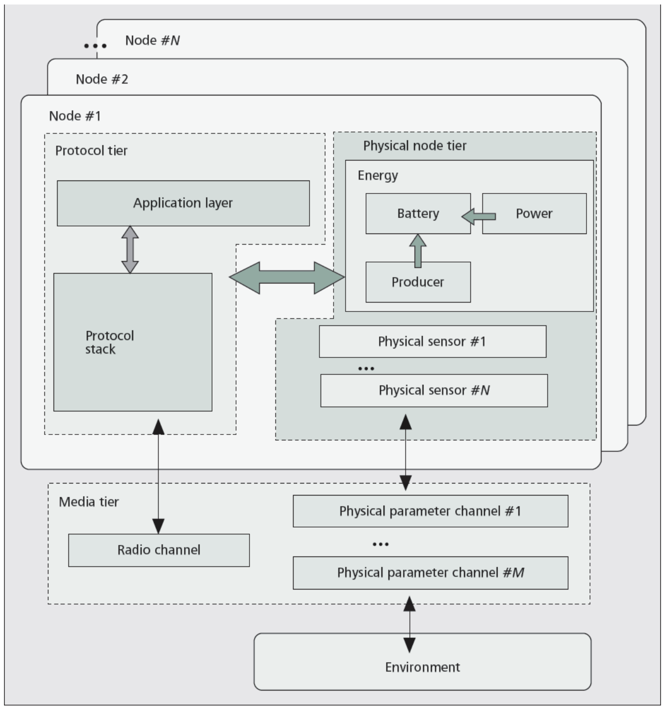

Figure 3: A network-level WSN simulation model (reproduced from [1]).

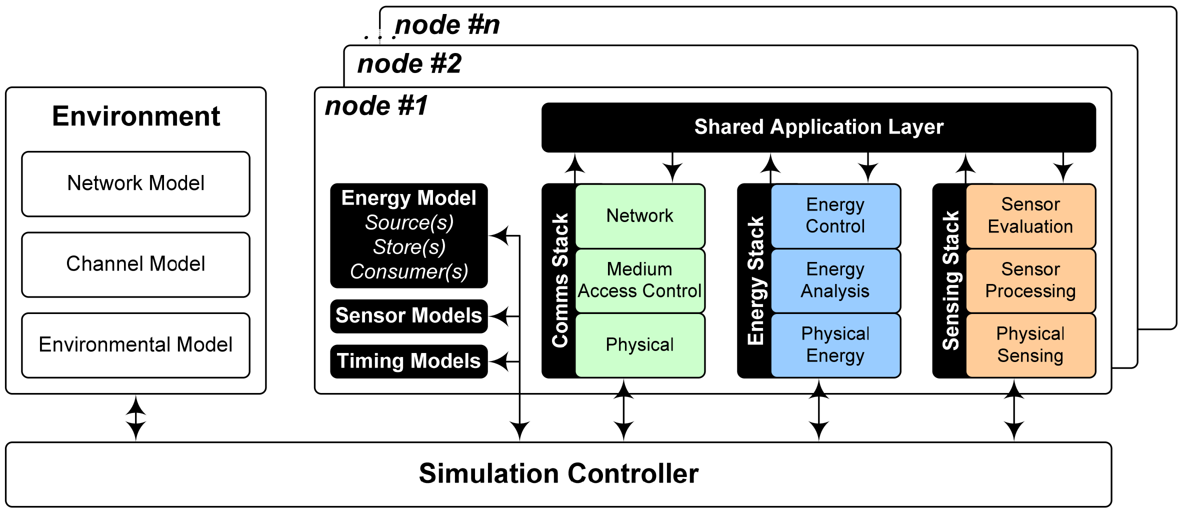

Figure 4: The proposed energy-aware structure for WSN simulation.

WSNs where energy-management and sensing are arguably more important than communication to the operation of the nodes. This shortcoming is most likely a result of the historical development from the simulation of traditional communication networks, where communication was of primary interest.

With the exception of the model proposed by Egea-Lopez *et. al* [1] (shown in Fig. 3), none of the simulators investigated provide support for energy sources (such as energy harvesting devices). Most energy sources harvest energy from the surrounding environment, and should hence interact with the same environmental models that dictate the behavior of sensors (for example, both photovoltaic cells and photodiodes use the environmental 'light' model). Furthermore, none of the simulators considered energy stores other than a battery (even though a lot of current research is being performed into supercapacitor-based devices) or provided for the inclusion of multiple storage devices (for example allowing a battery and a supercapacitor). In some simulators, energy consumption is limited to only the microcontroller or the microcontroller and radio transceiver, with the consumption of additional peripherals such as sensors, actuators and RTCs ignored.

All of the simulators investigated model the simulated embedded software as having a communication stack (usually a PHY/MAC/MET/APP), with some also providing a limited stack for sensing (consisting, at best, of a physical layer and processing layer). Energy-management and awareness is however not given any structure, instead assumed to be implemented and processed as part of the application. The integration of hardware models into the node's simulated software is often too restrictive to allow for the easy extension and development of secondary functions. For example, in many of the simulators investigated, the only method for a node to interrogate energy hardware provides the 'actual' remaining energy in the node, information which a practical sensor node would not have access to.

The following section presents a simulation structure that aims to overcome these limitations and shortcomings.

IV. A Structure for Energy-Aware WSN Simulation

As a result of the findings summarized in Section III, we propose a novel structure for energy-aware WSN simulation. To overcome the identified shortcomings, such a structure should satisfy the following requirements:

- Provision of adequate modeling for node hardware including, but not limited to, energy sources (for example energy harvesting devices), energy stores (for example batteries and supercapacitors), and energy consumers. Hardware models should be extensible to allow the future incorporation of other hardware.
- Provision of an integrated and extensible environment model to integrate with any hardware or other environmental model. The environment model should be dynamic to allow the modeling of 'real world' environments.
- Incorporation of an embedded software stack that enhances and considers the importance of energy-aware operation.

Our proposed energy-aware WSN simulation structure is shown in Fig. 4, and contains an environment model and a number of nodes (whether nodes are implemented using an object-orientated or component-based model is unrelated to the proposed structure) containing hardware models and a structured architecture for embedded software. Objects communicate with each other via the simulation controller. The primary novel contributions of this structure are 1) a flexible and integrated environment model, 2) sufficiently detailed hardware models, and 3) a structured architecture for embedded software.

1) Environment Model: The environment model (shown in the 'environment' object in Fig. 4) describes the environment into which the WSN is deployed. The model includes the network model (specifying where nodes are physically located), the channel model (dictating the propagation of data through a wireless medium), and the environmental model (containing a range of models specifying how physical parameters in the environment change through space and time). Further information on these models is given in Section V, where the models implemented in an in-house energy-aware WSN simulator (WSNsim) are outlined. The range of hardware models shown in Fig. 4 (energy, sensing and timing) are not exhaustive and are only representative of those that have been subsequently incorporated into WSNsim.

Use of the environment models is flexible, allowing the hardware models or other environmental models to access them. Furthermore, the environmental models are fully extensible; for example, an additional environmental model describing the variation of wind or atmospheric pressure could easily be added. This flexibility and extensibility aids in the evaluation of energy-aware WSNs by allowing models to be dependent on multiple physical phenomena, energy sources (and other energy hardware) to harvest energy from a modeled physical phenomena, and new energy source and sensor models to be easily added.

2) *Hardware Models:* The hardware models describe the operation of physical hardware on the node, which includes energy hardware (sources, stores and consumers), various sensors, and timing hardware (for example a Real Time Clock [RTC]). Further information on these models is given in Section V, where the models implemented in an in-house energy-aware WSN simulator (WSNsim) are outlined. Like the environment models, the hardware models are flexible and extensible, allowing new hardware to be easily integrated into a simulator. Furthermore, the hardware models are able to query the environment models (for example a photovoltaic cell or light sensor querying the environment model for the incident light level at location [*x, y*]). This flexibility and extensibility enhances energy-aware WSN simulation by giving adequate representation to the various aspects of a node's hardware.

3) *Structured Architecture for Embedded Software:* The incorporation of a 'unified' stack for embedded software in the proposed simulation structure extends our work published in [27] by highlighting the benefits that it can have on energy-aware WSN simulation. The architecture provides a sensor node with multiple stacks (each stack implementing a separate node function) that are linked by a shared application layer. The purpose and content of each of the layers of the 'unified' stack shown in Fig. 4 are described in detail in Section V. This promotes structured and modular design, allows for efficient code reuse, and enables the situation where future generations of sensor nodes can utilize interchangeable components. We believe that, in addition to being beneficial to the deployment of practical sensor nodes, this architecture can also enhance the simulation of energy-aware WSNs.

The structured architecture for embedded software can act as both an aid and a constraint for algorithm designers. Designers of energy-management or intelligent-sensing algorithms are likely to find it of assistance, providing a logical structure to shape their design process. Application or communication algorithm designers however may find it more constraining, as aspects of energy, sensing and hardware that it was not necessary to previously consider now need addressing. However these rigid design constraints are interpreted, they enforce the designer to consider fundamental aspects of a node, a process that is likely to enable their design to transfer more easily to a practical deployment. Note that it is not intended for a communication protocol or application designer to have to develop algorithms for intelligent sensing and energy-management; with community support in an established simulator, these would be available in the same way as the considerable number of different communication protocol layers are today.

Through the combination of detailed, flexible and extensible models and a structured architecture for embedded software, a simulator can provide the necessary level of hardware and environmental representation that modern energy-aware WSNs require. The proposed structure has been subsequently incorporated into an in-house simulator (WSNsim), discussed in the next section.

V. A CASE STUDY: WSNSIM

WSNsim is an in-house object orientated discrete-event based simulator for WSNs, developed by the authors using Microsoft Visual Studio .net 2005. WSNsim is built around the proposed energy-aware structure (shown in Fig. 4), providing hardware- and model-centric operation. The operation of WSNsim is coordinated by a central controller, using a packet scheduler to manage discrete event simulation. Logging and GUI components (an example of WSNsim's GUI is shown in Fig. 5) are also implemented, and operate via the simulation controller. WSNsim was developed to evaluate other research into WSNs that was being performed at the University. WSNsim is not currently available to the wider research community, as it does not have the required documentation or personnel to support it. Due to interest that it has received, this is planned future work; this paper represents the first step in

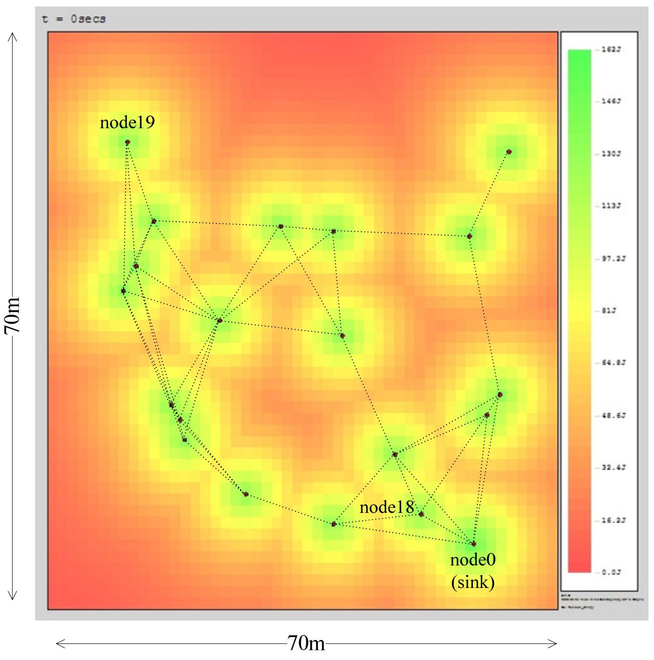

Figure 5: A network under simulation in WSNsim, showing the residual energy remaining in each node (with regular communication links annotated).

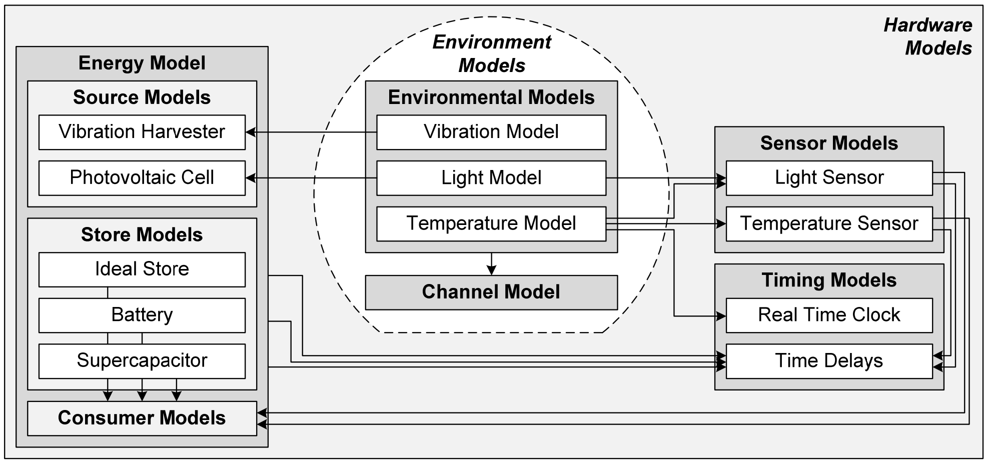

Figure 6: The interaction between hardware and environment models present in the implementation of WSNsim.

opening WSNsim to the wider research community.

To place the proposed simulation structure in context and show how it is suited to the evaluation of energy-aware WSNs, the following subsection explains the function of WSNsim's various models and software stacks. Details of the actual simulations that have been undertaken are outside the scope of this paper.

A. *Physical and Environmental Models*

Already highlighted as a requirement for a WSN simulator, interaction between different 'shared' models in the simulator is important; for example, the same environmental temperature model could be used by a temperature sensor, thermoelectric harvester, plus any other device models that exhibit temperature dependence. Fig. 6 shows the interaction between various models in the implemented version of WSNsim; the reasons for these interactions are described in context in the following subsections.

1) Energy Models: The energy model in each node contains energy source(s), energy stores(s) and energy consumer(s).

Energy sources in WSNsim provide an interface between the environment model (which dictates, for example, the intensity of the light incident on a photovoltaic cell) and the energy that is subsequently added to the node's energy store. The sources modeled in WSNsim include a photovoltaic cell (which includes consideration for the non-linear P-V relationship exhibited by a photovoltaic, whereby the instantaneous store voltage dictates the fraction of the maximum power that is transferred) and a vibration harvester.

Energy stores are modeled in WSNsim for an ideal store, a battery, and a supercapacitor. The battery store demonstrates dynamic rate discharge, relaxation, and self-discharge effects, while the supercapacitor store suffers from considerable leakage. Further information on (and evaluation of) the implemented ideal and supercapacitor store models can be found in [13].

Energy consumers are modeled in WSNsim using empirical values for the current consumption, that are multiplied by the current store voltage and the duration for which they are consuming. Consumers modeled include the microcontroller, radio transceiver, sensors, ADCs, and the RTC. Further information on the modeling and evaluation of implemented energy consumers can be found in [13].

2) Sensor Model: WSNsim contains sensor models for both a digital temperature sensor and a light sensing circuit. These models interface with the environmental models to obtain the value of the sensed parameter at the current time and location. This value is then 'sampled' by introducing inaccuracy and error, and converting the signal into a quantized digital signal or voltage. Additionally, the light sensing circuit allows the range or sensitivity to be controlled by the embedded software, through the use of a digital input.

3) Timing Model: While timing has not been considered in this paper, the timing model in WSNsim considers delays, clock drift, and RTC errors. As shown in Fig. 6, the RTC model uses the temperature model to include temperature dependence, and the time delay model gets values for delays directly from the hardware models.

4) Environmental Model: The environmental model dictates the variation of physical parameters in the environment, including temperature, vibration and light. Each model can be used by multiple hardware models (as shown in Fig. 6) and, in response to being parsed a location and a time, returns the value of the parameter.

5) Channel Model: The channel model implemented in WSNsim is very similar to that used in the Castalia simulator [20]. An empirical log-distance path loss model with log-normal shadowing [28] is used to calculate the signal strength (and hence the SNR) at the receiving node. The SNR is affected by temperature (due to the presence of thermal noise), and hence the channel model interfaces with the temperature model. Using the relationship between SNR and BER given in [29], bit errors are introduced into the received packet, which results in the packet being successfully received or else discarded.

6) Network Model: The network model is used to contain information on the location of nodes (information that is required by the environmental models, but that is hidden from the node). If a node requires location awareness (for example, for a geographical routing algorithm), a hardware model for GPS (or equivalent) should be implemented to interface with the network model.

B. *Simulated Embedded Software*

1) Energy Stack: The simulated energy stack is formed of three layers: the physical energy layer (PYE), energy analysis layer (EAN), and energy control layer (ECO). The PYE is responsible for interfacing with the node's energy model to manage and monitor the supply and demand of energy. To monitor the residual energy in a store, the PYE obtains a reading of the voltage across the store from the energy model; this is performed on receipt of a measure request from the higher layer and, when obtained, the resultant measurement is sent back to the higher layer. The EAN converts voltage measurements obtained by the PYE into meaningful statistics such as the absolute residual energy, relative residual energy, or remaining lifetime. This is performed through the use of models for the dual-supercapacitor and battery (Ni-MH, Ni-Cad, and Li-ion) stores that convert the voltage into a residual energy. The ECO takes a high-level view of the energy subsystem, and controls the energy-aware operation of the node. Upon waking up from a sleep state, the shared APP notifies the ECO which subsequently sends a measure request to the EAN to monitor the store voltage. Upon receiving a measurement of the energy residual in the store, the ECO calculates the 'energy-state' of the node.

2) Sensing Stack: the sensing stack implemented in the simulated node's embedded software is formed of three layers: the physical sensing layer (PYS), sensor processing layer (SPR), and sensor evaluation layer (SEV). The PYS interfaces with the sensor model to energize sensors and obtain raw readings. The implemented PYS handles two requests, sense and range, from the higher layer. The sense request causes the parsed sensor to be sampled, and after the necessary sensing period has passed, the reading is passed to the higher layer. The range request updates the current setting of the light sensor's sensitivity range to the value parsed, and returns the new value to the higher layer. The purpose of the SPR is to convert raw sensor readings into high-level interpreted, error corrected values with associated error bounds where appropriate. The implemented SPR handles one request, sense, from the higher layer, which is simply passed on to the lower layer. In order to convert the raw values obtained from the PYS into meaningful values, models are implemented in the embedded software. Obviously, the effects of the random variables introduced by the sensor models cannot be reversed, and hence error exists in the sampled data. The bounds of this error can be calculated through either inspection, or through analysis of the error sources. Additionally, if the reading obtained from the light sensor is nearing saturation, the SPR layer sends a range request to the lower layer in order to change the sensitivity range. The SEV layer is responsible for coordinating sensing, evaluating the significance of sensed data and performing event detection, and identifying faults in the sensor hardware.

3) Communication Stack: The communication stack implemented in WSNsim consists of three layers: the PHY, MAC, and NET. The PHY is based upon the 2.4GHz air interface specified in IEEE 802.15.4-2006 [29], implementing a complete PD-SAP, PLME-SAP and PHY-PIB. The PHY has been extended with primitives to support long preambles in order to accommodate the low-power-listening techniques that are required by the MAC layer. Finally, the NET implements packet flooding and 'minimum cost' routing algorithms.

C. *Observations and Discussion*

The previous subsection gave details of the structure implemented in WSNsim, providing an overview of the models and layers that have been incorporated. This subsection comments on the user experience of developing algorithms (communication, energy-management and intelligent sensing) using WSNsim, and highlights some of the enhanced design experiences that the proposed simulation structure offers.

WSNsim, and the proposed structure that it is based around, allowed for the evaluation of energy-aware algorithms for energy-harvesting nodes that are the subject of other research being performed at the University. The presence of a flexible environmental model allows the simple addition of new energy sources, and the level of hardware modeling means that the results produced by the simulator are believable. The structure of the embedded software that is imposed by the 'unified' stack meant that the tasks of energy-management and intelligent sensing had to be carefully considered in order to break down the task into the various layers of the stack. While this could be construed as limiting and cumbersome, it was the author's experience that this process led to algorithms that were better designed, and that were closer to that which could be practically deployable on real hardware. Furthermore, the versatility of the environmental and physical models allowed for the simple inclusion of sensor node hardware (the models in WSNsim are currently configured to consider the Texas Instruments CC2430 platform, though changing this is a simple process), indeed the extensibility of the proposed structure allowed the easy development and evaluation of a range of hardware models [13], that have been able to continue to evolve and expand throughout the process of this research.

When designing the MAC protocol in the communication stack, thought had to be given towards how the node's microcontroller and radio transceiver would be turned off or put into sleep modes at various states. This subsequently has various effects on other aspects of the embedded software, for example energy-management and sensing tasks must fit into this schedule dictated by the MAC. While not enforced or required in some other simulators, the control of different devices' power states must be considered in WSNsim, or the microcontroller and radio transceiver would remain continually active resulting in considerable energy consumption.

The overall experience of developing algorithms and protocols in WSNsim was one that was greatly aided by its structure, both in terms of the range and integration of hardware and physical models, and the structured architecture for developing embedded software.

VI. Conclusions

While simulation is well established for mobile and ad hoc networks, the simulation of energy-aware WSNs (which is the most used method of evaluation) using currently available tools has a number of shortcomings. First, most available simulators lack adequate support for the node's hardware (especially energy sources such as photovoltaics and vibration harvesters) and the needed environmental models (available environmental models are often too inflexible to allow easy integration of new hardware). Second, the node's simulated embedded hardware is structured largely around communications, giving little consideration to the implementation and interfacing of sensor processing and energy-aware algorithms. In this paper, we have presented a structure for the simulation of energy-aware WSNs. The novelties of this structure are in the provision of adequate modeling for node hardware and environment, and the integration of a structured architecture for embedded software to enhance the design of energy-aware sensor nodes. An implementation of the structure has been demonstrated in our in-house simulator, WSNsim. User experience of designing and evaluating energy-aware algorithms using WSNsim have shown that the design process can be significantly improved through the use of the proposed structure.

The next step for this research is in integrating the proposed structure into a publicly available simulator. This will either be via an extension to an established simulator such as ns-2 or OMNET++ or, alternatively, through the development of WSNsim into a package that can be adequately documented and supported.

References

[1] E. Egea-Lopez, J. Vales-Alonso, A. Martinez-Sala, P. Pavon-Mario, and J. Garcia-Haro, "Simulation scalability issues in wireless sensor networks," IEEE Communications Magazine, vol. 44, pp. 64-73, Jul. 2006.

[2] S. Park, A. Savvides, and M. B. Srivastava, "SensorSim: A simulation framework for sensor networks," in Int'l Workshop Modeling, Analysis and Simulation of Wireless and Mobile Systems, Boston, MA, 2000, pp. 104-111.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," Communications Magazine, vol. 40, pp. 102-114, Aug. 2002.

[4] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in Mediterranean Conf. Control and Automation, Limassol, Cyprus, 2005, pp. 719-724.

[5] V. Prasad and S. H. Son, "Classification of Analysis Techniques for Wireless Sensor Networks," in Int'l Conf. Networked Sensing Systems (INSS'07), 2007, pp. 93-97.

[6] G. Chen, J., M. J. Branch, L. Z. Pflug, and B. Szymanski, "SENSE: A Sensor Network Simulator," in Advances in Pervasive Computing & Networking, B. Szymanksi and B. Yener, Eds., 2004, pp. 249-267.

[7] G.-Z. Yang, Body Sensor Networks. London, UK: Springer-Verlag, 2006.

[8] C. Park and P. H. Chou, "AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes," in Sensor and Ad Hoc Communications and Networks (SECON'06), 2006, pp. 168-177.

[9] R. N. Torah, P. Glynne-Jones, M. J. Tudor, and S. P. Beeby, "Energy Aware Wireless Microsystem Powered By Vibration Energy Harvesting," in PowerMEMS, Freiburg, Germany, 2007, pp. 323-326.

[10] X. Jiang, J. Polastre, and D. Culler, "Perpetual Environmentally Powered Sensor Networks," in 4th Int'l Conf. Information Processing in Sensor Networks (IPSN'05), Los Angeles, CA, 2005.

[11] A. S. Weddell, N. R. Harris, and N. M. White, "An Efficient Indoor Photovoltaic Power Harvesting System for Energy-Aware Wireless Sensor Nodes," in Eurosensors 2008 Dresden, Germany, 2008.

[12] T. Stathopoulos, D. McLntire, and W. J. Kaiser, "The Energy Endoscope: Real-Time Detailed Energy Accounting for Wireless Sensor Nodes," in Int'l Conf. Information Processing in Sensor Networks (IPSN '08), 2008, pp. 383-394.

[13] G. V. Merrett, A. S. Weddell, A. P. Lewis, N. R. Harris, B. M. Al-Hashimi, and N. M. White, "An Empirical Energy Model for Supercapacitor Powered Wireless Sensor Nodes," in 17th International IEEE Conference on Computer Communications and Networks St Thomas, Virgin Islands (USA): IEEE, 2008.

[14] M. Mekni and B. Moulin, "A Survey on Sensor Webs Simulation Tools," in Int'l Conf. Sensor Technologies and Applications. (SensorComm '08), 2008, pp. 574-579.

[15] ISI, University of Southern California, CA, USA, "The Network Simulator - ns2." [Online]. Available: www.isi.edu/nsnam/ns. [Accessed: Jan., 2009].

[16] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in Int'l Conf. Embedded Networked Sensor Systems (SenSys'03), Los Angeles, CA, 2003, pp. 126-137.

[17] I. T. Downard, "Simulating Sensor Networks in NS-2," Naval Research Lab, Washington DC ADA423595, 31 May 2004.

[18] A. Varga, "The OMNET++ discrete event simulation system," in European Simulation Multiconference, Prague, Czech Republic, 2001, pp. 319-25.

[19] C. Mallanda, A. Suri, V. Kunchakarra, S. S. Iyengar, R. Kannan, and A. Durresi, "Simulating Wireless Sensor Networks with OMNeT++," unpublished, submitted to IEEE Computer, 2005.

[20] National ICT Australia Ltd, "Castalia - A Simulator for WSNs," Mar. [Online]. Available: http://castalia.npc.nicta.com.au/. [Accessed: Mar., 2009].

[21] A. Sobeih, J. C. Hou, L.-C. Kung, N. Li, H. Zhang, W.-P. Chen, H.-Y. Tyan, and H. Lim, "J-Sim: a simulation and emulation environment for wireless sensor networks," IEEE Wireless Communications, vol. 13, pp. 104-19, Aug. 2006.

[22] S. Sundresh, W. Kim, and G. Agha, "SENS: A sensor, environment and network simulator," in Annual Simulation Symposium, Arlington, VA, 2004, pp. 221-228.

[23] J. Polley, D. Blazakis, J. McGee, D. A.-R. Rusk, D., and J. S. A.-B. Baras, J.S., "ATEMU: a fine-grained sensor network simulator," in Conf. Sensor & Ad Hoc Communications & Networks (SECON'04), 2004, pp. 145-152.

[24] B. L. Titzer, D. K. Lee, and J. Palsberg, "Avrora: scalable sensor network simulation with precise timing," in Int'l Symp. Information Processing in Sensor Networks (IPSN'05), 2005, pp. 477-482.

[25] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in Int'l Conf. Embedded networked sensor systems Baltimore, MD, USA: ACM, 2004, pp. 188-200.

[26] O. Landsiedel, K. Wehrle, and S. Gotz, "Accurate Prediction of Power Consumption in Sensor Networks," in 2nd IEEE Workshop Embedded Networked Sensors (EmNetS-II), 2005, pp. 37-44.

[27] G. V. Merrett, A. S. Weddell, N. R. Harris, B. M. Al-Hashimi, and N. M. White, "A Structured Hardware/Software Architecture for Embedded Sensor Nodes," in 17th International Conference on Computer Communications and Networks St Thomas, Virgin Islands (USA), 2008.

[28] T. S. Rappaport, Wireless Communications: Principles & Practice: Prentice Hall, 1996.

[29] IEEE, "802.15.4™-2006," in IEEE Standard for Information Technology - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs): IEEE, 2006.