

# *Sidh*: A Wireless Sensor Network Simulator

Thomas W. Carley (tcarley@eng.umd.edu)

Department of Electrical & Computer Engineering

University of Maryland at College Park

## **1 Introduction**

Wireless sensor networks are a hot research area, and gathering increasing attention from industry. Several new workshops, conferences, and journals are dedicated to wireless sensor networks. In addition, many popular conferences and journals have sessions, tracks, or special issues dedicated to sensor networks. The research in these publications investigate new hardware platforms, protocols, applications, programming environments, etc. One open research problem is to be able to accurately and efficiently evaluate these new approaches.

In many cases it is impractical to experiment on real wireless sensor network systems; there are several reasons for this. First, a proposed hardware platform, while theoretically possible, may not be manufactured. An example of this is a low power system-on-chip hardware platform that, while possible, is not yet practical to manufacture due to up-front design, mask, and fabrication costs. Second, even if the hardware platform exists, it may be prohibitively expensive. An example of this is research protocol or application that requires hundreds or thousands of nodes to evaluate. With current nodes costing approximately one hundred of dollars, evaluating this research could cost thousands to tens of thousands of dollars. Third, even if it is practical to evaluate research on the real hardware platform, it may not be practical to experiment in an appropriate environment. An example of this are wireless sensor networks which operate on glaciers [Martinez et al. 2004], remote wildlife habitats [Szewczyk et al. 2004], detect tanks [Abdelzaher et al. 2004], and other environments with which it is expensive or dangerous to experiment.

One solution to this problem is to simulate the sensor network system. This is an approach that is common in other network research for similar reasons to those above. In fact, a number of tools are currently used to simulate sensor networks systems. These simulators have limitations that make them less than ideal for wireless sensor networks; we discuss them briefly here, and in

more detail in section 3. First, many of these simulators are designed for other types of networks; for instance, wired TCP/IP networks. Others, though wireless, are made to simulate WLAN networks and 802.11 protocols, and are also inherently IP networks. Also, some simulators that are designed specifically for sensor networks are limited to a specific programming environment, TinyOS. In addition, wireless sensor network simulators vary greatly in their accuracy of modeling the wireless medium; some only model the medium at a high level that is appropriate for roughly evaluating applications, others model the medium at the accuracy needed to evaluate low level protocols such as MAC. Finally, many simulators do not simulate the sensors, the physical phenomenon they measure, or the environment in which they operate.

We introduce *Sidh* (pronounce shee as in banshee); a wireless sensor network simulator designed to address these problems. *Sidh* is built from the ground up to simulate sensor networks. *Sidh* is efficient; it scales to simulate networks with thousands of nodes faster than real-time on a typical desktop computer. *Sidh* is component based and easily reconfigurable to adapt to different: levels of simulation detail and accuracy; communication media; sensors and actuators; environmental conditions; protocols; and applications.

The remainder of this paper is organized as follows. Section 2 expands on some background material. Section 3 describes related work in network simulators used to evaluate sensor networks. Section 4 details the design of *Sidh*. Section 5 concludes the paper.

## **2 Background**

In section 1 we state that several new workshops, conferences, and journals are dedicated to wireless sensor networks; in addition, many popular conferences and journals have special sessions, tracks, and issues dedicated to wireless sensor networks. Workshops on sensor networks include: Workshop on Security of Ad Hoc and Sensor Networks (SASN); International Workshop on RFID and Ubiquitous Sensor Networks (USN); International Workshop on Broadband Advanced Sensor Networks (BaseNets); European Workshop on Wireless Sensor Networks (EWSN); International Workshop on Heterogeneous Wireless Sensor Networks (HWISE). Conferences on sensor networks include: IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), Conference on Embedded Networked Sensor Systems (SenSys); International Conference on Sensor Networks (SENET);

Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP); International Symposium on Information Processing in Sensor Networks (IPSN). Journals include: ACM Transactions on Sensor Networks (TOSN). Popular conferences and journals with sessions, tracks, or special issues dedicated to sensor networks include: International Conference on Mobile Computing and Networking (MobiCom); International Conf. on Mobile Systems, Applications, and Services (Mobisys); IEEE Conference on Computer Communications (INFOCOM); IEEE International Real-Time Systems Symposium (RTSS); International Journal of Wireless and Mobile Computing (IJWMC); EURASIP Journal on Wireless Communications and Networking; Inderscience International Journal of Ad Hoc and Ubiquitous Computing; IEEE International Conference on Mobile and Wireless Communications Networks (MWCN); IEEE Wireless Communications and Networking Conference (WCNC).

### **3 Related Work**

As we discuss in section 1, there are a number of simulators that have been used in sensor network research. Here we discuss only simulators that can be used to evaluate wireless sensor network research. There are a number of other network simulators; however, for one reason or another they are inappropriate for evaluating wireless sensor network research.

#### **3.1 Ns-2**

*Ns* (<http://www.isi.edu/nsnam/ns/>) is a discrete event networks simulator that began in 1989 as a variant of an even earlier network simulator. It is written in a combination of C++ and OTcl, an object oriented scripting language. *Ns* is a very popular network simulator, but is effectively limited to IP networks due to low level assumptions. Support for wireless networks was added in 1997; it is designed to simulate wireless LAN protocols, though later expanded to mobile ad-hoc networks. *Ns* supports only two wireless MAC protocols, 802.11, and a single-hop TDMA protocol. The single-hop TDMA MAC protocol was added in 2000, and is still considered preliminary. A project at the Naval Research Laboratory (NRL) produced an extension to *ns* for sensor networks (<http://nrlsensorsim.pf.itd.nrl.navy.mil/>) [Downard 2004]. This extension adds a *phenom channel* for modeling physical phenomena such as sensor and the environment. However, 802.11 is used for the MAC protocol, which is not representative of sensor network MAC protocols. In conclusion, while *ns* has been used to evaluate wireless sensor networks, the

accuracy of results are questionable since the MAC protocols, packet formats, and energy models are very different from those of typical sensor network platforms.

### 3.2 GloMoSim / QualNet

*GloMoSim* (<http://pcl.cs.ucla.edu/projects/glomosim/>) [Zeng et al. 1998] began in 1998 as a simulator for mobile wireless networks. It is written in parsec, a variant of C with parallel programming extensions. The latest version of *GloMoSim* dates back to 2000. Further development has been commercialized into the *QualNet* product (<http://www.scalable-networks.com/>). *GloMoSim* has several choices for radio propagation, CSMA MAC protocols (including 802.11), mobile wireless routing protocols, and implementations of UDP and TCP. *GloMoSim* is good at simulating of mobile IP networks. However, *GloMoSim* is effectively limited to IP networks because of low level design assumptions. Therefore, it suffers the same problems as ns; the packet formats, energy models, and MAC protocols are not representative of those used in wireless sensor networks. Also, *GloMoSim* does not provide support for sensors, actuators, physical phenomena, or environmental conditions. While *GloMoSim* has been used to evaluate wireless sensor networks, the accuracy of results are questionable.

### 3.3 J-Sim

*J-Sim* (<http://www.j-sim.org/>) began in 1999 as a generic component based, compositional simulation framework. *J-Sim* is similar to *ns* in that is written in two languages; however, in *J-Sim*'s case they are Java and Jacl, a java version of Tcl. On top of this framework a general packet switching network was built. This includes many internet protocols. Support for mobile wireless networks and sensor networks was added in 2004 (<http://www.j-sim.org/v1.3/sensor/JSim.pdf>). *J-Sim* provides support for sensors and physical phenomena. Energy modeling, with the exception of radio energy consumption, is also appropriate for sensor networks. However, the only MAC protocol provided for wireless networks is 802.11. Therefore, accuracy of simulations still suffers.

### 3.4 SENSE

*SENSE* (<http://www.cs.rpi.edu/~cheng3/sense/>) is a recent sensor network simulator started in 2004 [Chen et al. 2004]. It is similar to *J-Sim* in that it is component based, but is written in C++ in order to avoid the perceived inefficiency of Java. *SENSE* runs on top of *COST*, a component

based discrete event simulator that is written in CompC++, a component extension to C++. *SENSE* supports an energy model that is sufficient for wireless sensor networks. *SENSE* supports only an unrealistic perfect NullMAC protocol, where messages arrive instantly with no possibility of collision or error, and an 802.11 MAC protocol. *SENSE* does not support sensors, physical phenomena, or environmental effects. Overall, the MAC protocol support and radio propagation make *SENSE* less than ideal for accurate evaluation of wireless sensor network research.

### 3.5 VisualSense

*VisualSense* (<http://ptolemy.eecs.berkeley.edu/visualse/>) is a modeling and simulation framework for wireless sensor networks that build on and leverages Ptolemy II (<http://ptolemy.eecs.berkeley.edu>) [Baldwin et al. 2005]. *VisualSense* provides an accurate and extensible radio model. The radio model is based on a general energy propagation model that can be reused for physical phenomena. *VisualSense* provides a sound model based on this propagation model that is accurate enough to use for localization. *VisualSense* is a good framework; however, it does not provide any protocols above the wireless medium, nor any sensor or physical phenomena other than sound.

### 3.6 (J)Prowler

*Prowler* (<http://www.isis.vanderbilt.edu/projects/nest/prowler/index.html>) and *JProwler* (<http://www.isis.vanderbilt.edu/projects/nest/jprowler/index.html>) are probabilistic wireless sensor network simulators. *Prowler* is written in Matlab, while *JProwler* is written in Java. *(J)Prowler* is targeted to the Berkeley MICA Mote hardware platform running application built on TinyOS, though it could be modified to simulate more general systems. *(J)Prowler* provides an accurate radio model. However, it provides only one MAC protocol, the default MAC protocol of TinyOS (circa 2004). *(J)Prowler* also does not provide support for sensors or physical phenomena.

### 3.7 SENS

*SENS* (<http://osl.cs.uiuc.edu/sens/>) is a high level sensor network simulator [Sundresh et al. 2004]. *SENS* is written in C++ and built with the gcc compiler. *SENS* provides a few models of wireless medium ranging from: a simple perfect model; to a probabilistic message loss model; to

an interference and collision model. *SENS* does not accurately simulate a MAC protocol. *SENS* provides support for sensors, actuators, and physical phenomena only for sound.

### **3.8 TOSSIM & TOSSF**

*TOSIM* [Levis et al. 2003] and *TOSSF* [Perrone et al. 2002.] are both sensor network simulators build specifically to simulate the Berkeley MICA Mote hardware platform running applications built on TinyOS. In the name of efficiency *TOSIM* uses an inaccurate probabilistic bit error model for the wireless medium. This is sufficient when evaluating a high level application, but not when evaluating low level protocols such as MAC. *TOSIM* does simulate the Mote's devices including digital I/O and A/D. This enables simulating sensors and actuators; however, it does not simulate the physical phenomena that are sensed. Another limitation of *TOSIM* is that each node must run the exact same code. *TOSSF* is similar to, and inspired by, *TOSIM*. *TOSSF* addresses the above limitations of *TOSIM*. One limitation of *TOSSF* is that it no longer simulates the devices as accurately as *TOSIM*.

### **3.9 ATEMU**

*ATEMU* is a software emulator for the AVR processor and the Berkeley MICA2 Mote hardware platform [Polley et al. 2004]. Since *ATEMU* is binary compatible with the MICA2 Mote hardware platform, it also supports TinyOS binary application unchanged. *ATEMU* only supports a simple free-space propagation model of the wireless medium. In *ATEMU* different sensor nodes can run different executables. The primary strength of *ATEMU* is that it is the most accurate simulator for a particular hardware platform. Conversely, the main limitation of *ATEMU* is its dependence on the MICA2 Mote hardware architecture.

## **4 Sidh**

*Sidh* is a simulator specifically designed for wireless sensor networks and is greatly influenced by related work discussed in section 3. *Sidh* is efficient; it scales to simulate networks with thousands of nodes faster than real-time on a typical desktop computer. *Sidh* is component based and easily reconfigurable to adapt to different: levels of simulation detail and accuracy; communication media; sensors and actuators; environmental conditions; protocols; and applications. *Sidh* is written in Java for portability.

*Sidh* is composed of a set of modules. A few modules are accessed via method calls; these modules are defined by a Java interface. The use of an interface allows modules to be replaced with different implementations. All other modules interact, via Simulator, through events. The use of events ensures that the timing of interactions is respected. The use of events also ensures that modules are not directly dependent on each other. Therefore, *Sidh* is configured by selecting independent modules, and can even be re-configured on-line. The selection of modules is governed by matching the events generated and consumed by modules. The modules currently supported in *Sidh* can be organized in to the following categories: Simulator; Events; Medium; Environment; Node; Transceiver; Protocols, Applications. Each category is represented by an interface that defines its methods and events generated and consumed. The following subsections discuss each of these categories, and other modules available in *Sidh*.

#### **4.1 Simulator**

Simulator is a discrete event simulator module, and is the foundation of *Sidh*. It is an implementation of the interface Simulator\_I. The interface Simulator\_I is how modules interact with the simulator. Simulator provides methods to: add and remove events; clear the event queue; generate random numbers and time values; convert time units; and run the simulation either for a given time or until the event queue is empty. Simulator uses a binary heap data structure to manage events based on the time at which they are fired.

#### **4.2 Event**

Event is an abstract base class that provides basic functionality for all events. Particular events are sub-classes of class Event. Event contains the time at which an event should fire. Event provides methods to: compare events based on their fire times; determine whether events are equal; print themselves to a string; and an abstract method to fire the event. Sub-classes of Event provide an interface for Listeners of that event. They also have members that are parameters for the event. The abstract inherited fire event is overridden to call the Interface's method with the parameters.

#### **4.3 Medium**

Medium models the wireless medium. Medium is allows nodes to broadcast signals, and is responsible for informing nodes of signals that affect it. In order to do this Medium must be

informed of the presence of every node, and any changes in position or radio properties such as transmitter power or receiver sensitivity. Medium has the properties of bandwidth and wavelength of the medium modeled. For instance a wavelength of 2.4 GHz with a bandwidth of 250 Kbps is one medium used by the 802.15.4 physical layer specification. Medium also has a reference to a propagation model that is given to it at the time of construction. The propagation model provides the strength at a particular receiver from a signal transmitted by a given transmitter.

#### **4.4 PropagationModel**

PropagationModel defines the strength of a signal at a particular receiver from a particular transmitter at a particular instant of time. The interface of PropagationModel defines methods for getting the maximum and the instantaneous signal strength for a given transmitter and receiver pair. The maximum signal strength is useful for determining interference relationships. There are many theoretical and practical models for signal propagation; these can be realized as implementations of the PropagationModel interface. Models that have been implemented include: free space propagation; two ray ground reflection; Rayleigh and Ricean fading; and Radio Interference Model (RIM) [Zhou et al. 2004]. Many more models exist, and wireless signal propagation is still an active research area.

#### **4.5 Environment**

The Environment module is similar to Medium module. The difference is that the implementation of Environment has properties that relate to the physical phenomenon modeled. Environment also has a propagation model similar to PropagationModel that models the propagation of the physical phenomena modeled. Physical phenomena of interest in sensor networks include: temperature; light; humidity; magnetic field; sound; optical; chemical presence.

#### **4.6 Node**

Node represents a single node in a wireless sensor network. As such, it serves as a container for all of the components, both hardware and software, in a node. These components include: processor; transceiver; sensors; actuators; energy source (such as a battery); network protocols; and applications. In addition each node has the properties of location and identification.



## **4.7 Processor**

The Processor module models the processor on the sensor node; examples include TI's msp430 or Atmel's AVR. This module models processor state such as low power modes and the energy consumption of each state. Processor consumes events to change its state, and generates events to notify of changes in power consumption.

## **4.8 Transceiver**

This module models the hardware transceiver on each sensor node. It models the transceiver states (i.e. sleep, standby, receive, and transmit), and their associated behavior and power consumption. Sleep mode is the lowest power state the transceiver can be in, in which even the oscillators are off. This state takes the longest to switch out of. Standby mode uses more power than sleep since the oscillators are running. Receive and Transmit use the most power since they are either powering the antenna or actively trying to decode a signal. Which mode, transmit or receive, uses more power changes with transceiver type. For instance, in a narrow band transceiver transmit uses more power than receive. However, in wideband transceivers, such as Direct Sequence Spread Spectrum (DSSS), receive uses more power than transmit.

Transceiver consumes events informing it of the beginning and ending of every signal it receives. It sums active signals to maintain the interference. Transceiver generates events for the beginning and ending of every signal it transmits. These events are all exchanged with an instance of the Medium module. Transceiver maintains properties for the Signal to Interference and Noise Ratio (SINR) thresholds necessary to begin receiving a signal, and to receive a signal without corruption. It also maintains properties for receiver gain, and transmitter power output. When a signal start event is received its SINR is checked against the receive SINR threshold; if it is greater than the threshold then the message is tracked as a potential message to receive. When the matching signal end event is received then its minimum SINR is compared to the SINR corruption threshold; if it is greater than the threshold then the message is received.

## **4.9 Sensor & Actuator**

Sensor and Actuator modules are similar to Transceiver modules. The biggest difference is that they interface to an Environment module rather than a Medium module. Another difference is in

the properties they maintain. Sensor and Actuator modules maintain properties consistent with the physical phenomenon they model.

#### **4.10 EnergySupply**

This module models the energy supply for each node. Many energy supplies are possible, but batteries are most common. Several battery technologies exist, and have different properties; for instance: alkaline; lithium; nickel cadmium; and nickel metal hydride. EnergySupply consumes events that announce changes in power consumption and voltages requirements of other modules. EnergySupply generates events only under conditions in which it cannot supply the power or voltage requirements of other modules; for instance, a battery is drained.

#### **4.11 Physical Protocol**

The Physical protocol is the lowest layer in a network stack. It is often implemented in the transceiver hardware. The Physical layer provides services for: changing the state of the transceiver; carrier sensing or Clear Channel Assessment (CCA); sending and receiving packets; received energy detection on received packets; changing channels on physical layers that support multiple channels.

#### **4.12 MAC Protocol**

The MAC protocol is the next layer in a network stack. It is usually implemented in software running on the node's processor. The MAC layer provides services for: changing the state of the MAC layer (i.e. low power mode); setting and getting protocol parameters; sending and receiving packets; etc. *Sidh* offers implementations for several sensor network MAC protocols, including: a simple CSMA MAC protocol; *Bel* [Carley et al. 2005]; *B-MAC* [Polastre et al. 2004]; and *TRAMA* [Rajendran et al. 2003].

#### **4.13 Routing Protocol**

The Routing protocol resides above the MAC protocol and provides services for routing messages over multiple hops between nodes that cannot communicate directly. One routing protocol implemented in *Sidh* is Greedy Perimeter Stateless Routing (GPSR) [Karp and Kung 2000]

#### 4.14 Application Layer

The Application layer resides at the top of the network stack. It interfaces with the lower layers in the network stack as well as the sensors and actuators to implement a wireless sensor network application. An example of a wireless sensor network application implemented on *Sidh* is EnviroTrack [Abdelzaher et al. 2004]. In [Carley et al. 2005] we use EnviroTrack to evaluate MAC protocols including *Bel*, *B-MAC*, and *TRAMA*.

### 5 Conclusions

In conclusion, in this report we introduced *Sidh*, a simulator specifically designed for wireless sensor networks. *Sidh* is efficient; it scales to simulate networks with thousands of nodes faster than real-time on a typical desktop computer. *Sidh* is component based and easily reconfigurable to adapt to different: levels of simulation detail and accuracy; communication media; sensors and actuators; environmental conditions; protocols; and applications.

### 6 References

Downard. 2004. Simulating Sensor Networks in NS-2. NRL/FR/5522--04-10073, Naval Research Laboratory, Washington, D.C., May 2004.

Zeng X., Bagrodia R., Gerla M. 1998. GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations (PADS)*, May 26-29, 1998 in Banff, Alberta, Canada.

Chen G., Branch J., Pflug M. J., Zhu L., Szymanski B. 2004. SENSE: A Sensor Network Simulator. *Advances in Pervasive Computing and Networking*, Springer: 249-267.

Baldwin P., Kohli S., Lee E. A., Liu S., Zhao Y. 2005. VisualSense: Visual Modeling for Wireless and Sensor Network Systems. *Technical Memorandum UCB/ERL M05/25*, University of California, Berkeley, CA 94720, USA July 15, 2005.

Sundresh S., Kim W., Agha G. 2004. SENS: A Sensor, Environment and Network Simulator, In *Proceedings of 37th Annual Simulation Symposium*, pages 221-230, 2004.

Levis P., Lee N., Welsh M., Culler D. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

Perrone L. F., Nicol D. 2002. A Scalable Simulator for TinyOS Applications. In *Proceedings of the Winter Simulation Conference*, 2002.

Polley J., Blazakis D., McGee J., Rusk D., Baras J. S., Karir M. 2004. ATEMU: A Fine-grained Sensor Network Simulator. In *Proceedings of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2004.

Martinez K., Ong R., Hart J. 2004. Glacsweb: a sensor network for hostile environments. In *Proceedings of the IEEE Conf. Sensor, Ad Hoc Comm. & Networks*, October 2004, pp. 81-87.

Szewczyk R., Osterweil E., Polastre J., Hamilton M., Mainwaring A., Estrin D. 2004. Wireless sensor networks: Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6), June 2004.

Abdelzaher T., Blum B., Cao Q., Chen Y., Evans D., George J., George S., Gu L., He T., Krishnamurthy S., Luo L., Son S., Stankovic J., Stoleru R., Wood A. 2004. EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, Tokyo, Japan, March 2004, IEEE, Inc., 582-589.

Zhou G., He T., Krishnamurthy S., Stankovic J.A. 2004. Impact of Radio Irregularity in Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Boston, MA, June 2004, ACM Press New York, NY, USA, 125-138.

Rajendran V., Obraczka K., Garcia-Luna-Aceves J. J. 2003. Energy-Efficient Collision-Free Medium Access Control for Wireless Sensor Networks. In *Proceedings of the 1<sup>st</sup> International Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, California, November 2003, ACM Press New York, NY, USA, 181-192.

Polastre J., Hill J., Culler D. 2004. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2<sup>nd</sup> International Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, Maryland, November 2004, ACM Press New York, NY, USA, 95-107.

Carley T. W., Barua R., Blum B. 2005. *Bel*: Periodically Scheduled Medium Access in Mesh Multi-Hop Wireless Sensor Networks. Under submission to ACM Transactions on Sensor Networks.

Karp B., Kung H.T. 2000. Greedy Perimeter Stateless Routing (GPSR) for Wireless Networks. In *Proceedings of the 6<sup>th</sup> International Conference on Mobile Computing and Networking (Mobicom)*, Boston, MA, August 2000, ACM Press New York, NY, USA, 243–254.