Wireless Sensor Network, 2011, 3, 215-240
doi:10.4236/wsn.2011.37025 Published Online July 2011 (http://www.SciRP.org/journal/wsn)

An Exploratory Study of Experimental Tools for Wireless Sensor Networks

A. K. Dwivedi[1], O. P. Vyas[2]
[1]*School of Studies in Computer Sc. & I.T., Pandit Ravishankar Shukla University, Raipur, India*
[2]*Indian Institute of Information Technology, Allahabad, India*
E-mail: {anuj.ku.dwivedi, dropvyas}@gmail.com
Received March 5, 2011; *revised June* 5, 2011; *accepted June* 25, 2011

Abstract

The objective of this contribution is to present expositive review content on currently available experimental tools/services/concepts used for most emerging field Wireless Sensor Network that has capability to change many of the Information Communication aspects in the upcoming era. Currently due to high cost of large number of sensor nodes most researches in wireless sensor networks area is performed by using these experimental tools in various universities, institutes, and research centers before implementing real one. Also the statistics gathered from these experimental tools can be realistic and convenient. These experimental tools provide the better option for studying the behavior of WSNs before and after implementing the physical one. In this contribution **63** simulators/simulation frameworks, **14** emulators, **19** data visualization tools, **46** testbeds, **26** debugging tools/services/concepts, **10** code-updation/reprogramming tools and **8** network monitors has been presented that are used worldwide for WSN researches.

Keywords: Experimental Tools, Simulation, Emulation, Testbed, Data Visualization, Debugger, Network Monitor, Code-Updater, Wireless Sensor Network

1. Introduction

Wireless Sensor Networks (WSNs) employ a large number of miniature disposable autonomous devices known as sensor nodes to form the network without the aid of any established infrastructure. In a WSN, the individual nodes are capable of sensing their environments, processing the information locally and sending it to one or more collection points through a wireless link. Research activities in the area of WSNs need expositive performance statistics about scenario, systems, protocols, gathered data and applications. There are various experimental tools for fulfilling these requirements, someone are in practical use while others are in literatures.

There are some highly cited research contributions that present comparative study of some simulators, testbeds, and other tools but this is just a little bit of an emerging broad area and also there is no any single literature that presents an extensive survey or review study on experimental tools available for WSN research purposes. The objective of this contribution is to present an extensive survey on experimental tools especially used for WSN research purposes that are based on various criteria, scenario conditions, parameters and other factors, also presenting the other relevant contents related to experimental tools, as well as focusing on the highly summarized pros and cons of mostly presented experimental tools with respect to WSNs.

2. Simulators for Wireless Sensor Networks

A simulator is a software that imitates selected parts of the behavior of the real world and is normally used as a tool for research and development. Depending on the intended usage of the simulator, different parts of the real-world system are modeled and imitated. The parts that are modeled can also be of varying abstraction level. Earlier simulators especially designed for WSN imitates the wireless media and the constraints nodes in the network but currently sensor network simulators have a detailed model of the wireless media including effects of obstacles between nodes, while other simulators have a more abstract model.

2.1. Necessity of Simulation

The emergence of wireless sensor networks brought

many new emerging issues to network designers. Traditionally, the three main techniques for analyzing the performance of wired and wireless networks are analytical methods, computer simulation, and physical measurement. Due to many constraints imposed on sensor networks, such as energy limitation, decentralized collaboration and fault tolerance, algorithms for sensor networks tend to be quite complex and usually defy analytical methods that have been proved to be fairly effective for traditional networks. Furthermore, few sensor networks have come into existence, for there are still many unsolved research problems, so measurement is virtually impossible [1]. It appears that simulation is the only feasible approach to the quantitative analysis of sensor networks.

2.2. Limitations of the Simulation

The challenge of developing, deploying, and debugging applications on the realistic environment will be unmet with simulations. Many of the current simulators are unable to model many essential characteristics of the real world. Simulations are based on common simplified assumptions and these do not produce accurate results. The simulation results are only as good as the model and they are still only estimated or projected outcomes. Especially for wireless sensor networks simulation models do not capture the radio and sensor irregularity. In a research contribution [2] some issues are arises that really influencing the simulation results when simulators are directly used:

The first one is that there should be an impact on simulation results by operating system architecture on which simulator is installed and result would have been taken.

The second one is that all of cases simulators use a simulated clock, which advances in constant increments of time. Constant increment of time is decided by time stamp of the earliest event. So we must replace simulation clock with real system clock.

The third one is that all simulators has its own protocol stack in its core (kernel) called simulator protocol stack. There are several problems in order to use a simulator protocol stack as a real network protocol stack, such as most of the simulators have various redundant protocols at various levels of simulator protocol stack to support other types of networks for example TCP/IP based networks, Wireless Mesh Networks, Mobile Ad hoc Networks etc. The simulator also includes different radio, mobility, and propagation models. So according to our view these unnecessary redundant components must be removed in order to use a simulator protocol stack as a real protocol stack for WSNs.

The fourth one is that all simulators have its own architecture and design objectives. We have not evaluated but we can say that these factors could also influence the simulation results.

2.3. Type of Simulation

Simulators either run as in an asynchronous mode, event triggered mode, or in synchronous mode, where events happen in parallel in fixed time slots [3]:

- *Synchronous Simulation*: The synchronous simulation is based on rounds. At the beginning of each round, the simulators increments the global time by one unit. Then, it moves the nodes according to their mobility models and updates the connections according to the connectivity model. After that, the framework iterates over the set of nodes and performs these steps for each node.
- *Asynchronous Simulation*: The asynchronous simulation is purely event based. The simulator holds a list of message events and timer events, which is sorted by the time when these events should happen (arrival of message, execution of timer-handler). The simulator repeatedly picks the most recent event and executes it.

In general, the asynchronous simulation mode runs much faster than the synchronous mode. The main reason lies in the fact that the synchronous simulation mode loops over all nodes and performs for each node the set of fixed steps even if most of the nodes may not do anything at all. Whereas in asynchronous mode, only message and timer events are processed and no unnecessary cycles are wasted.

2.4. Categorization of Simulators

In a research contribution WSN Simulators are categorized [4] as:

- *Generic Network Simulators*: Generic network simulators simulate systems with a focus on networking aspects. The user of the simulator typically writes the simulation application in a high level language different from the one used for the real sensor network. Since the focus of the simulation is on networking the simulator typically provides detailed simulation of the radio medium, but less detailed simulation of the nodes.
- *Code Level Simulators*: Code level simulators use the same code in simulation as in real sensor network nodes. The code is compiled for the machine that is running the simulator, typically a PC workstation that is magnitudes faster than the sensor node. Typically code level simulators are operating system specific

since they need to replace driver code for the sensors and radio chips available on the node with driver code that instead have hooks into the simulator.

- *Firmware Level Simulators*: These simulators are based on emulation of the sensor nodes and the software that runs in the simulator is the actual firmware that can be deployed in the real sensor network. This approach gives the highest level of detail in the simulation and enables accurate execution statistics. This type of simulation provides emulation of microprocessor, radio chip and other peripherals and simulation of radio medium. Due to the high level of detail provided by firmware level simulators, they are usually slower than code level or generic network simulators.

In another research contribution [5], simulators have been classified into the following three major categories based on complexity:

- *Algorithm Level Simulators*: Some simulators focus on the logic, data structure and presentation of the algorithms. For example, AlgoSensim analyzes specific algorithms in WSNs, e.g. localization, distributed routing, flooding, etc. Shawn is targeted to simulate the effect caused by a phenomenon, improve scalability and support free choice of the implementation model. Sinalgo offers a message passing view of the network, which captures well the view of actual network devices.
- *Packet Level Simulators*: Some simulators implement the Data Link and Physical Layers in a typical OSI network stack. The most popular and widely used network simulator NS-2 is not originally targeted to WSNs but IP networks. SensorSim is an extension to NS-2 which provides battery models, radio propagation models and sensor channel models. J-Sim adopts loosely-coupled, component-based programming model, and it supports real-time process-driven simulation. GloMoSim is designed using the parallel discrete-event simulation capability provided by PARSEC.
- *Instruction Level Simulators*: Some simulators model the CPU execution at the level of instructions or even cycles. They are often regarded as emulators. They compute the power of a particular sensor's hardware platform in WSNs. TOSSIM simulates the TinyOS network stack at the bit level. Atemu is an emulator that can run nodes with distinct applications at the same time.

Several simulators exist that are either adjusted or developed specifically for wireless sensor networks. Here is a list presenting **63** simulators/simulation frameworks with their highly influential features related to WSNs:

- Network Simulator [6,7] (specially higher versions, like NS-3) has been used to evaluate WSNs but the accuracy of results with lower versions (NS-2) are questionable since the MAC protocols, packet formats, and energy models are very different from those of typical sensor network platforms. NS-3 is a discrete-event network simulator for Internet systems. NS-3 is a new simulator (not backwards-compatible with NS-2).
- Mannasim (NS-2 Extension for WSNs) [8] is a wireless sensor networks simulation environment based on the NS-2. Mannasim extends NS-2 introducing new modules for design, development and analysis of different WSN applications. Having Script Generator Tool (SGT) for TCL script creation.
- TOSSIM [9,10] is a TinyOS mote simulator which is useful for testing both the algorithms and implementations; however it does not simulate the physical phenomena that are sensed.
- TOSSF [11] is very similar to, and inspired by TOSSIM. TOSSF addresses the limitations of TOSSIM but one limitation of TOSSF is that it no longer simulates the devices as accurately as TOSSIM.
- PowerTOSSIMz [12] is a power modeling extension to TOSSIM. PowerTOSSIM accurately models power consumed by TinyOS applications, *i.e.* efficient power simulation for TinyOS applications.
- ATEMU [13] is a Sensor Network Emulator/Simulator/Debugger. The primary strength of ATEMU is that it is most accurate simulator for a particular hardware platform. Conversely, the main limitation of ATEMU is its dependence on the Mica-2 Mote hardware architecture.
- COOJA [14] is a Contiki OS simulator which allows for cross-level simulation. It is a novel type of wireless sensor network simulation that enables holistic simultaneous simulation at different levels. In COOJA one simulation can contain nodes from several different abstraction levels. These are the network level, the operating system level, and the machine code level.
- GloMoSim (Global Mobile Information Systems Simulation) [15] suffers the same problems as NS, *i.e.*, the packet formats, energy models, and MAC protocols are not representative of those used in WSNs. While GloMoSim has been used to evaluate WSNs but the accuracy of results is questionable.
- QualNet [16] is the commercial version of GloMoSim with upgraded features such as, providing a comprehensive environment for designing protocols, creating and animating experiments, and analyzing the results of those experiments.
- SENSE [17] 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 WSNs re-

search.

- VisualSENSE [18] is a good framework but it does not provide any protocols above the wireless medium, nor sensor or physical phenomena other than sound.
- AlgoSenSim [19] is a framework used to simulate distributed algorithms. It is not protocol stack oriented but algorithm oriented. It focuses on network specific algorithms like localization, distributed routing, flooding etc. AlgoSenSim is easily modularly: It uses XML configuration file. It is efficiency oriented, but optimizations are hidden to the user. AlgoSenSim's main purpose is to facilitate the implementation and quality analysis of new algorithms.
- Georgia Tech Network Simulator (GTNetS) [20] is a full-featured network simulation environment that allows researchers in computer networks to study the behavior of moderate to large scale networks, under a variety of conditions. The design philosophy of GTNetS is to create a simulation environment that is structured much like actual networks are structured.
- OMNet++ [21] [22] is an extensible, modular, component-based C++ simulation library and framework, with an Eclipse-based IDE and a graphical discrete event simulator.
- Castalia [23] is OMNet++ Extension for WSNs and can be used by researchers and developers who want to test their distributed algorithms and/or protocols in realistic wireless channel and radio models, with a realistic node behavior especially relating to access of the radio. Castalia can also be used to evaluate different platform characteristics for specific applications, since it is highly parametric, and can simulate a wide range of platforms.
- J-Sim (formerly JavaSim) [24] is a truly platform-neutral, component-based, compositional simulation environment. 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 IEEE 802.11. Therefore, accuracy of simulations still suffers.
- JiST/SWANS (Java in Simulation Time/Scalable Wireless Ad hoc Network Simulator) [25]: JiST is a high-performance discrete event simulation engine that runs over a standard Java virtual machine. It is a prototype of a new general-purpose approach to building discrete event simulators, called virtual machine-based simulation. SWANS is a scalable wireless network simulator built atop the JiST platform. Its capabilities are similar to NS-2 and GloMoSim but are able to simulate much larger networks.
- JiST/SWANS++ [26] is an extended version of JiST/SWANS provides more realistic and meaningful simulation results.
- Avrora [27] is a cycle-accurate instruction level sensor network simulator which scales to networks of up to 10,000 nodes and performs as much as 20 times faster than previous simulators with equivalent accuracy, handling as many as 25 nodes in real-time. Avrora's ability to measure detailed time-critical phenomena can shed new light on design issues for large-scale sensor networks.
- Sidh [28] is a simulator specifically designed for WSNs. Sidh is efficient; it scales to simulate networks with thousands of nodes faster than real-time on a typical desktop computer. It 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.
- Prowler [29] is a probabilistic wireless sensor network simulator. Prowler is written in MATLAB and also running under MATLAB thus providing an easy way of application prototyping with nice visualization capabilities.
- (J) Prowler [30] is a discrete event simulator similar to Prowler but written in Java. The simulator supports pluggable radio models and MAC protocols and multiple application modules. Currently two radio models are implemented: Gaussian and Rayleigh, and one MAC protocol: Mica-2 with no acknowledgment. Though it could be modified to simulate more general systems. It does not provide support for sensors or physical phenomena.
- LecsSim [31] is a simulator for large wireless networks provides an easy way to simulate distributed algorithms in wireless networks. It includes propagation models, modules for common node functionality and documentation.
- OPNET [32] is slightly different from NS and GlomoSim, it supports the use of modeling different sensor-specific hardware, such as physical-link transceivers and antennas. It can also be used to define custom packet formats. OPNET suffers from the same object-oriented scalability problems as NS.
- SENS [33] is a component-based simulator with four main components: application, network, physical, and environment. The former three components make up the sensor node. SENS is less customizable than any other simulator, providing no opportunity to change the MAC protocol, along with other low level network protocols.
- EmStar/Em* [34,35] is a software environment for developing and deploying complex WSN applications on networks of 32-bit embedded Microserver plat-

forms, and integrating with networks of Motes. EmStar consists of libraries that implement message-passing IPC primitives, tools that support simulation, emulation, and visualization of live systems, both real and simulated, and services that support networking, sensing, and time synchronization.

- EmTOS [34,35] is an extension to EmStar. It can be used either for deployment or simulation of WSNs. It enables a complete NesC/TOS app to run unmodified under EmStar.
- SenQ [36] is an accurate and scalable evaluation framework for sensor networks that integrates sensor network operating systems with a very high-fidelity simulation of wireless networks such that sensor network applications and protocols can be executed, without modifications, in a repeatable manner under a diverse set of scalable environments. SenQ extends beyond the existing suite of simulators and emulators in four key aspects: it supports emulation of WSN applications and protocols in an efficient and flexible manner; it provides an efficient set of models of diverse sensing phenomena; it provides accurate models of both battery power and clock drift effect which have been shown to have a significant impact on sensor network studies; and finally it provides an efficient kernel that allows it to run experiments that provide substantial scalability in both the spatial and temporal contexts.
- SIDnet-SWANS [37] is a simulation-based environment that enables run-time interactions with the network for the purpose of observing the behavior of algorithms protocols in the presence of various conditions such as phenomena fluctuations, or a sudden loss of service both at an individual node, as well as a collection of nodes.
- SensorSim [38] is a simulation framework that inherits the core features of traditional event driven network simulators, and builds up new features that include ability to model power usage in sensor nodes, hybrid simulation that allows the interaction of real and simulated nodes, new communication protocols and real time user interaction with graphical data display.
- Shawn [39] is a discrete event simulator for sensor networks. Due to its high customizability, it is extremely fast but can be tuned to any accuracy that is required by the simulation or application.
- SSFNet (Scalable Simulation Framework) [40] is a command-line-based simulator. Accordingly, the realization of specific application scenarios and the user interaction is difficult. SSFNet focuses on static application scenarios. An important feature of SSFNet is the possibility to parallelize the simulation. This speedup enables the analysis of large scale network behaviour. Both toolkits are limited to a single communication interface per node.
- Atarraya [41] simulator is specifically focused on the evaluation of topology control protocols in WSNs.
- NetTopo [42] is a research oriented sensor network simulator. NetTopo has the functions of general sensor networks but specially reflects the research results of following: Streaming Data Gathering and Topology Prediction in WSNs within Expected Lifetime, Reward Oriented Packet Filtering Algorithm for Heterogeneous Sensor Networks, VIP Bridge: Integrating Several Sensor Networks into One Virtual Sensor Network, Transmitting Streaming Data in Wireless Sensor Networks with Holes and many more.
- WiseNet [43] is a software simulator that can be very useful to carefully plan and select the right type of motes and sensors in a cost-effective manner. WiSeNet simulates random distribution of sensors. Through repeated experimentation it is possible to arrive at an optimal spatial configuration of the sensors that is most effective for a given application. WiSeN*et al*so allows the wireless range of a sensor to be varied and study the effects on the application.
- SimGate [44] is a full-system simulator for the Intel Stargate, intermediate-level, resource-constrained, sensor network device.
- SimSync [45] is a time synchronization simulator for wireless sensor networks. SimSync models the distribution of packet delay and the frequency of crystal oscillator as Gaussian.
- SNetSim [46] is event-driven simulation software for WSNs running on Windows based operating systems.
- SensorMaker [47] is a simulator for wireless sensor networks. It supports scalable and fine-grained instrumentation of the entire sensor networks.
- TRMSim-WSN (Trust and Reputation Models Simulator for Wireless Sensor Networks) [48] is a Java-based simulator aimed to test trust and reputation models for WSNs.
- PAWiS [49] is a simulation framework for WSN that provides functionality to simulate the network nodes with their internal structure as well as the network between the nodes. One main feature is the contemporaneous simulation of the power consumption of every single node. The framework is based on the discrete event simulator OMNeT++. The user defined model (expressed with C++ classes) is compiled to an executable simulator.
- OLIMPO [50] is a discrete-event simulator for WSN, designed to be easily reconfigured by the user, providing a way to design, develop and test communication protocols.

- DiSenS (Distributed SENsor network Simulation) [51] is a complete scalable and extensible distributed simulation system for sensor networks, which provides a cycle-accurate device emulator that is extendable by various fidelity-enhancing models (radio, power etc.) for tunable simulation accuracy. A key distinguishing feature of DiSenS is that it is implemented for distributed-memory parallel cluster systems.
- WISDOM [52] simulator is written in Java and uses the recursive porous agent simulation toolkit (Repast) O as the simulation engine to perform discrete event-driven simulations. WISDOM can be to simulate and verify middleware services for routing, sensing activity scheduling, group formation and management, target detection and tracking, and collaborative classification and fusion in a wireless sensor network. The versatility and performance of WISDOM for middleware service protocol development and evaluation have proven to be valuable.
- Sinalgo [3] is a simulation framework for testing and validating network algorithms for WSN. Unlike most other network simulators, which spend most time simulating the different layers of the network stack, Sinalgo focuses on the verification of network algorithms, and abstracts from the underlying layers: It offers a message passing view of the network, which captures well the view of actual network devices?
- Sensoria [53] is a fully fledged simulator for WSNs that has considerable differences to all other existing simulators. Sensoria is very powerful in simulating a range of small to large scale WSNs based on a simple and complete Graphical User Interface (GUI). Sensoria's GUI allows users to design various simulation scenarios and display the simulation results graphically with many formats. Sensoria is a component-based simulator and it can be easily reconfigured to adapt to different levels of simulation details and accuracy.
- Capricorn [54] is a large-scale discrete-event wireless sensor network simulator developed at Wayne State University.
- H-MAS (a Heterogeneous, Mobile, Ad-hoc Sensor-Network Simulation Environment) [55] provide a convenient platform on which to evaluate a variety of MAS (Mobile Ad-hoc Sensor nets) configurations at the physical, medium access, network, and application layers, and to extract meaningful design rules from the experimental data. Also provide an intuitive visualization that can give insight to the design engineer and casual observer alike.
- Stargate Simulator (starsim) [56] is a full-system simulator for Stargate, the XScale-based gateway device for WSN. It also boots original Linux image from xbow. It also features an XScale pipeline simulator to provide cycle estimation.
- Mote simulator (motesim) [56] is a full-system, cycle-accurate simulator for Mica-2 and Mica-Z motes. It runs with original TinyOS binaries.
- SNSim [57] is a prototype software tool, designed to support the balance the lifetime of a WSN and the quality of data (QoD) that is sampled and processed. Including elements of power consumption characteristics and built to mimic real performances of Mica motes (both in data transfer rate and power consumption - on/off modes), this graphical interface tool is created towards investigating various aspects of development, as well as building applications/simu- lations for such networks. Both SNSim and its event driven simulation engine are written in Java, which offers an enhanced portability and efficiency of development time.
- SNIPER-WSNSim [58] is a less known simulator that is specifically designed for WSNs and benefits from the richness of the .Net framework 3.5 and from the portability of the C# language. It is a graphical interface based simulator that deals with particular sector of WSN development such as sensor nodes distribution, routing protocols and clustering.
- SNAP [59] is defined as an integrated hardware simulation and deployment platform. It is a microprocessor that can be used in two ways: as the core of a deployed sensor or as a part of an array of processors that performs parallel simulation. Again, “real” code for sensors can be simulated. By combining arrays of SNAPs (called Network on a Chip), it is claimed to be able to simulate networks on the order of 100,000 nodes.
- SimPy [60] is a bare simulation written in Python. In SimPy, the basic simulation entities are processes. They are executed in parallel and may exchange Python objects among each other. Most processes include an infinite loop in which the main actions of the process are performed. Besides abstractions for processes and the related exchange of objects, SimPy provides instructions for the synchronization of simulation processes and commands for the monitoring of simulation data. Unlike the other simulators, there is no public available network models exist for it.
- Mule [61] is a hybrid simulator that combines the ease of debugging multiple simulated motes on a host PC with high fidelity of message transmission and sensor data acquisition of physical motes.
- CaVi [62] provides a uniform interface to state-of-the-art simulation methods and formal verification methods for WSN. Due to the probabilistic behavior of WSNs systems, however, the simulation covers

only a small fraction of all possible behaviors. Formal model checking techniques, based on Markov Decision Processes, use less detailed and more abstract models and compute exact probabilities and expected values for the entire behavior, where simulation can only give averages. It allows for creating a single model for simulation, Monte-Carlo simulation, and model checking.

- Ptolemy [63] is a discrete event simulator and a design tool for concurrent, real time, embedded systems. It could be used to simulate WSNs. In fact VisualSense, which is a framework built on top of Ptolemy, is intended to assist researchers in the design, visualization and simulation of wireless sensor networks. In Visual Sence sensor nodes could be defined using either the discrete event blocks or the continuous time and real time blocks available on Ptolemy. In addition, the sensor nodes could also be written in Java to meet specific needs.
- Maple [64] is a simulator which allows researchers and WSNs developers to focus on particular aspects such as sensor nodes distribution and WSNs lifetime estimation. Moreover, the parallel processing capability presents an important feature for further distributed simulations. Furthermore, the intuitive and convivial user interface makes the simulator accessible for the average users while being flexible and scalable for improvements for advanced users.
- WISENES (Wireless Sensor Network Simulator) [65] simulates high level WSN protocol and application designs and provides accurate information about their performance in a real environment. The WISENES framework implements models for transmission medium (for modeling wireless communications), sensing channel (for physical phenomena) and nodes (for physical node platforms). The designer selects the protocols from the library or implements new ones in SDL and integrates them to the WISENES framework. The framework components and node protocols communicate using SDL signals. A node model can be dynamically instantiated separately for each simulated node. Thus, virtually any number of nodes can be simulated simultaneously.
- WSNet-Worldsens and WSim [66]: WSNet is an event driven, large scale wireless sensor network simulator. WSNet uses models for applications, protocols and radio medium communication with a parameterized accuracy. WSim can be connected to WSNet, in place of the application and protocol models used during the high level simulation to achieve a full distributed application simulation. WSNet and WSNet+WSim allow a continuous refinement from high level estimations down to low level real-time validation.
- LSU Sensor Simulator [67] is a framework for simulating WSNs. It is a customizable and extendible simulator, which allows testing and analyzing software for WSNs. The users can subclass the framework classes and customize the behavior of various network layers. This sub classing gives a way to the developers and an opportunity to analyze and investigate phenomenological, networking, robustness and scaling issues, to explore arbitrary algorithms for distributed sensors, independent of hardware constraint.
- WSNGE [68] is a flexible and extensible environment that provides a highly scalable simulator with unique characteristics such as: focuses on user friendliness, providing every function in both scriptable and visual way, allowing the researcher to define simulations and view results in an easy to use graphical environment. Unlike other solutions, WSNGE does not distinguish between different scenario types, allowing multiple different protocols to run at the same time. It enables rich online interaction with running simulations, allowing parameters, topologies or the whole scenario to be altered at any point in time.
- TikTak [69] is a scalable simulator for WSNs including hardware/software interaction. Specifically allows the design exploration and the complete microprocessor-instruction-level debug of network formation, data congestion, nodes interaction, all in one simulation environment. An innovative feature is the co-emulation of selected nodes at clock-cycle-accurate hardware processing level, allowing code debug and exact execution latency evaluation (considering both protocol stack and application), together with other nodes at abstract protocol level, meeting a designer's needs of simulation speed, scalability and reliability. The simulator is centered on the Zigbee protocol and can be retargeted for different node micro-architectures.

3. Emulators for Wireless Sensor Networks

As a networked embedded system, a WSN application involves sensor node hardware, its drivers, operating systems, and networking protocols. As a result, the performance of the WSN application depends on all of these factors in addition to its implementation. An emulator is a special type of simulator whose aims is to enable realistic performance evaluation for WSN applications. Emulation environment or emulators are good choice, in which WSN applications can be directly run for testing, debugging, and performance evaluation. Additionally, studies on the lower layers (e.g., hardware drivers, OS, and networking)

as well as cross-layer techniques can also be done in this environment by plugging the target modules into the emulator. For example, emulators can compute the power of a particular sensor's hardware platform in WSNs [70,71].

Here is a list which presents **14** emulators with their highly vital features related to WSNs:

- VMNET (Virtual Mote Network) [70,71] has a highly modularized architecture for assembling virtual hardware components. Target WSN is emulated as a virtual mote network. The CPU of a mote (sensor node) is emulated at the CPU clock cycle level, and the sensing units and other hardware peripherals are also emulated in sufficient detail. The radio signal transmission is emulated by the communication between VMs with the effects of signal loss and noise. Moreover, VMNet takes parameter values from the real world and logs detailed running status of application code. As a result, the binary code of the target WSN application can be run directly on the VMN, and the application performance, both in response time and in power consumption, can be reported realistically in VMNet.
- ATEMU [13] is a sensor network emulator/simulator/debugger for AVR processor based systems. Along with support for the AVR processor, it also includes support for other peripheral devices on the Mica-2 sensor node platform such as the radio. Atemu can be used to perform high fidelity large scale sensor network emulation studies in a controlled environment. The atemu emulator core can simulate arbitrary numbers of nodes and can model their execution and interactions between them, such as radio communications in extremely fine detail. It offers nearly complete emulation of the Mica-2 hardware platform and as a result provides results that are closer to real life operation of a distributed sensor network. The only difference between running an actual network of the Mica-2 sensor nodes and emulating it in Atemu is the operation of the "air".
- Emstar [34,35] is a programming model and software framework for creating Linux-based sensor network applications that are self configuring, reactive to dynamics, and can either be interactively debugged or operate without user interaction. The goal of EmStar is to facilitate a more direct interaction with underlying modules by doing away with strict layering, but in a way that does not sacrifice very much modularity or layers of conceptual abstraction. The EmStar execution environment makes code easier to debug. The same code and configuration can be run on real nodes (either using low-power radios such as motes, or 802.11), as a pure simulation, or in a hybrid mode that combines processing done in simulation and communication, sensing, and actuation on real (physical) channels. The same source code can be used in any of these modes without changes. Developers can seamlessly iterate between simulation and reality.
- TOSSIM [9,10]: Initially, work on TinyOS was very low level, exploring things such as media access, sensor filtering, and timer implementations. The initial design of TOSSIM was focused on this work: it simulates every bit of the Mica platform radio i stack. As this work was matured, more and more effort has been spent on higher layers, such as complex applications. In order to support developers of these larger systems, TOSSIM currently implementing a packet-level simulation for the Mica-2 platform. The challenge is to capture all of the issues and problems that can arise in communication (timing, packet corruption, MAC) while remaining efficient.
- AvroraZ/Avrora [27]: AvroraZ, is an extension of the Avrora emulator - the AVR Simulation and Analysis Framework—which allows the emulation of the Atmel AVR microcontroller based sensor node platforms with IEEE 802.15.4 compliant radio chips thus allowing emulation of sensor nodes such as Crossbow's Mica-Z. AvroraZ is based on design, implementation and verification of several extensions to Avrora such as address recognition algorithm, indoor radio model, clear channel assessment (CCA) and link quality indicator (LQI) of the IEEE 802.15.4 standard.
- Freemote [72] emulator is a lightweight and distributed Java based emulation tool for developing WSN softwares. The objective of this platform is to support the emerging Java based Motes based on optimized JVM (for example, Squawk, Sentilla Point) and platforms (for example, Java Cards, SunSPOT). The Freemote emulator focuses on behavior credibility by mixing emulated nodes and real nodes reachable through a specialized bridge rather than on time based performance evaluation accuracy. This emulator splits the Software architecture of a Mote in three independent layers connected through well defined interfaces (Application, Routing and Data Link and Physical). Freemote is a fully configurable WSNs emulator. It can easily be used to develop new algorithms for WSNs but is also capable to support large scale experiments (up to 10,000 nodes) including all kind of real nodes based on the IEEE 802.15.4 communication standard. It also allows the developer following the behavior of WSNs and debugging tricky implementation problems.
- EmPro [73] is an environment/energy emulation and profiling system for WSNs. It accurately outputs electrical signals to emulate not only digital and ana-

log inputs to the sensors but also the power sources as well as RF attenuation according to pre-programmed sequences. This emulation approach enables researchers to run the networked sensors in real-time in a realistic manner with full controllability and reproducibility. EmPro in profiling mode can also capture the observable behavior of WSNs for detailed analysis. Experimental results on the Eco and Mica-2 WSN platforms show that EmPro can drive these hardware systems in real-time with high accuracy.

- NetTopo [42]: With respect to the simulation module, users can easily define a large number of on-demand initial parameters of sensor nodes, e.g. residential energy, transmission bandwidth, and radio radius. Users also can define and extend the internal processing behavior of sensor nodes, such as energy consumption, bandwidth management. It allows users to simulate extremely large scale heterogeneous networks. Since the sensor node attributes and internal operations are user definable, this feature guarantees that the simulated virtual nodes have the same properties with those of real sensor nodes. The sensed data captured from the real sensor nodes can drive our simulation in a pre-deployed virtual WSN. Additionally, topology layouts and algorithms of virtual WSN are customizable and work as user-defined plug-ins, both of which can easily match the corresponding topology and algorithms of real WSN testbed.
- OCTAVEX [74] wireless sensor framework is designed to assist end users, systems integrators, software developers, and OEMs (Original Equipment Manufacturers) in the deployment and management of WSNs. The framework provides a backbone for WSN applications while taking an approach that is hardware and standards agnostic, allowing the user to implement an end to end solution quickly, easily and at a much lower cost than developing one in-house. It is able to simultaneously support any number of sensor points using different types of wireless protocols (including mesh networks like Zigbee, Wireless HART, point to point RF sensors, Bluetooth, WiFi, RFID tags). The framework captures incoming sensor data at the OCTAVEX Universal Gateway. Once the data comes into the framework, the OCTAVEX Core Services provide built in business logic such as archiving, reporting, alerting and trending. Through Web Services and software APIs, the OCTAVEX Framework can be easily integrated into other enterprise applications, building automation systems, or industrial process and control systems.
- SENSE [17] does not support sensor's physical phenomena, or environmental effects. Overall, the MAC protocol support and radio propagation make SENSE less than ideal for accurate evaluation of WSNs research.
- UbiSec&Sens [75] will also prototype implementations on emulators and on actual sensor networks.
- Emuli [76] is a method of effectively substituting sensor data by synthetic data on physical wireless nodes (motes). That is, a method of emulating sensor stimuli of sensors. Emuli implements a model of a sensor behavior. In contrast to the earlier ap-proaches, does not record and play back spot measurements. Instead, Emuli stores the model parameters. This results in a rather compact data memory footprint and a convenient and flexible sensor model. Emuli is d esigned to increase the capability of sensor testbeds and other de-ployments to experiment with environment sensing and monitoring.
- MSPSim [77] is a Java-based instruction level emulator of the MSP430 series microprocessor and emulation of some sensor networking platforms. Supports loading of IHEX and ELF firmware files, and has some tools for monitoring stack, setting breakpoints, and profiling.
- MEADOWS [78] is a software framework for modeling, emulation, and analysis of data of wireless sensor networks. This software framework is motivated by the unique need of intertwining modeling, emulation, and data analysis in studying sensor databases.

4. Data Visualization Tools for Wireless Sensor Networks

With the increase in applications for sensor networks, data manipulation and representation have become a crucial component of sensor networks. The data gathered from WSNs is usually saved in the form of numerical form in a central base station. There are many programs that facilitate the viewing of these large amounts of data. These special programs are called data visualization tool for WSNs. Visualization tools can support different data types, and visualize the information using a flexible multi-layer mechanism that renders the information on a visual canvas.

Here is a list presenting **19** data visualization tools [79] that are especially designed and developed for WSNs applications:

- SpyGlass's [80] aim is to ease the life for sensor network debugging, evaluation and deeper understanding of the software by visualizing the sensor network, its topology, the state and the sensed data. SpyGlass has a very flexible drawing and plug-in architecture. The visualization framework consists of three major functional entities: The sensor network,

the gateway nodes located in the sensor network and the visualization software.

- MoteView [81] monitoring software is a Crossbow's product to visualize WSNs which provides users to simplify deployment and monitoring. It also makes it easy to connect to a database, to analyze, and to graph sensor readings. The Mote-view features topology and network statistics visualization as well as logging of sensor readings and the viewing of the logged data. The statistics function includes the end-to-end data packet yield, a prediction for the future and the RF link quality, but is limited to these features. It allows querying the sensor network for collected data in a database-like manner, hiding the distribution of the data collection software on the sensor nodes.
- TinyViz [9,10] is not just a visualization tool but a software framework to which application specific user plugins can be added to suite specific simulation requirements. It visualizes sensor readings; LED states; radio links and allows direct interaction with running TOSSIM simulations. The architecture of TinyViz allows adding application specific visualization functionality. This functionality includes specialized drawing operations, subscription and reaction to events and providing feedback to the TOSSIM simulator. It is very tightly coupled to the TinyOS software, the TOSSIM simulator and the Mica sensor network hardware.
- Surge Network Viewer [76] is Crossbow's product to visualize wireless sensor networks. It is a Java application that comes standard in the TinyOS Tools distribution. The Surge Network Viewer is useful for monitoring a sensor network and analyzing mesh network performance.
- MonSense [82] applications are very modular and have various extension points. It reuses various software libraries in order to reach the intended behaviour. The MonSense application displays the existing connections (routes) as an undirected graph, whose nodes are the sensor devices and edges are the current connections. MonSense can be used for different goals like planning, deployment, monitoring and control of WSNs. This application is intended to serve two different types of users: WSN Customers and WSN Researchers. The gathered data must be easily understood by the final users and, optionally, this data can be published in the intern*et al*lowing the access to the information without the need to any previous software installation, through the use of html, plain text or images.
- NetTopo [42] is an extensible integrated framework for the Simulation of virtual WSN, the visualization of real testbed, and the interaction between simulated WSN and testbed to assist investigation of algorithms in WSNs.
- Octopus [83] is also a WSN Visualization and Control tool. Its main Objective is to provide flexible access and control of deployed sensor networks.
- TOSGUI [82] project is composed of modular components that can be used to create a customized application. Unfortunately, the component architecture is tightly connected with the TinyOS operating system and the MOTE hardware platform.
- MSR Sense [84] project is also able to collect data from a WSN and visualize it, but the visualization can't be done in real time and the software is not platform independent.
- Trawler [85]: The Trawler application from MoteIV is well suited for monitoring small sized WSNs but, as the size increases the current network state becomes less obvious.
- SNAMP (self-developed Sensor Network Analysis and Management Platform) [86] is a novel multi-sniffer and multi-view visualization platform for WSNs. In SNAMP, data emitted by individual sensor nodes is collected by a multi-sniffer data collation network and passed to a flexible multi-view visualization mechanism. SNAMP indicates network topology, sensing data, network performance, hardware resource depletion, and other abnormalities in WSNs and allows developers adding application specific visualization functions, which will facilitate the research and development of various sensor networks and shorter the time from laboratory to applications.
- MeshNetics WSN Monitor [87] tool shows the network topology, sensor data and the signal quality between the nodes. The WSN Monitor automatically generates network topology diagrams as network nodes are detected and added to the system. These nodes are then regularly monitored, with any sensor data received automatically displayed in charts and tables on a PC screen. MeshNetics WSN Monitor features an XML-based framework for rapid customization of user interfaces and measured sensor parameters.
- Mica Graph Viewer [88] is a 2D visualization and monitoring tool for WSN.
- MARWIS [89] is a management architecture for heterogeneous WSNs, which supports common management tasks such as visualization, monitoring, (re)configuration, updating and reprogramming. It uses a wireless mesh network as a backbone and offers mechanisms for visualization, monitoring, reconfiguration and updating program code. Using a graphical user interface, the topology of the heterogeneous WSN with all the sensor sub-networks is vi-

sualized.

- Oscilloscope [90] tool is also used to show the sensing data graphically on host screen and visualizing tool for the nodes.
- GSN [91] is a software middleware for a variety of WSNs. It facilitates the viewing of large amount of data that is gathered form WSNs and saved in the form of numerical data in a central base station.
- WiseObserver [92] tool visualizes and analyzes data collected by a WSN in a generic scope of application. It also tries to establish a sensor network control interface. The tool will include several facilities to treat sensor network data. It allows the generation of evolution charts, interpolation maps, evolution data videos, and report generation. It also includes modules to add external data not collected by nodes, but related to the network conditions. Node Management will be possible thanks to the execution of commands in network nodes, to perform changes in network operation.
- SenseView [93] is a tool that enables hierarchical and visual browsing of physical location information and sensor values. Visual maps can be created by composing polygons, each with the ability to link to a different view. Access to real-time data is provided by directly subscribing to event nodes captured as links in the map. The event nodes also provide attribute information describing the sensors. Map information is fetched from a dedicated map server with its own access control lists based on SOX authentication. Much like a web browser with hyperlinks, SenseView allows a user to traverse through different views by clicking on different parts of the map. The user can select and subscribe to available event nodes given the correct permissions.
- XbowNet [94] is a CrossBow's sensor network visualization tool for xbow sensor nodes. A corresponding software driver called xServe is installed on gateway for the purpose of converting sensed data into XML stream and providing a TCP/IP service on port 9005, which can be used for visualization.

5. Testbeds for Wireless Sensor Networks

To achieve high-fidelity in WSN experiments use of testbed is very prolific. Testbeds are an environment that provides support to measure number of physical parameters in controlled and reliable environment. This environment contains the hardware, instrumentations, simulators, various software and other support elements needed to conduct a test. Generally, testbeds allow for rigorous, transparent and replicable testing. Obstacles to using testbeds are:

- *Large Scale (LS)*: Until today, due to limited financial support it is very expensive to buy and maintain a testbed with large number of sensor nodes.
- *Not Replicable Environment (NRE)*: For hazardous applications deploying a real testbed can cause serious damage of sensor nodes and testbeds.

By providing the realistic environments for testing the experiments, the testbeds bridge the gap between the simulation and deployment of real devices. The testbeds thus deployed can improve the speed of innovation and productive research.

Here is a list presenting **28** testbeds in highly conclusive manner used for experimental purposes in various universities, colleges, research institutions or by individuals:

- Motelab [95] is an experimental WSN deployed in Maxwell Dworkin Laboratory, the Electrical Engineering and Computer Science building at Harvard University. MoteLab consists of a set of permanently-deployed sensor network nodes connected to a central server which handles reprogramming and data logging while providing a web interface for creating and scheduling jobs on the testbed. MoteLab accelerates application deployment by streamlining access to a large, fixed network of real sensor network devices; it accelerates debugging and development by automating data logging, allowing the performance of sensor network software to be evaluated offline. Additionally, by providing a web interface MoteLab allows both local and remote users access to the testbed, and its scheduling and quota system ensure fair sharing. The MoteLab source is freely available, easy to install, and already in use at several other research institutions.
- Tutornet: A Tiered Wireless Sensor Network Testbed [96] currently consists of 13 clusters, with each cluster consisting of a stargate and several motes attached to it via USB cables. These stargates communicate with a central PC over 802.11 b, from where any node on the testbed can be programmed. Thus a testbed consisting of 13 stargates and 104 motes (91 tmoteSky and 13 Mica-Z). Tiered sensor network testbed: consists of 3 tiers. It provide remote and parallel programming mote.
- WUSTL [97] testbed at Washington University currently consists of 79 wireless sensor nodes (motes). This testbed deployment is based on the TWIST architecture originally developed by the telecommunications group (TKN) at the Technical University of Berlin. It is hierarchical in nature, consisting of three different levels of deployment: sensor nodes, microservers, and a desktop class host/server machine.
- CitySense [98] is an urban scale sensor network testbed that is being developed by researchers at

Harvard University and BBN Technologies. CitySense will consist of 100 wireless sensors will cover the city of Cambridge, MA, with wireless-sensor nodes mounted to telephone poles that could allow researchers to see the specific locations and times of day when pollution peaks. Each node will consist of an embedded PC, 802.11 a/b/g interfaces, and various sensors for monitoring weather conditions and air pollutants. Most importantly, CitySense is intended to be an open testbed that researchers from all over the world can use to evaluate wireless networking and sensor network applications in a large-scale urban setting.

- Kansei [99] at the Ohio State University is a large-scale testbed including both 210 Extreme Scale Motes (XSM) and Extreme Scale Stargates (XSS). The devices are specially designed for Kansei testbed. The topology is using both Ethernet and 802.11b wireless LAN to control the testbed. Kansei also provide a web interface for users to upload programs, scheduled jobs, and retrieve results with EmStar software framework.
- MistLab [100] consists of a mixture of 47 Mica-2 nodes and 14 Cricket nodes spread across multiple rooms located on the 9th floor of MIT's CS department.
- Orbitlab [101] is short for Open-Access Research Testbed for Next-Generation Wireless Networks (including WSN also). It supports experimental research on a broad range of wireless networking issues and application concepts with various network topologies and network layer protocol options. It also supports virtual mobility for mobile network protocol and application research.
- Emulab [102] is a network emulation testbed, giving researchers a wide range of experimental environments in which to develop, debug, and evaluate their systems. In addition to fixed wireless nodes (currently predominantly 802.11), Emulab also features wireless nodes attached to robots that can move around a small area. These robots consist of a small body (shown on the right) with an Intel Stargate that hosts a mote with a wireless network interface. The goal of this "mobile wireless testbed" is to give users an opportunity to conduct experiments with wireless nodes that are truly mobile. TrueMobile and Mobile Emulab are some modified versions for dynamic WSNs.
- WISEBED (Wireless Sensor Network Testbeds) [103] provides a multi-level infrastructure of interconnected testbeds of large-scale wireless sensor networks for research purposes, pursuing an interdisciplinary approach that integrates the aspects of hardware, software, algorithms, and data.
- REALnet [104] is an embryonic environmental WSN at the "Campus del Baix Llobregat" of the UPC (Universitat Politècnica de Catalunya). The technical objective of REALnet is to monitor physical parameters from the air (atmospheric temperature, humidity and pressure, and ambient light), ground (humidity, temperature) and water (level, temperature, conductivity).
- KonTest [105] is a 60-node indoor wireless sensor network testbed, distributed among six office rooms located on the fourth floor of the Faculty of Sciences of Vrije Universiteit Amsterdam. The testbed includes 60 TelosB-class nodes.
- SANDbed (Sensor Actuator Network Development Testbed) [106] is an integrated testbed system for WSN monitoring and management. SANDbed consists of 3 levels of hardware components organized in a hierarchical tree. The root level comprises the user interface, where the management of the testbed and configuration of the experiments is taking place. Management nodes connected to the Internet form the second level. They are responsible for managing the testbed nodes and controlling the execution of experiments. The leaves of the tree are the testbed nodes, consisting of a mote and the SNMD.
- BANAID [107] consists of seven Mica-2 motes and two Stargate sensor devices. It is the first actual testbed that shows the visibility of the Wormhole attack in WSN and mainly used to simulate the wormhole attack on a wireless sensor network.
- CENSE (a Century of Sensor nodes) [108] providing flexible modular platform for testing and optimization of nodes for Sensor Network applications. `CENSE' currently provides for processing, communication, sensing and power modules but the design can be easily extended to add more modules like mobility and localization. This test bed is very flexible, cost efficient, easy to use, power efficient, provides excellent debugging facilities and covers all major requirements of sensors networks. Each node of the testbed consists of four modules: Power, Processor, Sensor and Communication. System has been design to accommodate more modules, if needed.
- WINTeR (Wireless Industrial Sensor Network Testbed for Radio-Harsh Environments) [109] is an open access, multi-user experimental facility (MXF) that supports the development and evaluation of wireless sensor networks (WSNs) for radio-harsh environments (RHEs). The testbed supports the R&D of emerging WSN technologies, including protocols, security, physical layer, the validation of wireless solutions for industrial processes, propagation models, and cross-layer optimization.
- NESC-Testbed [110] 1.0 provides an actual platform

for the testing and developing of algorithm, protocol in WSN. B/S 3-tier and wireless-wired combined framework are used in this testbed, which makes the system user-friendly, easily used, robust and stable.

- SWOON (Secure Wireless Overlay Observation Network) [111] is an emulation-based testbed for real world experiences and scalable tests over an overlay network, consisting of wireless sensor networks, 802.11 a/b/g, etc. It can evaluate protocols, mechanisms and techniques for secure wireless communication. Researchers and designers can create their own topologies and run experiments on the SWOON testbed without re-establishing and re-installing hardware and software modules required for their wireless networks. In addition, the SWOON testbed also allows researchers to monitor the network traffic, evaluate the performance of the protocols under test and validate the researches they presented.
- INDRIYA [112] is a large-scale 3D WSN testbed with 140 TelosB nodes deployed at the National University of Singapore. The Testbed facilitates research in sensor network programming environments, communication protocols, system design, and applications. It provides a public, permanent framework for development and testing of sensor network protocols and applications. Users can interact with the Testbed through an intuitive web-based interface designed based on Harvard's Motelab's interface.
- CLARITY [113] Centre for Sensor Web Technology in Ireland is currently constructing a ubiquitous robotics testbed by integrating a collective of mobile robots with a WSN and a number of portable devices. The new, mixed testbed will be hosted at University College Dublin, (UCD), and will also avail itself of the laboratory facilities hosted in Dublin City University (DCU) and Tyndall, Cork. This testbed integrates and extends some pre-existing facilities, specifically: WSN of 70 Berkeley motes measuring humidity, light and temperature, 10 mobile robots, equipped with an array of state-of-the-art sensors, including USB cameras, laser range finders, sonar, infrared, odometers and bumpers. Each robot carries a mote able to measure ambient variables, which is also equipped with triple-axis accelerometers, magnetometer, compass and microphone, a variable number of Internet gateways, a variable number of PDAs and mobile phones equipped with Bluetooth.
- Imote2 Sensor Network Testbed [114]: In its current version, the testbed consists of a set of Crossbow Imote2 nodes programmed with a Linux kernel and running localization and routing codes written in C. One node is connected to a PC via USB and acts as a base node for the network. The PC runs a Java-based GUI intended as an interface for a user to read data from the nodes and to issue commands to the network. The user can control and observe the performance of various localization protocols in the network which are run locally in the Linux operating system on each device.
- WSNTB [115] is designed for heterogeneous WSN experiments. It involves two WSNs and three gateways. Each WSN has 17 sensor nodes. According to users' requirements, users can choose the single one or both of WSNs, with or without the gateways to experiment. Users can use both the web-based interface and the special function, called local mode, to run their applications on testbed.
- TWIST [116] testbed is owned by Technical University Berlin. They help users load programs and run experiments such as time synchronization and power control. The system is divided into two major parts. The first part is the server to serve the demands of users and control all of nodes. The second part includes two types of sensor nodes, eyesIFX v2 and Telos motes which are plugged onto the switch. The architecture is extended form the UC Berkeley's Omega testbed and Motescope testbed.
- ENL Sensor Network Testbed [117] is intended to provide a multi-hop sensor network that could be used for the real time analysis and evaluation of sensor network application. The ENL sensor network testbed consist of a number of mote assemblies hanging from the ceiling forming a grid pattern. Each mote assembly consists of a Mica mote and the standard programming board. The testbed provides means of remotely programming the motes and collecting data from the testbed.
- X-sensor [118] is a new sensor network testbed integrates multiple sensor networks deployed at different sites. X-sensor provides three functionalities: (a) a sensor network search which enables users to find a sensor networks appropriate for experiment and data acquisition, (b) a sensor data archive which provides users with various sensor data acquired by sensor nodes, and (c) an experimental testbed which enables remote users to evaluate their network and data management protocols.
- GNOMES [119] is a lowcost hardware and software testbed. This testbed was designed to explore the properties of heterogeneous wireless sensor networks, to test theory in sensor networks architecture, and be deployed in practical application environments.
- PICSENSE [120] is a single hop WSN testbed which will send the sensor information to the gateway node. The gateway node will act as an embedded web server which serves the web pages with the dynamic data.

Most of the testbed uses commercial gateway nodes like stargate which is not as flexible as the Rabbit gateway node, which is used in this testbed to integrate the WSNs with the IP networks. The firmware in the Rabbit gateway node can be designed to make the configuration of the gateway node either as a HTTP server or a FTP server or a simple router.

- SOWNet [121] Technologies T301 Testbed is primarily a WSN testbed consisting of SOWNet G-Node G301 wireless sensor nodes. Each sensor node is attached to a G-Node testbed adaptor module for connecting the sensor emulation feature. Up to 4 G-Nodes and GTA301 modules can be interfaced to a single mini PC. The mini PC is connected to an IP network using its wired or wireless networking capability. This allows many mini PCs and a vast number of G-Nodes to form a WSN testbed together and still be managed from a single management console, possibly over the internet.
- NetEye [122] is a high-fidelity testbed consists of 130 TelosB motes at Wayne State University. In addition to providing a local facility for supporting research and educational activities, NetEye is being connected to Kansei as a part of the Kansei consortium. NetEye testbed consists of a controlled indoor environment with a set of sensor nodes and wireless nodes deployed permanently. NetEye testbed provides a web interface to create and schedule a job on the testbed while automated reprogramming of the sensor devices and storing the experimental data on to the server.

In addition to above discussed 28 WSN testbeds there are also a number of other academic and industrial testbed deployments. Some of these WSN testbeds are: SenseNet [123], Omega [124], Motescope [124], Sharesense [125], Trio [126], sMote [127], CTI-WSN Testbed [128], FEEIT WSN Testbed [129], Roulette [130], BigNet [131], UCR Wireless Networking Research Testbed [132], IP-WSN [133], WHYNET [134], CENS-Testbed [135], SCADDS WSN Testbeds [136], Crossbow WSN Testbed [5], GaTech Testbed [137], Intel Research Berkeley's 150-mote SensorNet Testbed [138].

6. Debuggers for Wireless Sensor Networks

Due to extreme resource constraints nature, deployment in harsh and unattended environments, lack of run-time support tools and limited visibility into the root causes of system and application level faults make WSNs notoriously difficult to debug. Currently, most debugging systems in WSNs are aimed at diagnosing specific faults, such as detection of crashed nodes, sensor faults, or identifying faulty behavior in nodes. There are few debugging solutions for WSNs available, with a fairly wide range of goals and feature sets. Debuggers for WSNs have been categorized [139] into three distinct categories:

- Source-level debuggers
- Query-oriented debuggers, and
- Decision-tree debuggers.

Here is a list presenting **26** debuggers and debugging concepts with their summarized content related to WSNs:

- Clairvoyant [140] is comprehensive source-level debugger for wireless, embedded networks. With Clairvoyant, a developer can wirelessly connect to a sensor network and execute standard debugging commands including break, step, watch, and back trace, as well as new commands that are specially designed for debugging WSNs. Clairvoyant attempts to minimize its effect on the program being debugged in terms of network load, memory footprint, execution speed, clock consistency, and flash lifetime.
- Dustminer [141] is a tool for uncovering bugs due to interactive complexity in networked sensing applications. Such bugs are not localized to one component that is faulty, but rather result from complex and unexpected interactions between multiple often individually non-faulty components. Because of the distributed nature of failure scenarios, this tool looks for sequences of events that may be responsible for faulty behavior, as opposed to localized bugs such as a bad pointer in a module. With this tool an extensible framework is developed where front-end collects runtime data logs of the system being debugged and an offline back-end uses frequent discriminative pattern mining to uncover likely causes of failure. The tool helped uncover event sequences that lead to a highly degraded mode of operation. Fixing the problem significantly improved the performance of the protocol.
- Sympathy [142] is a tool for detecting and debugging failures in sensor networks. Sympathy has selected metrics that enable efficient failure detection, and includes an algorithm that root-causes failures and localizes their sources in order to reduce overall failure notifications and point the user to a small number of probable causes.
- FIND [143] is a novel method to detect nodes with data faults that neither assumes a particular sensing model nor requires costly event injections. After the nodes in a network detect a natural event, FIND ranks the nodes based on their sensing readings as well as their physical distances from the event. It works for systems where the measured signal attenuates with distance. A node is considered faulty if there is a significant mismatch between the sensor data rank and the distance rank.

- REDFLAG [144] is the fault detection service for WSN applications, a Run-timE, Distributed, Flexible, detector of faults that is also Lightweight and Generic. REDFLAG addresses the two most worrisome issues in data-driven wireless sensor applications: abnormal data and missing data. REDFLAG exposes faults as they occur by using distributed algorithms in order to conserve energy. Simulation results show that REDFLAG is lightweight both in terms of footprint and required power resources while ensuring satisfactory detection and diagnosis accuracy. Being unrestrictive, REDFLAG is generically available to a myriad of applications and scenarios. As a matter of fact, REDFLAG has been applied into a subsurface contaminant transport model to improve the model performance in the presence of erroneous sensor data.
- Chowkidar [145] is a stabilizing protocol that provides accurate and efficient network health monitoring in WSNs. This approach adapts the well-known problem of message-passing rooted spanning tree construction and its use in propagation of information with feedback (PIF) for the case of a WSN. The Chowkidar protocol is initiated upon demand; that is, it does not involve ongoing maintenance, and it terminates with accurate results, including detection of failure and restart during the monitoring process. Chowkidar is distinguished from others in two important ways. Given the resource constraints of WSNs, it is message-efficient in that it uses only a few messages per node. Also, it tolerates ongoing node and link failure and node restart, in contrast to requiring that faults stop during convergence. Chowkidar protocol has been implemented as part of enabling a network health status service that is tightly integrated with a remotely accessible wireless sensor network testbed, Kansei, at the Ohio State University.
- ActorNet [146] is an agent based framework for dynamically programming and debugging WSNs. end-users (WSN operators but not necessarily programmers) define actors in an expressive, high-level language to specify debugging logic. The framework allows actors to move through the network in order to accomplish their objective.
- Debugging WSNs Using Mobile Actors [147] approach is for post-mortem debugging of WSNs using autonomous and mobile actors. By allowing the computation (mobile actor) to move to the nodes where the data is located, to overcome the necessity of moving the data while still providing the flexibility necessary to diagnose errors in WSNs. In this approach two mechanisms are defined for debugging—namely, forward tracking and backward tracking in which an actor, starting at an error state, tracks the causal events, respectively, forward or backward in time in order to determine the root cause of the error.
- Monitored External Global State (MEGS) [148] [149] is a tool that leverages existing debugging techniques to recreate (part of) the global state of a WSN on an external PC. A global state is the combined local states of all nodes in a system. The idea is that the global state of a system does not only allow a developer to see what is happening inside a WSN at a given time, but also relevant events that happened earlier. Using the recreated state the developer of a WSN can gain insight into the operation of the WSN. MEGS also allows the developer to define assertions and predicates on the recreated state to easily find locations in the execution where anomalous behavior occurred.
- Declarative Tracepoints [150] is a debugging system that allows the user to insert a group of action-associated checkpoints, or tracepoints, to applications being debugged at runtime. Tracepoints do not require modifying application source code. Instead, they are written in a declarative, SQL-like language called TraceSQL independently. By triggering the associated actions when these checkpoints are reached, this system automates the debugging process by removing the human from the loop. Declarative tracepoints are able to express the core functionality of a range of previously isolated debugging techniques, such as EnviroLog, NodeMD, Sympathy, and StackGuard.
- Envirolog [151] is a distributed service that improves repeatability of experimental testing of sensor networks via asynchronous event recording and replay. To use EnviroLog, an application programmer needs only to specify two types of simple annotations to the source code. Automatically, the preprocessor embeds EnviroLog into any desired level of an event-driven architecture. It records all events generated by lower layers and can replay them later to upper layers on demand.
- NodeMD [152] is a fault management system designed to improve node debugging capabilities prior to deployment, and enable remote debugging on in-situ sensor nodes that fail. This system successfully implements lightweight run-time detection, logging, and notification of software faults on wireless mote-class devices. It introduces a debug mode that catches a failure before it completely disables a node and drops the node into a state that enables further diagnosis and correction, thus avoiding on-site redeployment. It offers simple annotations to trace and avoid critical errors in specific parts of the code.

- StackGaurd [153] is a more generic tool for detecting stack corruption caused by buffer overruns (e.g., when the return address of a function is over-written). This problem is also known as the buffer overrun security exploit. Having received intensive attention, this problem is addressed in multiple ways, and StackGuard is one of the better-known techniques in that it virtually eliminates all buffer overflows with the help of the canary word. More specifically, StackGuard modifies the generated prologue and epilogue code for functions to insert canary words. The assumption held by StackGuard is that if some code in a function modifies the return address, it must have modified the canary word as well, assuming that the application does not know the value and size of the canary word. By checking the integrity of the canary word, StackGuard can detect malfunctioning code.
- KleeNet [154] is a debugging environment for high-coverage testing of sensor network applications before deployment. It enables the detection of bugs that result from complex interactions of multiple nodes, nondeterministic events in the network, and unpredictable data inputs. KleeNet executes unmodified sensor network applications on symbolic input and automatically injects non-deterministic failures. As a result, KleeNet generates distributed execution paths at high-coverage, including low-probability cornercase situations. Built on the symbolic virtual machine KLEE, KleeNet makes the following four key contributions and facilitates rigorous testing of distributed WSN applications and protocols: Coverage, Non-determinism, Distributed Assertions and Repeatability.
- Marionette [155] is a system that allows calling functions and inspecting and changing memory locations in deployed nodes through an RPC-based system. A developer can use this much like a debugger, although because it uses the wireless channel of the network it has a large impact on the operation of the network.
- Passive Distributed Assertions (PDA) [156] allows developers to detect such failures and provides hints on possible causes. PDA allow a programmer to formulate assertions over distributed node states using a simple declarative language, causing the sensor network to emit information that can be passively collected (e.g., using packet sniffing) and evaluated to verify that assertions hold. This passive approach allows us to minimize the interference between the application and assertion verification. Further, this system provides mechanisms to deal with inaccurate traces that result from message loss and synchronization inaccuracies.
- Nucleus [157] network management system (NMS) facilitates monitoring of running WSN applications by providing access to the internal data structures of TinyOS nesC components over the network. To reduce interference of the Nucleus system and the WSN application, data is only sent over the network in response to a user query. In addition to this query approach, unexpected events are logged to persistent local storage and can be retrieved later from the node on demand.
- MDB [158] is a GDB like post-mortem debugging system for wireless embedded networks that exploits macro programming to provide four abstract views of a system state: 1) the temporally synchronous view, 2) the logically synchronous view, 3) the historical view, and 4) the hypothetical view. It enables application development and debugging at a single level of abstraction. It eliminates the need for a programmer to reason about low-level event traces and message passing protocols, instead allowing debugging in terms of abstract data types.
- SNTS (Sensor Network Troubleshooting Suite) [159] uses distributed sniffer sensor nodes that record overheard traffic in local Flash storage. After an experiment, the nodes are collected and the packet traces are transferred to a central server. SNTS decodes the raw packet dumps based on a text file that describes the packet format. As an example for a possible processing of the packet traces, the authors employed machine-learning algorithms to identify bad sequences of events, which lead to an observed bug in the protocol/system, allowing them to fix the problem.
- ANDES [160] is a framework for detection and finding the root causes of anomalies in operational WSNs. The key novelty of ANDES is that it correlates information from two sources: one in the data plane as a result of regular data collection in WSNs, the other in the management plan implemented via a separate routing protocol, making it resilient to routing anomaly in the data plane. Unlike existing WSN diagnosis tools, it does not distinguish among faults due to software or hardware bugs and those induced by security threats or intrusion. Rather, it utilizes specifications of the targeted application known apriori and normal behaviors established by a self-learning algorithm to identify potential anomalies. Localizing the faulty entities is made possible by incorporating inferred knowledge of routes and fault signatures on a central node (typically the sink).
- EvAnT [161] allows for specifying queries that are executed on the collected traces. EvAnT is specifically tailored to WSN testing and debugging.

- Storage-centric method for Debugging [162]: In this method performance and log data are stored locally on the node. This method opens new possibilities for debugging: A node can process the data itself; Neighbors can share performance and log data to detect root causes of problems and maybe even resolve problems; An operator can in an efficient one-hop batch download the data using a mobile node; A node can decide on its own when it is time to send stored log data to the sink for processing; The sink can request data on demand at the desired level of granularity.
- Model-based diagnosis for WSNs [163] is a technique where a model of a system is combined with observations from that system, to generate diagnoses for failures of the system. The basic premise of this solution is to use the distributed nature of a WSN to solve these problems. By running a simple inference engine on every node, with a small local model, nodes can generate conflict sets. If the model can be made to only use local observations, observations no longer need to be sent over the network. As a final step, conflicts generated by the local inference engine can be sent to the sink node, which then calculates the minimal hitting set of these conflicts to produce the diagnoses.
- Post-Deployment Performance Debugging (PD2) [164] is a data-centric approach that focuses on the data flows that an application generates, and relates poor application performance to significant data losses or latencies of some data flows (problematic data flows) as they go through the software modules on individual nodes and through the network. PD2 derives a few inference rules based on the data dependencies between different software modules, as well as between different nodes, and use them to trace back in each problematic flow. Then, PD2 turns on the performance monitoring of, and collects debugging information from, only those modules and nodes that the problematic flows go through. Finally, PD2 provides the debugging information to help users isolate the causes of poor performance.
- S_2DB [165] is a debugger based on a distributed full system WSN simulator with high fidelity and scalable performance, DiSenS. By exploiting the potential of DiSenS as a scalable full system simulator, S_2DB extends conventional debugging methods by adding novel device level, program source level, group level, and network level debugging abstractions.
- Wringer [139] is an integrated debugging framework that allows for rapid prototyping and deployment of debugging tools: passive, active, in-network, and gateway-based. The goal is to utilize these rapid-prototyping capabilities to discover the core set of debugging primitives that can detect and fix the majority of WSNs bugs.

7. Code-Updaters for Wireless Sensor Networks

Large scale WSNs may be deployed for long periods of time during which the requirements from the network or the environment in which the nodes are deployed may change. This may necessitate modifying the executing application or re-tasking the existing application with different sets of parameters, which will collectively refer to as code-updation/reprogramming. The relevant forms of code-updation/reprogramming are [166]:

- *Remote Multi-hop Reprogramming*: It is the most relevant form of code-updation/reprogramming which uses the wireless medium to reprograms the nodes as they are embedded in their sensing environment.
- *Incremental Reprogramming*: It is also attractive because it transfers a small delta (difference between the old and the new software) so that code-updation/reprogramming time and energy can be minimized.

Incremental Reprogramming poses several challenges. A class of operating systems, including the widely used TinyOS, does not support dynamic linking of software components on a node. SOS and Contiki, do support dynamic linking, however, their reprogramming support also does not handle changes to the kernel modules.

Here is a list presenting **10** code-updaters/ reprogramming with their highly conclusive features related to WSNs:

- Trickle [167] is an algorithm for propagating and maintaining code updates in WSNs. Borrowing techniques from the epidemic/gossip, scalable multicast, and wireless broadcast literature, Trickle uses a "polite gossip" policy, where motes periodically broadcast a code summary to local neighbors but stay quiet if they have recently heard a summary identical to theirs. When a mote hears an older summary than its own, it broadcasts an update. Instead of flooding a network with packets, the algorithm controls the send rate so each mote hears a small trickle of packets, just enough to stay up to date.
- MARWIS (Management ARchitecture for WIreless Sensor Networks) [89] supports common management tasks such as visualization, monitoring, (re)configuretion, updating and reprogramming. One of the main features of MARWIS is its hierarchical architecture. The applications running on the sensor nodes or net work properties can be reconfigured using the user in terface. Furthermore, updating and reprogramming the sensor nodes is a very important issue. In large WSNs manual execution of this task is unfeasible, and a

mechanism to handle it automatically and dynamically over the network is required. Both the OS and applica tions must be updated, either fully or partially.

- Multihop Over-the-Air Programming (MOAP) [168] is a code distribution mechanism specifically targeted for Mica-2 Motes. These are the following implementation choices for MOAP: Ripple dissemination protocol, Unicast retransmission policy and Sliding Window for segment management. The current version of MOAP has been successfully used to repeatedly reprogram motes up to four hops away from the base station, using code images of various sizes, ranging from 600 up to 30K bytes.
- FlexCup [169] is a flexible and efficient code update mechanism for WSNs that enable on fly reinstallation of software components in TinyOS-based nodes. FlexCup is an application that consists of a compiler-extension (FlexCup-Analyzer), a middleware component (code distribution algorithm), a standalone operating system (FlexCup-Linker), and a kernel component (FlexCup-Bootloader). It is able to reconfigure exchange or reinstall parts of an application. It has two phases: Code generation phase and linking phase.
- Zephyr [166] is a multi-hop incremental reprogramming protocol. It reduces the delta size by using application-level modifications to mitigate the effects of function shifts. Then it compares the binary images at the byte-level with a novel method to create small delta that is then sent over the wireless network to all the nodes.
- Deluge [170] is an epidemic protocol and operates as a state machine where each node follows a set of strictly local rules to achieve a desired global behavior: the quick, reliable dissemination of large data objects to many nodes. Deluge considers many subtle issues to achieve high performance. The first is its density-aware capability, where redundant advertisement and request messages are suppressed to minimize contention. Second, Deluge's three-phase handshaking protocol helps ensure that a bidirectional link exists before transferring data. Third, Deluge dynamically adjusts the rate of advertisements to allow quick propagation when needed while consuming few resources in the steady state. Fourth, Deluge attempts to minimize the set of nodes concurrently broadcasting data within a given cell. Finally, Deluge emphasizes the use of spatial multiplexing to allow parallel transfers of data.
- Stream [171] is a sensor network reprogramming protocol that significantly reduces the number of bytes to be transmitted over the wireless medium for reprogramming. It addresses a fundamental problem in all existing network reprogramming protocols, whereby the application image together with the reprogramming protocol image is transferred. Stream pre-installs the reprogramming protocol image in a node and transfers the application image with a small addition. Consequently, it reduces the reprogramming time, the number of bytes transferred, the energy expended, and the usage of program memory. Stream is implemented on TinyOS for the Mica-2 sensor node.
- Hermes [172] is a multi-hop incremental reprogramming protocol. It reduces the delta by using techniques to mitigate the effects of function and global variable shifts caused by the software modifications. Then it compares the binary images at the byte level with a method to create small delta that needs to be sent over the air to all the nodes.
- FIGARO [173] is a programming model supported by an efficient run-time system and distributed protocols, collectively enabling an unprecedented fine-grained control over what is being reconfigured, and where. Using FIGARO, the programmer can deal explicitly with component dependencies and version constraints.
- MNP [174] is a multi-hop network reprogramming service designed for Mica-2/XSM motes. To reduce the problem of collision and hidden terminal problem it implements a sender selection algorithm that attempts to guarantee that in a neighborhood there is at most one source transmitting the program at a time. Further, this sender selection is greedy in that it tries to select the sender that is expected to have the most impact. It also uses pipelining to enable fast data propagation. MNP is energy efficient because it reduces the active radio time of a sensor node by putting the node into "sleep" state when its neighbors are transmitting a segment that is not of interest.

8. Network Monitoring Tools for Wireless Sensor Networks

WSNs are typically composed of low cost tiny hardware devices and tend to be unreliable, with failures a common phenomenon. Accurate knowledge of network health status, including nodes and links of each type, is critical for correctly configuring applications on really deployed WSN and/or WSN testbeds and for interpreting the data collected from them.

Here is a list presenting **8** network monitoring tools with their summarized features related to WSNs:

- Memento [175] is a failure detection system that requires nodes to periodically send heartbeats to the so called observer node. Those heartbeats are not directly forwarded to the sink node, but are aggregated in form of a bitmask (*i.e.* bitwise OR operation). The

observer node is sweeping its bitmask every sweep interval and will forward the bitmask with the node missing during the next sweep interval if the node fails sending a heartbeat in between. Hence the information of the missing node is disseminated every sweep interval by one hop, eventually arriving at the sink. Memento is not making use of acknowledgements and proactively sends multiple heartbeats every sweep interval, whereas this number is estimated based on the link's estimated worst-case performance and the targeted false positive rate.

- NUCLEUS [157] is one of the network management systems for data-gathering application of WSN.
- DiMo [176] is a distributed and scalable solution for node and topology monitoring, especially designed for use with event-triggered WSNs. The monitoring is done by so called observer nodes that monitor whether the target node has checked in by sending a heartbeat within a certain monitoring time.
- MARWIS (Management Architecture for heterogeneous Wireless Sensor Networks) [89] supports common management tasks such as visualization, monitoring, (re)configuration, updating and reprogramming. The status information about every sensor node is monitored and displayed. This includes hardware features (micro-controller, memory, transceiver), software details (operating system versions, protocols, applications), dynamic properties (battery, free memory) and, if available, geographical position information.
- Sympathy [142] is a tool for detecting and debugging failures in pre- and post-deployment sensor networks, especially designed for data gathering applications. The nodes send periodic heartbeats to the sink that combines this information with passively gathered data to detect failures. For the failure detection, the sink requires receiving at least one heartbeat from the node every so called sweep interval, *i.e.* its lacking indicates a node failure. Direct-Heartbeat performs poorly in practice without adaptation to wireless packet losses. To meet a desired false positive rate, the rate of heartbeats has to be increased also increasing the communication cost.
- HERMES [172] is a lightweight framework and prototype tool that provides fine-grained visibility and control of a sensor node's software at run-time. HERMES's architecture is based on the notion of interposition, which enables it to provide these properties in a minimally intrusive manner, without requiring any modification to software applications being observed and controlled. HERMES provides a general, extensible, and easy-to-use framework for specifying which software components to observe and control as well as when and how this observation and control is done.
- LiveNet [177] is a set of tools and techniques for reconstructing complex dynamics of live sensor network deployments. LiveNet is based on the use of passive sniffers co-deployed with the network. Sniffer nodes can be temporary or permanent, fixed or mobile, and wired or un-tethered. Sniffers record traces of all packet activity observed on the radio channel. Traces from multiple sniffers are merged into a single trace to provide a global picture of the network's behavior. The merged trace is then subject to a series of analyses to study application behavior, data rates, network topology, and routing protocol dynamics.
- Chowkidar [145] is a reliable and scalable health monitoring protocol for wireless sensor network testbeds. It provides accurate and efficient network health monitoring in WSNs. The Chowkidar protocol is initiated upon demand and adapts the well-known problem of message-passing rooted spanning tree construction and its use in propagation of information with feedback (PIF) for the case of a WSN; that is, it does not involve ongoing maintenance, and it terminates with accurate results, including detection of failure and restart during the monitoring process. Chowkidar is distinguished from others in two important ways. Given the resource constraints of WSNs, it is message-efficient in that it uses only a few messages per node. Also, it tolerates ongoing node and link failure and node restart, in contrast to requiring that faults stop during convergence.

9. Educt of this Exploratory Study

Simulation tools are widely used for the purpose of exploratory analysis in validating algorithms and protocols due to their rapid prototyping and tackling large scale systems. However, even the best simulator is still not able to simulate real wireless communication environments in terms of completeness, complexity, accuracy and authenticity. Researchers use emulators of WSNs to selectively track whether their applications have executed as intended. These emulators simulate the hardware environments to facilitate the development and checking software applications. The emulator approach is quite laborious since extensive prior profiling is required. Taking these drawbacks of simulators and emulators into account, using WSN testbeds to evaluate algorithms and protocols of WSNs is essentially necessary before applying them into real world applications. The other experimental tools are also necessary before or after real deployment during whole life of real or virtual

WSN. The objective of this study has clearly brought forth important findings that are very useful for researchers involve in any level of WSN experiments to find an appropriate tool. In real, efforts are on synchronic aspects for beginners than presenting highly technical aspects.

10. Acknowledgements

The full credit of this work is dedicated to all researchers whose research content or contribution is used in any form. Full citation is mentioned in reference section.

11. References

[1] Upal, Blog Post at 06:17A.M., August 24, 2009. http://ai1st.blogspot.com/

[2] A. K. Dwivedi, V. K. Patle and O. P. Vyas, "Investigation on Effectiveness of Simulation Results for Wireless Sensor Networks," *Information Processing and Management*, Vol. 70, 2010, pp. 202-208.

[3] Distributed Computing Group, "Sinalgo-Simulator for Network Algorithms." http://disco.ethz.ch/projects/sinalgo/tutorial/tuti.html

[4] J. Eriksson, "Detailed Simulation of Heterogeneous Wireless Sensor Networks," Dissertation for Licentiate of Philosophy in Computer Science, Uppsala University, Sweden, April 2009.

[5] L. Shu, M. Hauswirth, Y. Zhang, S. Mao, N. Xiong and J. Chen, "NetTopo: A Framework of Simulation and Visualization for Wireless Sensor Networks," *ACM/Springer Mobile Networks and Applications* (*MONET*), Vol. 13, No. 3-4, 2008, pp. 306-322.

[6] Network Simulator. http://www.isi.edu/nsnam/ns/index.html

[7] I. Downard, "Simulating Sensor Networks in NS-2," Naval Research Laboratory Formal Report 5522, April 2004.

[8] Mannasim Simulator. http://www.mannasim.dcc.ufmg.br

[9] P. Levis, N. Lee, M. Welsh and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, 2003.

[10] UC Berkeley TOSSIM. www.cs.berkeley.edu/~pal/research/tossim.html

[11] L. F. Perrone and D. M. Nicol, "A Scalable Simulator for TinyOS Applications," *Proceedings of the 2002 Winter Simulation Conference*, Vol. 1, San Diego, 2002, pp. 679-687.

[12] E. Perla, A. Ó Catháin, R. S. Carbajo, M. Huggard and C. M. Goldrick, "PowerTOSSIM Z: Realistic Energy Modelling for Wireless Sensor Network Environments," *Proceedings of the 3nd ACM workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, Vancouver, 2008, pp. 35-42.

[13] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. S. Baras, and M. Karir, "ATEMU: A Fine-Grained Sensor Network Simulator," *Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, City, 7 October 2004, pp. 145-152.

[14] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," *Proceedings of the 31st IEEE Conference on Local Computer Networks*, 14-16 November 2006, pp. 641-648. doi:10.1109/LCN.2006.322172

[15] X. Zeng, R. Bagrodia and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," *Proceedings of the 12th Workshop on Parallel and Distributed Simulation* (*PADS*), 29 May 1998.

[16] QualNet Simulator by Scalable Network Technologies. http://www.scalable-networks.com/products/qualnet/

[17] G. Chen, J. Branch, M. J. Pflug, L. Zhu and B. Szymanski, "SENSE: A Sensor Network Simulator," In: B. K. Szymanski and B. Yener, Eds., *Advances in Pervasive Computing and Networking*, Springer, New York, 2004, pp. 249-267.

[18] P. Baldwin, S. Kohli, E. A. Lee, X. Liu and Y. Zhao, "VisualSense: Visual Modeling for Wireless and Sensor Network Systems," Technical Memorandum UCB/ERL M05/25, University of California, July 2005. http://ptolemy.eecs.berkeley.edu/papers/05/visualsense/

[19] AlgoSenSim. http://tcs.unige.ch/doku.php/algosensim

[20] Georgia Tech Network Simulator (GTNetS). http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/

[21] OMNeT++ simulation system. http://www.omnetpp.org/

[22] A. Varga, "The OMNeT++ Discrete Event Simulation System," *Proceedings of the European Simulation Multiconference (ESM'2001)*, Prague, 6-9 June 2001.

[23] D. Pediaditakis, S. H. Mohajerani and A. Boulis, "Poster Abstract: Castalia: the Difference of Accurate Simulation in WSN," *Proceedings of the 4th European Conference on Wireless Sensor Networks* (*EWSN*), Delft, 29-31 January 2007. http://castalia.npc.nicta.com.au/index.php

[24] A. Sobeih, M. Viswanathan, D. Marinov and J. C. Hou, "J-Sim: An Integrated Environment for Simulation and Model Checking of Network Protocols," *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, 2007, pp. 1-6. http://www.j-sim.zcu.cz/

[25] JiST/SWANS. http://jist.ece.cornell.edu/

[26] JiST/SWANS++. http://www.aqualab.cs.northwestern.edu/projects/swans++/

[27] B. L. Titzer, D. K. Lee and J. Palsberg, "Avrora: Scalable Sensor Network Simulation with Precise Timing," *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, USA, April 2005.

[28] T. W. Carley, "Sidh: A Wireless Sensor Network Simulator," ISR Technical Report, Department of Electrical & Computer Engineering University of Maryland at College

Park, 2004. http://drum.lib.umd.edu/bitstream/1903/6565/1/TR_2005-88.pdf

[29] J. Prowler, "Institute for Software Integrated Systems," Vanderbilt University, Nashville, Tennessee, 2002. http://www.isisvanderbilt.edu/Projects/nest/prowler

[30] J. Prowler, "Institute for Software Integrated Systems," Vanderbilt University, Nashville, Tennessee. http://w3.isis.vanderbilt.edu/Projects/nest/jprowler/index.html

[31] A. Cerpa and D. Braginsky, "LecsSim-Wireless Network Simulator," 2002. http://sourceforge.net/projects/lecssim

[32] OPNET Modeller, OPNET Technologies Inc. http://www.opnet.com

[33] S. Sundresh, W. Kim and G. Agha, "SENS: A Sensor, Environment and Network Simulator," *Proceedings of the* 37*th Annual Symposium on Simulation*, Virginia, USA, 2004, p. 221. doi:10.1109/SIMSYM.2004.1299486

[34] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac and D. Estrin, "Emstar: A Software Environment for Developing and Deploying Heterogeneous Sensor-Actuator Networks," *ACM Transactions on Sensor Networks*, Vol. 3, No. 3, August 2007, pp. 13-es. doi:10.1145/1267060.1267061

[35] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil and T. Schoellhammer, "A System for Simulation, Emulation and Development of Heterogeneous Sensor Networks," *Proceedings of the* 2*nd ACM Conference on Embedded Networked Sensor Systems* (*SenSys'*04), Baltimore, November 2004.

[36] M. Varshney, D. Xu, M. Srivastava and R. Bagrodia, "SenQ: A Scalable Simulation and Emulation Environment for Sensor Networks," *Proceedings of the* 6*th International Conference on Information Processing in Sensor Networks*, Cambridge, 25-27 April 2007, pp. 196-205.

[37] O. C. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischof and N. Valtchanov, "SIDnet-SWANS: A Simulator and Integrated Development Platform for Sensor Networks Applications," *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, New York, 2008, pp. 385-386.

[38] S. Park, A. Savvides and M. B. Srivastava, "SensorSim: A Simulation Framework for Sensor Networks," *Proceedings of the* 3*rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Boston, August 2000, pp. 104-111.

[39] S. Fekete, P. Kroller, A. Fischer, S. Pfisterer and D. Braunschweig, "Shawn: The Fast, Highly Customizable Sensor Network Simulator," *Proceedings of the* 4*th International Conference on Networked Sensing Systems*, Brunswick, 2007, doi:10.1109/INSS.2007.4297441

[40] SSFNet. http://www.ssfnet.org/

[41] P. M. Wightman and M. A. Labrador, "Atarraya: A Simulation Tool to Teach and Research Topology Control Algorithms for Wireless Sensor Networks," *Proceedings of the SIMUTools*, Rome, 2009.

[42] L. Shu, M. Hauswirth, Y. Zhang, S. Mao, N. Xiong and J. Chen, "NetTopo: A Framework of Simulation and Visualization for Wireless Sensor Networks," *Proceedings of the ACM/Springer Mobile Networks and Applications*, 2009.

[43] V. Ramesh, "WiSeNet: A Software Simulator for Wireless Sensor Net- work Applications," 2005. http://wisenetsim.sourceforge.net/

[44] Y. Wen, S. Gurun, N. Chohan, R. Wolski and C. Krintz, "SimGate: Full-System, Cycle-Close Simulation of the Stargate Sensor Network Intermediate Node," *Proceedings of the International Conference on Architectures, Modeling and Simulation*, Greece, July 2006.

[45] C.-N. Xu, Z Lei, Y.-J. Xu and X.-W. Li, "Simsync: A Time Synchronization Simulator for Sensor Networks," *Acta Automatica Sinica*, Vol. 32, No. 6, 2006, pp. 1008-1014

[46] SNetSim. Naval Science and Engineering Institute, Istanbul, key. http://www.dho.edu.tr/enstitunet/snetsim/index.htm

[47] S. Yi, H. Min, Y. Cho and J. Hong, "SensorMaker: A Wireless Sensor Network Simulator for Scalable and Fine-Grained Instrumentation," *Computational Science and Its Applications*, Berlin, 2008.

[48] F. Gómez and G. Martínez, "TRMSim-WSN: Trust and Reputation Models Simulator for Wireless Sensor Networks," *Proceedings of the IEEE International Conference on Communications*, Germany, June 2009.

[49] D. Weber, J. Glaser and S. Mahlknecht, "Discrete Event Simulation Framework for Power Aware Wireless Sensor Networks," *Proceedings of the* 5*th IEEE International Conference on Industrial Informatics*, Vol. 1, July 2007, pp. 335-340. doi:10.1109/INDIN.2007.4384779

[50] J. Barbancho, F. J. Molina, C. León, J. Ropero and A. Barbancho, "OLIMPO: An Ad-hoc Wireless Sensor Network Simulator for Optimal SCADA-Applications," *Proceedings of the Communication Systems and Networks*, Spain, 2004.

[51] Y. Wen, R. Wolski and G. Moore, "DiSenS: Scalable Distributed Sensor Network Simulation," UCSB Computer Science Technical Report, University of California, 2005.

[52] H. B. Lim, B. Wang, C. F. Phull and A. D. Ma, "WISDOM: Simulation Framework for Middleware Services in Wireless Sensor Networks," *Proceeding of the 5th IEEE Consumer Communications and Networking Conference*, 2008.

[53] J. N. Al-Karaki and G. A. Al-Mashaqbeh, "SENSORIA: A New Simulation Platform for Wireless Sensor Networks," *Proceedings of the International Conference on Sensor Technologies and Applications*, 2007, pp. 424-429,

[54] K. Sha, J. Du and W. Shi, "Capricorn: A Large Scale Wireless Sensor Network Simulator," Technical Report MIST-TR-2004-001, Wayne State University, January

2004.

[55] B. C. Mochocki and G. R. Madey, "H-MAS: A Heterogeneous, Mobile, Ad-hoc Sensor-Network Simulation Environment," *Proceedings of the 7th Annual SWARM Users/Researchers Conference*, Notre Dame, Indiana, April 2003.

[56] Stargate Simulator (starsim) and Mote Simulator (motesim). http://www.cs.ucsb.edu/~wenye/

[57] L. Chuan-wen, G. Yu, L. Fang-fang, X. Jia and Y. Ge, "SNSim: Study and Implementation of a Wireless Sensor Network Simulator," *Journal of Chinese Computer Systems*, No. 6, 2010, pp. 1025-1029.

[58] S. N. Sinha, Z. Chaczko and R. Klempous, "SNIPER: A Wireless Sensor Network Simulator," *Proceedings of the EUROCAST*, 2009, pp. 913-920.

[59] C. Kelly, V. Ekanayake and R. Manohar, "SNAP: A Sensor Network Asynchronous Processor," *Proceedings of the 9th ACM Int. Symposium on Asynchronous Circuits and Systems*, Ithaca, 12-15 May, 2003, pp. 24-33. doi:10.1109/ASYNC.2003.1199163

[60] SimPy Documentation. By K. ler. http://simpy.sourceforge.net/discuss.htm

[61] D. Watson and M. Nesterenko, "Mule: Hybrid Simulator for Testing and Debugging Wireless Sensor Networks," *Proceedings of the Second International Workshop on Sensor and Actor Network Protocols and Applications*, Boston, 2004, pp. 67-71.

[62] A. Boulis, A. Fehnker, M. Fruth and A. McIver, "CaVi – Simulation and Model Checking for Wireless Sensor Networks," *Proceedings of the Quantitative Evaluation of Systems* (*QEST*), 2008.

[63] P. Baldwin, S. Kohli, E. A. Lee, X. Liu and Y. Zhao, "Modeling of Sensor Nets in Ptolemy II," *Proceedings of the Information Processing in Sensor Networks* (*IPSN*), 2004, pp. 359-368.

[64] Maple Simulator. http://www.maplesoft.com/

[65] WISENES Simulator. http://www.tkt.cs.tut.fi/research/daci/daci_wsn_wisenes.html

[66] A. Fraboulet, G. Chelius and E. Fleury, "Worldsens: Development and Prototyping Tools for Application Specific Wireless Sensors Networks," *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, USA, April 2007.

[67] A. Suri, "Simulation Study for Wireless Sensor Networks and Load Sharing Routing Protocol to Increase Network Life and Connectivity," Master's Thesis, Department of Computer Science, Louisiana State University, December 2005.

[68] M. Karagiannis, I. Chatzigiannakis and J. Rolim, "WSNGE: A Platform for Simulating Complex Wireless Sensor Networks Supporting Rich Network Visualization and Online Interactivity," *Proceedings of the Spring Simulation Multiconference* (*SpringSim*' 09), Article 35, 2009.

[69] F. Menichelli and M. Olivieri, "TikTak: A Scalable Simulator of Wireless Sensor Networks Including Hardware/Software Interaction," *Wireless Sensor Network*, November 2010, pp. 815-822. http://www.SciRP.org/journal/wsn

[70] H. Wu, Q. Luo, P. Zheng and L. M. Ni, "VMNet: Realistic Emulation of Wireless Sensor Networks," *IEEE Transaction on Parallel Distributed System*, Vol. 18, No. 2, 2007, pp. 277-288. doi:10.1109/TPDS.2007.33

[71] VMNET: A WSN Emulator, 2004. http://www.cse.ust.hk/vmnet/

[72] T. Maret, R. Kummer, P. Kropf and J. F. Wagen, "Freemote Emulator: A Lightweight and Visual Java Emulator for WSN. Wired/Wireless Internet Communications," *Springer LNCS*, Vol. 5031, 2008, pp. 92-103.

[73] C. Park and P. H. Chou, "EmPro: an Environment/Energy Emulation and Profiling Platform for Wireless Sensor Networks," *Proceedings of the IEEE SECON*, 2006, pp. 158-167.

[74] OCTAVEX Wireless Sensor work. http://www.octavetech.com/solutions/octavex.html

[75] UbiSec & Sens. http://www.ist-ubisecsens.org/

[76] Surge Network Viewer. By Crossbow Technology Inc. http://www.hoskin.qc.ca/uploadpdf/Instrumentation/divers/CrossBow/divers_Surge%20Network%20Viewer_4271286a0135f.pdf

[77] J. Eriksson, A. Dunkels, N. Finne, F. Österlind and T. Voigt, "Mspsim—An Extensible Simulator for Msp430-Equipped Sensor Boards," *Proceedings of the European Conference on Wireless Sensor Networks*, Poster/Demo session, The Netherlands, January 2007.

[78] Q. Luo, L. M. Ni, B. He, *et al*., "MEADOWS: Modeling, Emulation, and Analysis of Data of Wireless Sensor Networks," *Proceedings of the First Workshop on Data Management for Sensor Networks* (*DMSN* 2004), Toronto, 2004.

[79] B. Parbat, A. K. Dwivedi and O. P. Vyas, "Data Visualization Tools for WSNs: A Glimpse," *International Journal of Computer Applications*, Vol. 2, No. 1, 2010, pp. 14-20.

[80] C. Buschmann, D. Pfisterer, S. Fischer, S. P. Fekete and A. Kröller, "SpyGlass: A Wireless Sensor Network Visualizer," *Acm Sigbed Review*, Vol. 2, No. 1, 2005, pp. 1-6. doi:10.1145/1121782.1121784

[81] M. Tuton, "MOTEVIEW: A Sensor Network Monitoring and Management Tool," *Proceedings of Second IEEE Workshop on Embedded Networked Sensors* (*EmNetS*-II), 2005, pp. 11-18.

[82] J. Pinto, A. Sousa, P. Lebres, G. M. Gonçalves and J. Sousa, "Monsense—Application for Deployment, Monitoring and Control of Wireless Sensor Networks," *Poster in ACM RealWSN*'06, 2006.

[83] R. Jurdak, A. G. Ruzzelli, A. Barbirato and S. Boivineau, "Octopus: Monitoring, Visualization, and Control of Sensor Networks," *Wireless Communications and Mobile Computing*, John Wiley & Sons, 2009.

[84] MSR Sense: MSR Networked Embedded Sensing Toolkit. http://research.microsoft.com/nec/msrsense/

[85] MoteIV Corporation. http://www.sentilla.com/

[86] Y. Yang, P. Xia, L. Huang, Q. Zhou, Y. Xu and X. Li, "SNAMP: A Multi-sniffer and Multi-view Visualization Platform for Wireless Sensor Networks," *Proceedings of the 1st IEEE Conference on Industrial Electronics and Applications*, Singapore, 24-26 May 2006, pp. 1-4. doi:10.1109/ICIEA.2006.257222

[87] Meshnetics. http://www.meshnetics.com/press_releases/ ics. http://www.meshnetics.com/press_releases/MeshNetics_SensiLink_Press_Release_25Jun06.pdf

[88] D. Davcev and S. Gancev, "Monitoring of environment by energy efficient usage of Wireless Sensor Networks," *Information Technologies in Environmental Engineering*, Berlin, 2009, pp. 229-237.

[89] G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka and S. Morgenthaler, "MARWIS: A Management Architecture for Heterogeneous Wireless Sensor Networks," *Proceedings of the 6th International Conference on Wired/Wireless Internet Communications*, Finland, 2008, pp. 177-188.

[90] Oscilloscope. From TinyOS Community Forum, "TinyOS: An open-source OS for the networked sensor regime". http://www.tinyos.net

[91] K. Aberer, M. Hauswirth and A. Salehi, "Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks," *Proceedings of the 8th International Conference on Mobile Data Management*, Mannheim, Germany, May 2007. doi:10.1109/MDM.2007.36

[92] J. A. Castillo, A. M. Ortiz, V. López, T. Olivares and L. Orozco-Barbosa, "WiseObserver: A Real Experience with Wireless Sensor Networks," *Proceedings of the 3nd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired networks*, New York, USA, October 2008, pp. 23-26.

[93] P. Andreou, D. Zeinalipour-Yazti, A. Pamboris, P. K. Chrysanthis and G. Samaras, "Optimized Query Routing Trees for Wireless Sensor Networks," *Information Systems*, 2010.

[94] Crossbow Technology Inc. http://www.xbow.com

[95] G. Werner-Allen, P. Swieskowski and M. Welsh, "MoteLab: A Wireless Sensor Network Testbed," *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, Article 68, April 2005.

[96] Tutornet Testbed. http://enl.usc.edu/projects/tutornet/index.html

[97] WUSTL Wireless Sensor Network Testbed. http://www.cs.wustl.edu/wsn/index.php?title=Testbed

[98] R. Murty, G. Mainland, I. Rose, A. R.Chowdhury, A. Gosain, J. Bers and M. Welsh, "CitySense: A Vision for an Urban-Scale Wireless Networking Testbed," *Proceedings of the IEEE International Conference on Technologies for Homeland Security*, Waltham, May 2008. doi:10.1109/THS.2008.4534518

[99] E. Ertin, A. Arora, R. Ramnath, *et al*., "Kansei: A Testbed for Sensing at Scale," *Proceedings of the 4th Symposium on Information Processing in Sensor Networks*, USA, April 2006.

[100] MistLab Testbed. http://mistlab.csail.mit.edu/

[101] M. Ott, I. Seskar, R. Siracusa and M. Singh, "ORBIT Testbed Software Architecture: Supporting Experiments as a Service," *Proceedings of the IEEE Tridentcom*, Italy, February 2005.

[102] D. Johnson, D. Flickinger, T. Stack, R. Ricci, L. Stoller, R. Fish, K. Webb, M. Minor and J. Lepreau, "Emulab's Wireless Sensor Net Testbed: True Mobility, Location Precision, and Remote Access," *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, 2005.

[103] WISEBED—Wireless Sensor Network Testbed. http://www.wisebed.eu/

[104] J. Albesa, R. Casas, M. T. Penella and M. Gasulla, "REALnet: An Environmental WSN Testbed," *Proceedings of the International Conference on Sensor Technologies and Applications*, 2007, pp. 502-507.

[105] K. Iwanicki, A. Gaba and M. V. Steen, "KonTest: A Wireless Sensor Network Testbed at Vrije Universiteit Amsterdam," Technical Report IR-CS-045, Vrije Universiteit Amsterdam, August 2008.

[106] A. Hergenr¨oder, J. Horneber, D. Meier, *et al*., "Demo Abstract: Distributed Energy Measurements in Wireless Sensor Networks," *Proceedings of the SenSys*'09, USA, November 2009. http://doc.tm.uka.de/SANDbed_Poster.pdf

[107] H. Alzaid and S. Abanmi, "A Wireless Sensor Networks Test-bed for the Wormhole Attack," *International Journal of Digital Content Technology and its Applications*, Vol. 3, No. 3, 2009, pp. 19-27.

[108] U. Kumar, A. Ranjan, V. Jalan, *et al*., "CENSE: A Modular Sensor Network Testbed," *Proceedings of the National Conference on Embedded Systems*, February 2006.

[109] M. J. Murillo and J. A. Slipp, "Application of WINTeR Industrial Testbed to the Analysis of Closed-Loop Control Systems in Wireless Sensor Networks," *Demo Accepted in the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks*, April 2009.

[110] C. Kejie, G. Xiang, H. Xianying, C. Jiming and S. Youxian, "Design and Develop of the WSN Testbed: NESC-Testbed," *China Science Paper Online*, December 2008.

[111] Y. L. Huang, J. D. Tygar, H. Y. Lin, *et al*., "SWOON: A Testbed for Secure Wireless Overlay Networks," *Proceedings of the Workshop on Cyber Security Experimentation and Test (CSET '08)*, 2008.

[112] M. Doddavenkatappa, M. C. Chan and A. L. Ananda, "Indriya: A Low Cost, 3DWireless Sensor Network Testbed," Developed at School of Computing, National University of Singapore, Singapore, 2009. http://indriya.comp.nus.edu.sg/motelab/html/index.php

[113] Clarity-Testbed. http://ercim-news.ercim.eu/en76/special/ ty-Testbed. http://ercim-news.ercim.eu/en76/special/the-clarity-ubiquitous-robotic-testbed

[114] Imote2 Sensor Network Testbed. http://www.ee.washington.edu/research/nsl/Imote/

[115] J. P. Sheu, C. J. Chang, C. Y. Sun and W. K. Hu, "WSNTB: A Testbed for Heterogeneous Wireless Sensor Networks," *Proceedings of the 1st IEEE International*

Conference on Ubi-Media Computing, 2008. doi:10.1109/UMEDIA.2008.4570913

[116] V. Handziski, A. K¨opke, A. Willig, *et al*., "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks," *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing*, 2006, pp. 63-70.

[117] ENL Sensor Network Testbed. http://enl.usc.edu/projects/testbed/index.html

[118] A. Kanzaki, T. Hara, Y. Ishi, *et al*., "X-Sensor: A Sensor Network Testbed Integrating Multiple Networks," *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems*, 2009, pp. 1082-1087.

[119] E. Welsh, W. Fish and P. Frantz, "GNOMES: A Testbed for Low-Power Heterogeneous Wireless Sensor Networks," *Proceedings of the IEEE International Symposium on Circuits and Systems*, Thailand, 2003.

[120] M. Ramakrishnan and P. V. Ranjan, "PICSENSE–A Wireless Sensor Network Testbed," *International Journal of Recent Trends in Engineering*, Vol. 1, No. 4, May 2009, pp. 59-63.

[121] SOWNet WSN Testbed. http://www.sownet.nl/download/T301Web.pdf

[122] D. Sakamuri, "NetEye: A Wireless sensor Network Testbed," Thesis for Mater of Science Degree, Wayne State University, Detroit, Michigan, 2008.

[123] T. Dimitriou, J. Kolokouris and N. Zarokostas, "Sensenet: A Wireless Sensor Network Testbed," *Proceedings of the 10th ACM Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, 2007, pp. 143-150. doi:10.1145/1298126.1298153

[124] Omega and Motescope Testbed. http://www.millennium.berkeley.edu/sensornets/

[125] Sharesense Testbed. http://ru1.cti.gr/aeolus-workshop07/material/aeolus-workshop-07-talk13.pdf

[126] P. Dutta, J. Hui, J. Jeong, *et al*., "Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments," *Proceedings of the fifth International Conference on Information Processing in Sensor Networks*, 2006, pp. 407-415.

[127] sMote Testbed. University of California, Berkeley. http://www.cooperating-objects.eu/testbed-simulation/testbed-federation/testbed-directory/

[128] G. Mylonas, "Practical Aspects of CTI WSN Testbed," 2nd PROSENSE Meeting, Santorini, September 2008. www.prosense-project.eu/files/prosense.pdf

[129] A. Mateska, V. Atanasovski and L. Gavrilovska, "FEEIT WSN Testbed: Effective System for Providing Emergency Situations Prevention and Surveillance," In: A. Mateska, V. Atanasovski and L. Gavrilovska, Eds., *Application and Multidisciplinary Aspects of Wireless Sensor Networks*, Springer, London, 2011, pp. 245-256. doi:10.1007/978-1-84996-510-1_11

[130] Roulette. https://www.millennium.berkeley.edu/wiki/senslette. https://www.millennium.berkeley.edu/wiki/sensornets

[131] BigNet Sensor Network Testbed. http://www.issnip.unimelb.edu.au/project-item/bignet_testbed/bignet_sensor_network_testbed

[132] UCR Wireless Networking Research Testbed. University of California. http://networks.cs.ucr.edu/testbed/

[133] J. Sa Silva, R. Ruivo, T. Camilo, *et al*., "IP in wireless sensor networks Issues and lessons learnt," *Proceedings of the 3rd International Conference on Communication Systems Software and Middleware and Workshops*, 2008, pp. 496-502. doi:10.1109/COMSWA.2008.4554464

[134] Z. Ji, M. Marina, M. Varshney, *et al*., "WHYNET: A Hybrid Testbed for Large-Scale, Heterogeneous and Adaptive Wireless Networks," Technical Report CSD-TR060002, UCLA Computer Science Department, January 2006.

[135] CENS-Testbed. http://www.tinyos.net/ttx-02-2005/testbeds/ttx-testbed-panel-mlukac-v4.PPT

[136] SCADDS WSN Testbed. http://www.isi.edu/scadds/publications.html

[137] J. Hernstrom, "A Test-Bed Application for Analog Sensors Used in Wireless Sensor Networks: Student Paper," *Journal of Computing Sciences in Colleges*, Vol. 21, No. 4, 2006, pp. 184-189.

[138] N. Brent, P. Buonadonna, A. AuYoung, *et al*., "Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds," *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, Australia, May 2005.

[139] A. Tavakoli, "Wringer: A Debugging and Monitoring Framework for Wireless Sensor Networks," *Proceedings of the ACM SenSys Doctoral Colloquium*, 2007.

[140] J. Yang, M. L. Soffa, L. Selavo and K. Whitehouse, "Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks," *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, Sydney, 2007, pp. 189-203. doi:10.1145/1322263.1322282

[141] M. M. H. Khan, H. K. Le, H. Ahmadi, T. F. Abdelzaher and J. Han, "Dustminer: Troubleshooting Interactive Complexity Bugs in Sensor Networks," *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, Raleigh, 2008, pp. 99-112. doi:10.1145/1460412.1460423

[142] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler and D. Estrin, "Sympathy for the Sensor Network Debugger," *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, San Diego, 2005, pp. 255-267. doi:10.1145/1098918.1098946

[143] S. Guo, Z. Zhong and T. He, "FIND: Faulty Node Detection for Wireless Sensor Networks," *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, California, 2009, pp. 253-266. doi:10.1145/1644038.1644064

[144] I. Urteaga, K. Barnhart and Q. Han, "REDFLAG: A Run-timE, Distributed, Flexible, Lightweight, and Generic Fault Detection Service for Data-Driven Wireless Sensor Applications," *Pervasive and Mobile Computing*, Vol. 5, No. 5, 2009, pp. 432-446. doi:10.1016/j.pmcj.2009.08.001

[145] S. Bapat, W. Leal, T. Kwon, *et al*., "Chowkidar: Reliable and Scalable Health Monitoring for Wireless Sensor Network Testbeds," *ACM Transactions on Autonomous and Adaptive Systems* (*TAAS*), Vol. 4, No. 1, Article 3, January 2009.

[146] Y. Kwon, S. Sundresh, K. Mechitov and G. Agha, "ActorNet: An Actor Platform for Wireless Sensor Networks," Technical Report UIUCDCS-R-2005-2595, University of Illinois, Urbana-Champaign, 2005.

[147] R. K. Karmani and G. Agha, "Debugging Wireless Sensor Networks Using Mobile Actors," *Proceedings of the RTAS Poster Session*, 2008.

[148] M. Lodder, "Debugging Wireless Sensor Networks using a Global-State Perspective," Master's Thesis in Computer Science, Delft University of Technology, 2008.

[149] M. Lodder, G. P. Halkes and K. G. Langendoen, "A Global-State Perspective on Sensor Network Debugging," *Proceedings of the HotEmNets*'08, Charlottesville, Jun. 2008.

[150] Q. Cao, T. Abdelzaher, J. Stankovic, *et al*., "Declarative Tracepoints: A Programmable and Application Independent Debugging System for Wireless Sensor Networks," *Proceedings of the* 6*th ACM Conference on Embedded Network Sensor Systems*, Raleigh, 2008, pp. 85-98. doi:10.1145/1460412.1460422

[151] L. Luo, T. He, G. Zhou, *et al*., "Achieving Repeatability of Asynchronous Events in Wireless Sensor Networks with EnviroLog," *Proceedings of the IEEE INFOCOM*, 23-29 April 2006. doi:10.1109/INFOCOM.2006.114

[152] V. Krunic, E. Trumpler and R. Han, "NodeMD: Diagnosing Node-Level Faults in Remote Wireless Sensor Systems," *Proceedings of the* 5*th International Conference on Mobile Systems, Applications and Services*, San Juan, June 2007.

[153] P. D. K. Varma and V. Radha, "Prevention of Buffer Overflow Attacks Using Advanced Stackguard," *Proceedings of the International Conference on Advances in Communication, Network, and Computing* (*CNC*), Calicut, 4-5 October 2010.

[154] R. Sasnauskas, O. Landsiedel, M. H. Alizai, *et al*., "KleeNet: Discovering Insidious Interaction Bugs in Wireless Sensor Networks before Deployment," *Proceedings of the IPSN*'10, Stockholm, 12 April 2010.

[155] K. Whitehouse, G. Tolle, J. Taneja, *et al*., "Marionette: Using RPC for Interactive Development and Debugging of Wireless Embedded Networks," *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks* (*IPSN*), Nashville, 19-21 April 2006. doi:10.1109/IPSN.2006.243881

[156] K. Römer and J. Ma, "PDA: Passive Distributed Assertions for Sensor Networks," *Proceedings of the IPSN*'09, San Francisco, April 2009, pp. 337-348.

[157] G. Tolle and U. C. Berkeley, "Nucleus Network Management System." www.tinyos.net/ttx-02-2005/developments/Nucleus%20NMS.ppt

[158] T. Sookoor, T. Hnat, P. Hooimeijer, *et al*., "Macrodebugging: Global Views of Distributed Program Execution," *Proceedings of the SenSys*-09, Berkeley, 4-6 November 2009.

[159] M. M. H. Khan, L. Luo, C. Huang and T. F. Abdelzaher, "Snts: Sensor Network Troubleshooting Suite," *Proceedings of the* 3*rd IEEE International Conference DCOSS*, Springer LNCS, Vol. 4549, 2007, pp. 142-157.

[160] S. Gupta, R. Zheng and A. M. K. Cheng, "ANDES: an Anomaly Detection System for Wireless Sensor Networks," *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems*, 2007, pp. 1-9. doi:10.1109/MOBHOC.2007.4428636

[161] M. Woehrle, C. Plessl, R. Lim, *et al*., "EvAnT: Analysis and Checking of Event Traces for Wireless Sensor Networks," *Proceeding of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, 11-13 June 2008, pp. 201-208. doi:10.1109/SUTC.2008.24

[162] T. O'Donovan, C. J. Sreenan, N. Tsiftes, *et al*., "Poster Abstract: Storage-Centric Debugging of Performance Problems in Sensor Networks," *Proceedings of the* 7*th European Conference on Wireless Sensor Networks*, Coimbra, February 2010.

[163] A. D. Jong, "Evaluating Model-Based Diagnosis for Wireless Sensor Networks," Master's Thesis in Computer Science, Delft University of Technology, Delft, 2009.

[164] Z. Chen and K. G. Shin, "Post-Deployment Performance Debugging in Wireless Sensor Networks," *Proceedings of the* 30*th IEEE Real-Time Systems Symposium*, Washington DC, 2009, pp. 313-322. doi:10.1109/RTSS.2009.47

[165] Y. Wen, R. Wolski and S. Gurun, "S_2DB: A Novel Simulation Based Debugger for Sensor Network Appli- cations," *Proceedings of the EMSOFT*'06, Seoul, October 2006.

[166] R. K. Panta, S. Bagchi and S. P. Midkiff, "Zephyr: Efficient Incremental Reprogramming of Sensor Nodes using Function Call Indirections and Difference Computation," *Proceedings of the USENIX*, San Diego, 2009. http://www.usenix.org/events/usenix09/tech/full_papers/panta/panta_html

[167] P. Levis, N. Patel, D. Culler and S. Shenker, "Trickle: A Self Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," *Proceedings of the* 1*st conference on Symposium on Networked Systems Design and Implementation*, San Francisco, 2004.

[168] T. Stathopoulos, J. Heidemann and D. Estrin, "A Remote Code Update Mechanism for Wireless Sensor Networks," Technical Report CENS-TR-30, University of California, Los Angeles, November 2003.

[169] P. J. Marr´on, M. Gauger, A. Lachenmann, *et al*., "FlexCup: A Flexible and Efficient Code Update Mechanism for Sensor Networks," *Proceedings of the Third European Workshop on Wireless Sensor Networks*, Zurich, February 2006, pp. 212-227.

[170] J. W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," *Proceedings of the ACM SenSys*'04, Maryland, November 2004, pp. 81-94.

[171] R. Panta, I. Khalil and S. Bagchi, "Stream: Low Overhead Wireless Reprogramming for Sensor Networks," *Proceedings of the IEEE Infocom*, Anchorage, 2007, pp. 928-936.

[172] R. Panta and S. Bagchi, "Hermes: Fast and Energy Efficient Incremental Code Updates for Wireless Sensor Networks," *Proceedings of the IEEE Infocom*, 2009.

[173] L. Mottola, G. P. Picco and A. A. Sheikh, "Figaro: Fine-grained Software Reconfiguration for Wireless Sensor Networks," *Proceedings of the EWSN*, Bologna, 2008.

[174] S. S. Kulkarni and L. Wang, "Mnp: Multihop Network Reprogramming Service for Sensor Networks," *Proceedings of the* 25*th International Conference on Distributed Computing Systems* (*ICDCS*), Columbus, 2005.

[175] S. Rost and H. Balakrishnan, "Memento: A Health Monitoring System for Wireless Sensor Networks," *Proceedings of the* 3*rd IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks* (*IEEE SECON*), Reston, 2006.

[176] A. Meier, M. Motani, H. Siquan, *et al*., "DiMo: Distributed Node Monitoring in Wireless Sensor Networks," *Proceedings of the MSWiM*'08, Vancouver, October 2008.

[177] B. Chen, G. Peterson, G. Mainland, *et al*., "LiveNet: Using Passive Monitoring to Reconstruct Sensor Network Dynamics," *Proceedings of the* 4*th International Conference on IEEE/ACM Distributed Computing in Sensor Systems*, Santorini Island, June 2008.

