2012 IACSIT Hong Kong Conferences
IPCSIT vol. 30 (2012) © (2012) IACSIT Press, Singapore

WiEmu: The Design and Implementation of a Flexible Agent-Based Scalable Network Emulator for Wireless Sensor Networks

Sherin M.Youssef, M. Abou El-Nasr and Mohamed Aslan [+]

Department of Computer Engineering, Arab Academy for Science and Technology

Abstract. The accuracy of the simulation results which are calculated from the current network simulators is suspicious when compared to real network's test-bed. Thus there is a great need to a low cost solution that gives accurate results and takes into consideration the special needs of wireless sensor networks. In this paper, we try to overcome the limitations of network simulation by the incorporation of network emulation. The proposed agent-based wireless sensor networks emulator WiEmu achieves high fidelity results by emulating the hardware architecture of the wireless sensor motes. More specifically it has the ability of running heterogeneous wireless sensor network: software, operating systems, and applications transparently unmodified. Experiments show that the proposed emulator accurately models the behavior of the sensor mote. Results also indicate that the proposed emulator performs well especially when the motes tend to sleep most of time.

Keywords: WiEmu, wireless sensor network, network emulation, network simulation, mobile agent.

1. Introduction

Wireless Sensor Networks have recently received a great attention from researchers all over the world. One important step prior the actual deployment of wireless sensor networks is the evaluation of the network. However, the use of real test-beds to evaluate the network is not appropriate due to the high cost of implementation and scalability issues as the network size increases. It is also a time and effort consuming process that would turn the process of network evaluation into a process of maintenance.

Fortunately, there exist two feasible approaches for evaluating the networks: Simulation and Emulation. Most of the research conducted in the area of wireless sensor networks is experimented on network simulators, indeed. Simulation tends to ignore some details; hence it fails to accurately model the behaviour of the motes. Thus network simulation is not appropriate for the evaluation of wireless sensor networks when high fidelity is a requirement. An example is the Wireless Multimedia Sensor Networks WMSN, a special type of wireless sensor networks that incorporates in-network multimedia processing. Traditional network simulators will fail to measure the actual power consumed in the processing of multimedia by the sensor nodes.

A study [1] compared the results of different network simulators, has shown that the results conducted by the different simulators are quite different. The study also found that at low traffic rates the results of simulation matches that of test-beds. However it is not the case at high traffic rates, where the results were dramatically different. The study also showed that the results become incomparable when mobility is emulated.

On the other hand, approaches including emulation show promising solution for high fidelity evaluation of wireless sensor networks. Emulators can either be cycle-level accurate or operating system level accurate.

[+] Corresponding author. Tel.: + 203-5622366; fax: +203-5610950.
E-mail address: *maslan@aast.edu.*

Moreover, emulators trade scalability with high fidelity. Unlike simulators that can be highly scalable but lacks the high fidelity.

In this paper we provide a detailed look at the design of WiEmu available at [2] a flexible and open-source emulator for wireless sensor networks. The main objective of WiEmu was to develop a precise cycle-accurate, flexible and scalable software emulator for heterogeneous wireless sensor networks. A software emulator that is able to run the motes' software transparently unmodified as if it runs on real mote's hardware.

The rest of this paper is organized as follows: section 2 provides an overview of the related work, section 3 provides detailed view on the design and implementation of WiEmu, section 4 describes the validation and experimental results conducted, and finally section 5 provides a conclusion and an outline for possible future work.

2. Related Work

2.1. Atemu

ATEMU [5] is an instruction-level emulator for wireless sensor networks. ATEMU emulates the sensor node instruction-by-instruction with cycle accuracy and simulates the wireless communication between them. It was designed to be able to emulate heterogeneous wireless sensor networks. According to [5] it only supports MICA2 motes. It emulates the "Atmel ATmega 128L" micro-controller, the radio chip "CC1000", LEDs, ADC, EEPROM and Timers. ATEMU is able to run TinyOS built for MICA2 nodes unmodified as it runs on real MICA2 platform.

ATEMU uses configuration files written in XML to specify the specifications for a wireless sensor network where, the hardware configuration, software image and physical location of each node can be specified. Event tags can also be specified which allows specific events to occur explicitly during the emulation time.

ATEMU also includes XATDB a graphical user interface for ATEMU as well as a debugger for the code running on the emulated environment.

As for the scalability, the memory consumption of ATEMU is increasing linearly with the number of nodes while the emulation time is increasing exponentially with the number of nodes. [5]

Although [5] claims that ATEMU can be used with large scale network emulation. A limitation of ATMEU is it scalability, it scales poorly when used to emulate large scale networks.

2.2. Avrora

Avrora [3] is cycle-accurate instruction-level emulator for wireless sensor networks, which tries to overcome the scalability limitation of ATEMU without losing the cycle-level accuracy. Avrora is written in Java. The development of Avrora was stimulated by the observing that most energy-aware motes tend to sleep for long periods of time. This observation led to the idea of using event queues. Avrora also scales better with the number of processors, due to its multi-threading architecture. However, running every node in a separate thread raised synchronization problems. As it must be ensured that no node receiving a packet can proceed in a time before the time at which the packet was delivered. Avrora solves this problem by using synchronization intervals where each node is allowed to run to the end of the interval then it must wait for all other nodes to reach the end of interval too. Finally, Avrora lacks an accurate model for clock including drift and skew effects. It also lacks mobility emulation.

2.3. DiSenS

DiSenS[12] is a scalable network simulation framework for heterogeneous sensor networks. It uses an approach similar to that of Avrora where the hardware components of the wireless motes is cycle-accurately emulated and the radio model is simulated currently the "CC1000" and "CC2420 Zigbee" chips are supported. DiSenS currently supports the emulation of Mica2, MicaZ, iPAQ and Stargate devices. Unlike Avrora which uses discrete event queues and multi-threading to scale well on multi-processor systems.

DiSenS makes use of distributed clusters to overcome the scalability issue of cycle-accurate emulation while still maintains the fidelity of network evaluation.

DiSenS solves the synchronization problem by maintaining a local clock for every sensor node where, clocks are synchronized by using a clock update protocol where, every node broadcasts each clock time periodically. A node will only be allowed to receive a packet if and only if its clock is synchronized with the nodes transmitting the packet. As a distributed system, DiSenS partition the sensor nodes into groups and maintains a separate clock table for each group. Every group is emulated on a separate machine. DiSenS uses a graph partitioning package to minimize the communication links between neighbour nodes on different machines. As of the date of this publication, the DiSenS's website is no longer accessible as well as DiSenS itself is no longer publicly available.

3. The Proposed Emulator

Designing an emulator for wireless sensor networks dictates that some requirements should be satisfied as: high fidelity, scalability, flexibility and extendibility. In this paper we present the design and implementation of WiEmu, an open-source highly accurate, flexible and scalable emulator for wireless sensor networks. The architecture of an emulated sensor node is shown in Figure 1. Every node has its own attributes as: its ID, its geographic location, its state either dead or alive, energy consumed so far by the node, flash image file being executed, etc. At the core of WiEmu lies a cycle-accurate emulator. It accurately emulates the micro-controller at pin-level and its internal devices as: timers, RAM and flash. It also accurately emulates the external devices connected to the micro-controller as: LEDs, transceiver chips and batteries. Definitely cycle-level accuracy comes at the cost of performance. It was found by [3] that the sensor nodes tend to sleep most of the time. Thus rather than using busy-loop and polling the state of the node every clock cycle, WiEmu uses an event queue to boost the performance of the emulation process. Currently only Crossbow Mica2 motes are supported where WiEmu accurately emulates: Atmel ATmega128L micro-controller, including: RAM, timers, SPI and ADC, Chipcon CC1000 transceiver chip, and LEDs. WiEmu's object-oriented design allows the ease of extendibility as new architectures can be easily integrated by implementing an interface. As for the implementation C++ was chosen for its efficiency.

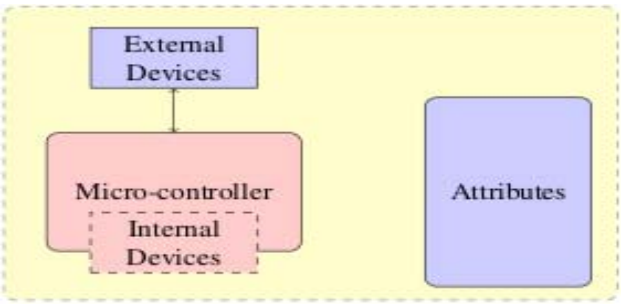

Fig. 1: WiEmu node architecture

WiEmu was designed with flexibility in mind. It provides a TCL interface for controlling the emulation parameters as well as the sensor network topology. TCL was chosen because of its popularity among network simulators as NS-2 and J-Sim [6, 7]. Regarding that WiEmu is written in C++ it is still bound to the TCL interpreter via SWIG [8, 9] an open-source tool that connects libraries written in compiled languages as C and C++ to scripting languages as Python or TCL. SWIG accepts an interface file as an input stating which classes, methods or variables from C/C++ libraries should be exposed to the scripting language interpreter.

Moreover, using an interpreted language gave WiEmu the ability to run as a mobile agent that can migrate from a machine to another in any heterogeneous network resuming execution on the new machine. Generally, mobile agents have several advantages as: fault tolerance and efficiency by reducing the network load as mobile agents tend to move the computation to data rather than versa. WiEmu was designed to be integrated with a mobile-agent system called D'Agents [10, 11] previously known as AgentTCL. D'Agents requires that the agent be written in TCL. Agents can migrate; they can also communicate together via

messages. Messages are used to transfer packets from one node to another, as well as for clock synchronization. The data flow diagram of WiEmu is shown in Figure 2.

In order to decrease the networking overhead DiSenS [12] uses a graph partitioning algorithm to group the neighbouring nodes on the same host using their known geographic location. While this approach enhances the performance of the network by decreasing the communication between hosts, it fails to take into consideration that the behaviour of sensor networks is not static. At a time two or more nodes may communicate extensively. However at other time they may not. On the other hand, WiEmu's agent-based design allows the extensively communicating nodes to migrate to the same host. They can also migrate once again to different hosts if their behaviour has changed.

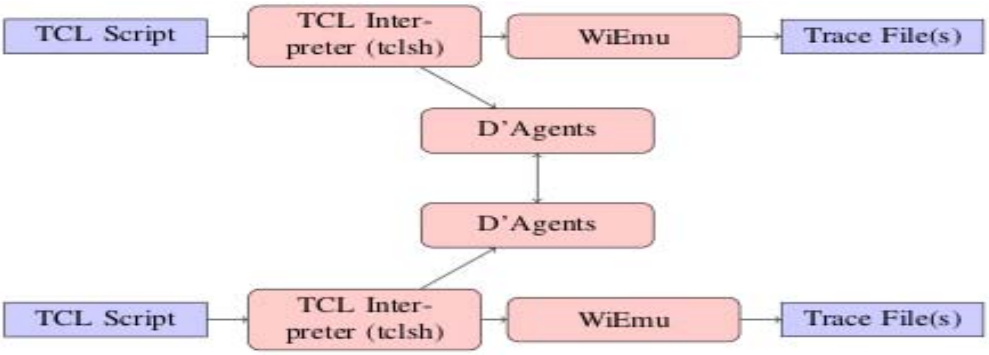

Fig. 2: WiEmu data flow diagram

4. Experimental Results

In this section, we show the result of the conducted experiments. WiEmu was able to run all standard TinyOS applications unmodified.

The cycle accuracy of WiEmu is validated versus that of Avrora-1.7 and Atemu-0.4, using Mica2 mote and as a reference AVR Studio 5 [15] was chosen. Four tests were conducted the first was an implementation of matrix multiplication algorithm, while the second was an implementation of binary search, the third was an implementation of the MD5 algorithm, and the fourth was a test for the accuracy of timers implementation. As shown in table 1 the results indicate that all of the emulators compared produced nearly the same results with close accuracy. However in the fourth test, WiEmu produced the exact result as that of AVR Studio 5 while Avrora and Atemu did not.

Finally, figure 3 shows the performance of WiEmu emulating a single node running TinyOS Blink application. Initially the time was linearly increasing as the number of cycles increased then it slowed down when the node tended to sleep most of the time.

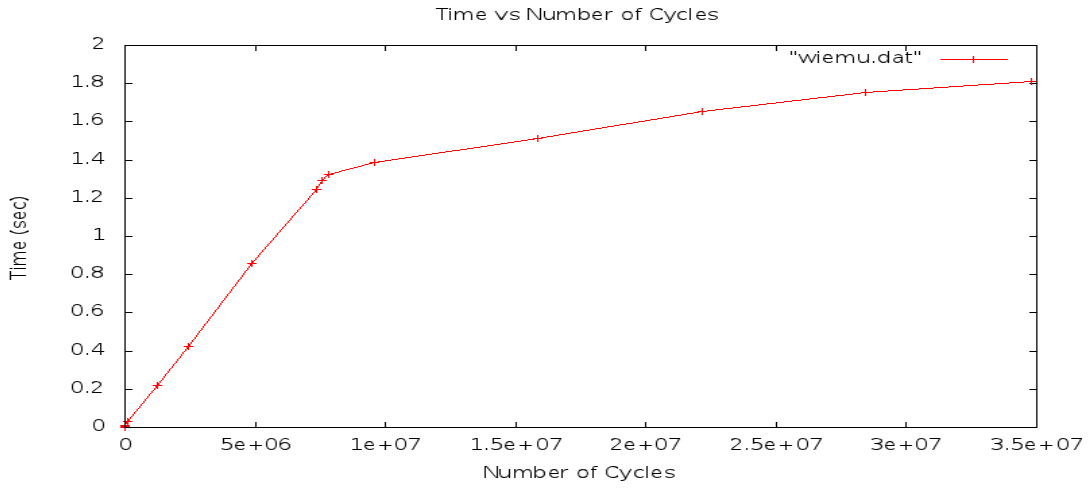

Fig. 3: Performance evaluation of WiEmu for a single node

Table. 1: Accuracy for the micro-controller implementation

Test	Description	MCU Clock Cycles			
		WiEmu	Avrora	Atemu	AVR Studio
1	Matrix Multiplication	4357	4357	4356	4357
2	Binary Search	864	864	863	864
3	MD5	37686	37686	37685	37686
4	Timer	1361	1379	1354	1361

5. Conclusion and Future Work

In wireless sensor network new development, modifications and implementations need to be evaluated before being actually deployed. It's not feasible to test every modification on real test-beds. Simulation can be highly scalable, but it lacks the high accuracy. On the other hand, Emulation is highly accurate, but it has limited scalability. In this paper we presented the design and implementation of a cycle-accurate flexible and scalable software emulator for wireless sensor networks that can be easily extended. For future work, new mote architectures can be added to support heterogeneous emulation, an accurate battery model and an accurate air model should be implemented. More work can be done on the support of mobility emulation.

6. Acknowledgements

The authors would like to thank Ramy Eltarras an assistant professor at the department of Computer Science, College of Computing and Information Technology, Arab Academy for Science and Technology for his guidance and support throughout the work.

7. References

[1] Furqan Haq and Thomas Kunz. *Simulation vs. Emulation Evaluating Mobile Ad Hoc Network Routing Protocols*, In Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN 2005), London, UK, May 2005.

[2] "WiEmu". Internet: http://wiemu.berlios.de, [Dec. 20, 2011]

[3] Ben Titzer, Daniel Lee, and Jens Palsberg. *Avrora: Scalable Sensor Network Simulation with Precise Timing*, In Proceedings of the Fourth International Conference on Information Processing in Sensor Networks, 2005. (IPSN'05)

[4] "NS-2". Internet: http://www.isi.edu/nsnam/ns, [Oct. 24, 2010]

[5] Jonathan Polley, Dionysys Blazakis, Jonathan McGee, Dan Rusk, John S. Baras and Manish Karir. *ATEMU: A Fine-grained Sensor Network Simulator*, In Proceedings of The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. (SECON'04)

[6] "J-Sim". Internet: http://sites.google.com/site/jsimofficial, [Oct. 24, 2010]

[7] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang, *J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks*, IEEE Wireless Communications magazine 2005.

[8] David M. Beazley. *SWIG: An Easy to Use Tool for Integrating Scripting Languages with C and C++*, Proceedings of the 4th conference on USENIX Tcl/Tk Workshop, 1996 - Volume 4 (TCLTK'96), Vol. 4. USENIX Association, Berkeley, CA, USA, 15-15., 1996.

[9] "SWIG". Internet: http://www.swig.org, [Dec. 21, 2011]

[10] "D'Agents". Internet: http://agent.cs.dartmouth.edu, [Dec. 21, 2011]

[11] Robert S. Gray. Agent *Tcl: A transportable agent system*, Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95), Baltimore, Maryland, December 1995.

[12] Ye Wen, Rich Wolski and Gregory Moore, DiSenS*: Scalable Distributed Sensor Network Simulation*, In Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'07). 2005.

[13] Sung Park, Andreas Savvides and Mani B. Srivastava, *SensorSim: a simulation framework for sensor networks*, In the proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, Boston, Massachusetts, United States. 2000.

[14] Ana Belenm, Garcia Hernando and et al, *Problem Solving for Wireless Sensor Networks*, London: Springer, 2008.

[15] "AVR Studio". Internet: www.atmel.com/microsite/avr_studio_5, [Jan. 14, 2012]