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

Eng. Mohamed A. Aslan

Under supervision of:
Prof. Dr. Mohamad S. Abou El-Nasr
Assoc. Prof. Dr. Sherin M. Youssef

April 12, 2012
Introduction

Literature Survey

The Proposed Emulator

Validation

Conclusion & Future Work
Networks of resource-constrained wireless interconnected devices that cooperatively monitor a physical or environmental phenomena.
Applications of WSN

WSNs have lots of applications varying from civilian to military. Example:

- Environmental Monitoring
- Surveillance
- Agricultural Monitoring
- Industrial Monitoring
- Traffic Monitoring
- Monitoring of elderly people
WSNs faces a lot of challenges that are under active research.
Constraints & Challenges of WSN

WSNs faces a lot of challenges that are under active research.

- **Constraints**
  - Limited resources
    - Battery lifetime
    - Processing power
    - Memory capacity
    - Storage capacity
  - Challenges
    - Energy harvesting
    - Communication
    - Sensing
    - Mobility
    - Clustering
Constraints & Challenges of WSN

WSNs faces a lot of challenges that are under active research.

- **Constraints**
  - Limited resources
    - Battery lifetime
    - Processing power
    - Memory capacity
    - Storage capacity

- **Challenges**
  - Energy harvesting
  - Communication
  - Sensing
  - Mobility
  - Clustering
Types of Sensor Networks

WSNs can be classified according to their deployment.

- Terrestrial Wireless Sensor Networks
- Underground Wireless Sensor Networks
- Underwater Wireless Sensor Networks
- Wireless Multi-media Sensor Networks
- Mobile Wireless Sensor Networks
Motivation

In WSNs software, new development, modifications and implementations need to be evaluated before the actual deployment.
In WSNs software, new development, modifications and implementations need to be evaluated before the actual deployment.

The actual deployment process:
Motivation

- In WSNs software, new development, modifications and implementations need to be evaluated before the actual deployment.

- The actual deployment process:
  - Time consuming
  - High in cost
  - Requires maintenance
Motivation

- In WSNs software, new development, modifications and implementations need to be evaluated before the actual deployment.

- The actual deployment process:
  - Time consuming
  - High in cost
  - Requires maintenance

- It’s not feasible to test every modification on real test-beds.
In WSNs software, new development, modifications and implementations need to be evaluated before the actual deployment.

The actual deployment process:
- Time consuming
- High in cost
- Requires maintenance

It’s not feasible to test every modification on real test-beds.

Thus a solution is required:
In WSNs software, new development, modifications and implementations need to be evaluated before the actual deployment.

The actual deployment process:
- Time consuming
- High in cost
- Requires maintenance

It’s not feasible to test every modification on real test-beds.

Thus a solution is required:
- Simulation
- Emulation
## Simulation vs Emulation

<table>
<thead>
<tr>
<th></th>
<th>Simulation</th>
<th>Emulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Def.</strong></td>
<td>▶ Imitation or modelling of real system</td>
<td>▶ Reproducing an accurate behaviour of a real system</td>
</tr>
<tr>
<td></td>
<td>▶ Representing certain key characteristics or behaviours of a real system</td>
<td>▶ Considering more low-level details</td>
</tr>
<tr>
<td></td>
<td>▶ Ignoring low-level details</td>
<td></td>
</tr>
<tr>
<td><strong>Pros</strong></td>
<td>High Scalability</td>
<td>High Accuracy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ Cycle-accurate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▶ OS-accurate</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>Limited Accuracy</td>
<td>Limited Scalability</td>
</tr>
</tbody>
</table>

Eng. Mohamed A. Aslan

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

- Our main objective is to develop a precise cycle-accurate, flexible and scalable software emulator for WSNs.
Objective

- Our main objective is to develop a precise cycle-accurate, flexible and scalable software emulator for WSNs.
- A software emulator that is able to run the motes’ software transparently unmodified as if it runs on real mote’s hardware to facilitate the evaluation process of the WSNs prior the actual deployment.
Literature Survey

- Discrete Event Simulators
  - J-SIM
  - NS-2
  - OMNET++
  - OPNET
Literature Survey

- Discrete Event Simulators
  - J-SIM
  - NS-2
  - OMNET++
  - OPNET
- Software Emulators
  - sQualNet
  - TOSSIM
  - Atemu
  - Avrora
  - DiSenS
Comparison between various network simulators

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Language</th>
<th>License</th>
<th>Mobility</th>
<th>Power model</th>
</tr>
</thead>
<tbody>
<tr>
<td>J-SIM</td>
<td>D.E.S</td>
<td>Java</td>
<td>OSS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NS-2</td>
<td>D.E.S</td>
<td>C++</td>
<td>OSS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OMNeT++</td>
<td>D.E.S</td>
<td>C++</td>
<td>OSS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>OPNet</td>
<td>D.E.S</td>
<td>N/A</td>
<td>Commercial</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Comparison between various network emulators

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Language</th>
<th>License</th>
<th>Accuracy</th>
<th>Mobility</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>sQualNet</td>
<td>DES + Emul.</td>
<td>N/A</td>
<td>OSS</td>
<td>OS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TOSSIM</td>
<td>Emulator</td>
<td>C</td>
<td>OSS</td>
<td>OS</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Atemu</td>
<td>Emulator</td>
<td>C</td>
<td>OSS</td>
<td>Cycle</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Avrora</td>
<td>Emulator</td>
<td>Java</td>
<td>OSS</td>
<td>Cycle</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DiSenS²</td>
<td>Emulator</td>
<td>N/A</td>
<td>N/A</td>
<td>Cycle</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

¹ via PowerTOSSIM
² The project/website is no longer available
J-SIM

- Open-source discrete event simulator based on a component-based architecture
J-SIM

- Open-source discrete event simulator based on a component-based architecture
- Includes GUI called GEditor
J-SIM

- Open-source discrete event simulator based on a component-based architecture
- Includes GUI called GEditor
- Composed of:
  - Autonomous Component Architecture
  - INET is a packet-switching inter-networking framework
  - Jacl a TCL scripting framework
NS-2

- Open-source discrete event simulator
NS-2

- Open-source discrete event simulator
- Ships with NAM a network animator
NS-2

- Open-source discrete event simulator
- Ships with NAM a network animator
- Supports TCL as the scripting language
Open-source discrete event simulator
Ships with NAM a network animator
Supports TCL as the scripting language
Does not include a complete support for wireless sensor networks
NS-2

- Open-source discrete event simulator
- Ships with NAM a network animator
- Supports TCL as the scripting language
- Does not include a complete support for wireless sensor networks
- Extensions as SensorSim were developed to provide NS-2 with support for WSNs
OMNET++

- Open-source component-based discrete-event simulator
OMNET++

- Open-source component-based discrete-event simulator
- Components are written in C++ and are bound together via the NED (NEtwork Description) programming language
OMNET++

- Open-source component-based discrete-event simulator
- Components are written in C++ and are bound together via the NED (NEtwork Description) programming language
- NED used to define regular topologies such as chain, ring, mesh, structures, ... etc
OMNET++

- Open-source component-based discrete-event simulator
- Components are written in C++ and are bound together via the NED (NEtwork Description) programming language
- NED used to define regular topologies such as chain, ring, mesh, structures, ... etc
- Also comes with an Integrated Development Environment (IDE) that facilitates the development and analysis of results
OPNET

- Commercial discrete-event simulator
OPNET

- Commercial discrete-event simulator
- Initially developed at the MIT before commercialization.
OPNET

- Commercial discrete-event simulator
- Initially developed at the MIT before commercialization.
- Offers a powerful GUI used tools for analysis, visualization and debugging
OPNET

- Commercial discrete-event simulator
- Initially developed at the MIT before commercialization.
- Offers a powerful GUI used tools for analysis, visualization and debugging
- Lacks detailed models for wireless sensor networks
sQualNet

- Scalable emulation framework for sensor networks
sQualNet

- Scalable emulation framework for sensor networks
- Extension to the commercial QualNet network simulator
sQualNet

- Scalable emulation framework for sensor networks
- Extension to the commercial QualNet network simulator
- Integrates the high fidelity of emulation and scalability of simulation
  - Discrete event network simulator, +
  - SOS operating system level emulator
sQualNet

- Scalable emulation framework for sensor networks
- Extension to the commercial QualNet network simulator
- Integrates the high fidelity of emulation and scalability of simulation
  - Discrete event network simulator, +
  - SOS operating system level emulator
- Deals with the emulation as an event, that has a separate event handler
TOSSIM

- Discrete event and operating system level emulator for TinyOS
TOSSIM

- Discrete event and operating system level emulator for TinyOS
- Ships with TinyViz a visualization tool written in Java
TOSSIM

- Discrete event and operating system level emulator for TinyOS
- Ships with TinyViz a visualization tool written in Java
- Modified the TinyOS by replacing the low-level hardware interrupts with discrete event simulation events
  - i.e. Allows TinyOS to be compiled into native machine code
TOSSIM

- Discrete event and operating system level emulator for TinyOS
- Ships with TinyViz a visualization tool written in Java
- Modified the TinyOS by replacing the low-level hardware interrupts with discrete event simulation events
  - i.e. Allows TinyOS to be compiled into native machine code
- Models a WSN as a directed graph
  - Every vertex represents a sensor node, and
  - Every edge represents the communication channel between two nodes
  - Each edge has a bit error probability
Atenu

- Atmel Emulator

Agenda
Introduction
Literature Survey
WiEmu
Validation
Conclusion & Future Work

J-SIM
NS-2
OMNET++
OPNET
sQualNet
TOSSIM
Atemu
Avrora
DiSenS

Eng. Mohamed A. Aslan
WiEmu: The Design and Implementation of a Flexible Agent-based Scalable Network Emulator for Heterogeneous Wireless Sensor Networks
Atemu

- **Atmel Emulator**
- Cycle-accurate instruction-level emulator for WSN
Atmel Emulator

- Cycle-accurate instruction-level emulator for WSN
- Simulates the wireless communication between the nodes
Atemu

- **Atmel Emulator**
- Cycle-accurate instruction-level emulator for WSN
- Simulates the wireless communication between the nodes
- Supposed to emulate heterogeneous nodes
  - Currently only support MICA2 motes
  - Runs TinyOS built for MICA2 motes unmodified
Atemu

- Atmel Emulator
- Cycle-accurate instruction-level emulator for WSN
- Simulates the wireless communication between the nodes
- Supposed to emulate heterogeneous nodes
  - Currently only support MICA2 motes
  - Runs TinyOS built for MICA2 motes unmodified
- XML-based files for the configuration of the network specifications
Avrora

- Cycle-accurate instruction-level multi-threaded emulator for WSN
Avrora

- Cycle-accurate instruction-level multi-threaded emulator for WSN
- Tries to overcome the scalability issue of ATEMU without losing the cycle accuracy
Avrora

- Cycle-accurate instruction-level multi-threaded emulator for WSN
- Tries to overcome the scalability issue of ATEMU without losing the cycle accuracy
- Realized that most energy-aware motes tend to sleep for long periods
  - i.e. Introduced event queueing
  - Synchronization problems appeared (solved via sync intervals)
    - Receive Signal Strength Indication $\text{RSSI}$ problem
    - Send-Receive problem
Avrora

- Cycle-accurate instruction-level multi-threaded emulator for WSN
- Tries to overcome the scalability issue of ATEMU without losing the cycle accuracy
- Realized that most energy-aware motes tend to sleep for long periods
  - i.e. Introduced event queueing
  - Synchronization problems appeared (solved via sync intervals)
    - Receive Signal Strength Indication RSSI problem
    - Send-Receive problem
- Lacks accurate clock model and mobility emulation
DiSenS

- Distributed Sensor network Simulator
DiSenS

- Distributed Sensor network Simulator
- Scalable network emulation cycle-accurate framework for WSNs
DiSenS

- **Distributed Sensor network Simulator**
- Scalable network emulation cycle-accurate framework for WSNs
- Makes use of distributed clusters to overcome the scalability issues of cycle-accurate emulation
DiSenS

- Distributed Sensor network Simulator
- Scalable network emulation cycle-accurate framework for WSNs
- Makes use of distributed clusters to overcome the scalability issues of cycle-accurate emulation
- Clocks are synchronized by using a clock update protocol
  - Every node broadcasts each clock time periodically
DiSenS

- **Distributed Sensor** network **Simulator**
- Scalable network emulation cycle-accurate framework for WSNs
- Makes use of distributed clusters to overcome the scalability issues of cycle-accurate emulation
- Clocks are synchronized by using a clock update protocol
  - Every node broadcasts each clock time periodically
- Uses a graph partitioning package
  - to minimize the communication links between neighbour nodes on different machines
DiSenS

- Distributed Sensor network Simulator
- Scalable network emulation cycle-accurate framework for WSNs
- Makes use of distributed clusters to overcome the scalability issues of cycle-accurate emulation
- Clocks are synchronized by using a clock update protocol
  - Every node broadcasts each clock time periodically
- Uses a graph partitioning package
  - to minimize the communication links between neighbour nodes on different machines
- The project & website are no longer accessible
The Proposed Emulator (WiEmu)

- A scalable cycle-accurate software emulator for WSNs.
The Proposed Emulator (WiEmu)

- A scalable cycle-accurate software emulator for WSNs.
- Uses a hybrid approach
  - Emulates mote architecture
  - Simulates air model
As for non-functional requirements, the proposed emulator is designed with:

- High fidelity (Accuracy)
- Flexibility
- Extendibility
- Scalability
System Design

- As for non-functional requirements, the proposed emulator is designed with:
  - High fidelity (Accuracy)
As for non-functional requirements, the proposed emulator is designed with:

- High fidelity (Accuracy)
- Flexibility
As for non-functional requirements, the proposed emulator is designed with:

- High fidelity (Accuracy)
- Flexibility
- Extendibility
As for non-functional requirements, the proposed emulator is designed with:

- High fidelity (Accuracy)
- Flexibility
- Extendibility
- Scalability
Currently supported mote architecture:

- Mica2 platform:
  - Atmega128 micro-controller
    - Timers
    - SPI
    - ADC
  - LEDs
  - CC1000 radio chip
Flexible TCL Interface Example

```
# Node: 0, Runs: TinyOS Blink Application
set n0 [Node]
$n0 setFlash "Blink.bin"
$n0 setID 1
$n0 setLocation 10 10 0
set d0 [Debugger]
$d0 setFile "debug0.dat"
$d0 enableDisassembly
$n0 setDebugger $d0
set t0 [Thread]
$t0 setNode $n0

# Node: 1, Runs: TinyOS CntToRfm Application
set n1 [Node]
$n1 setFlash "CntToRfm.bin"
$n1 setID 2
$n1 setLocation 50 50 0
set t1 [Thread]
$t1 setNode $n1

# Start all threads
Thread_start
```
WiEmu: The Design and Implementation of a Flexible Agent-based Scalable Network Emulator for Heterogeneous Wireless Sensor Networks

**WiEmu’s Data Flow Diagram**

1. **TCL Script** → **TCL Interpreter (tclsh)** → **WiEmu** → **Trace File(s)**
2. **Machine A**
   - Agent
   - **TCL Script** → **TCL Interpreter (tclsh)** → **WiEmu** → **Trace File(s)**
3. **Machine B**
   - Agent
   - **TCL Script** → **TCL Interpreter (tclsh)** → **WiEmu** → **Trace File(s)**
WiEmu is written in C++

WiEmu is compiled as a shared library (module) that is loaded by the TCL interpreter.

WiEmu relies on the following components:

- SWIG: links WiEmu to the TCL interpreter
- D'Agents: mobile-agents framework

Currently only Mica2 is supported.
System Implementation

- WiEmu is written in C++
- WiEmu is compiled as a shared library (module) that is loaded by the TCL interpreter
WiEmu is written in C++. WiEmu is compiled as a shared library (module) that is loaded by the TCL interpreter. WiEmu relies on the following components:
System Implementation

- WiEmu is written in C++
- WiEmu is compiled as a shared library (module) that is loaded by the TCL interpreter
- WiEmu relies on the following components:
  - SWIG: links WiEmu to the TCL interpreter
WiEmu is written in C++

WiEmu is compiled as a shared library (module) that is loaded by the TCL interpreter

WiEmu relies on the following components:
  - SWIG: links WiEmu to the TCL interpreter
  - D’Agents: mobile-agents framework
WiEmu is written in C++

WiEmu is compiled as a shared library (module) that is loaded by the TCL interpreter

WiEmu relies on the following components:
  ▶ SWIG: links WiEmu to the TCL interpreter
  ▶ D’Agents: mobile-agents framework

Currently only Mica2 is supported
SWIG

- **Simplified Wrapper and Interface Generator**
SWIG

- **Simplified Wrapper and Interface Generator**
- Open source software tool used to connect libraries written in C or C++ with scripting languages such as Perl, PHP, Python, Ruby, Tcl.
SWIG

- Simplified **W**rapper and **I**nterface **G**enerator

- Open source software tool used to connect libraries written in C or C++ with scripting languages such as Perl, PHP, Python, Ruby, Tcl.

- The aim is to allow calling native functions (C or C++) by the interpreted code
The C/C++ code:

```cpp
#include <time.h>

double My_variable = 3.0;

int fact(int n) {
    if (n <= 1) return 1;
    else return n * fact(n - 1);
}

int my_mod(int x, int y) {
    return (x % y);
}

char *get_time()
{
    time_t ltime;
    time(&ltime);
    return ctime(&ltime);
}
```
The interface file:

```c
module example {

    /* Put header files here or function declarations like below */
    extern double My_variable;
    extern int fact(int n);
    extern int my_mod(int x, int y);
    extern char *get_time();
}

extern double My_variable;
extern int fact(int n);
extern int my_mod(int x, int y);
extern char *get_time();
```
SWIG Example

Compilation and Linking:

$ swig -tcl example.i
$ gcc -fpic -c example.c example_wrap.c -l/usr/local/include
$ gcc -shared example.o example_wrap.o -o example.so
$ tclsh
% load ./example.so example
% puts $My_variable
3.0
% fact 5
120
% my_mod 7 3
1
% get_time
Sun Feb 11 23:01:07 1996
D’Agents

- D’Agents is a transportable agent system
D’Agents

- D’Agents is a transportable agent system
- Previously known as “Agent Tcl”
D’Agents

- D’Agents is a transportable agent system
- Previously known as “Agent Tcl”
- The transportable agents are written in the Tool Command Language (Tcl)
D’Agents

- D’Agents is a transportable agent system
- Previously known as “Agent Tcl”
- The transportable agents are written in the Tool Command Language (Tcl)
- The agents migrate from machine to machine using the jump command Execution resumes on the destination machine at the statement immediately after the jump
Atmega128L Micro-controller
CC1000 Transceiver
CC1000 Micro-controller Interfacing
## Configuration Interface

<table>
<thead>
<tr>
<th>Chip Line</th>
<th>Micro-controller Pin</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK</td>
<td>PD6</td>
<td>31</td>
</tr>
<tr>
<td>PDATA</td>
<td>PD7</td>
<td>32</td>
</tr>
<tr>
<td>PALE</td>
<td>PD4</td>
<td>29</td>
</tr>
</tbody>
</table>
Data Interface

<table>
<thead>
<tr>
<th>Chip Line</th>
<th>Micro-controller Pin</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCLK</td>
<td>SPI_SCK (PB1/SCK)</td>
<td>11</td>
</tr>
<tr>
<td>DIO</td>
<td>SPI_MISO (PB2/MOSI), SPI_MISO (PB3/MISO)</td>
<td>12, 13</td>
</tr>
<tr>
<td>RSSI</td>
<td>PF0/ADC0</td>
<td>61</td>
</tr>
</tbody>
</table>
WiEmu is able to run TinyOS applications unmodified
Validation

- WiEmu is able to run TinyOS applications unmodified
- The following TinyOS-1.x applications were able to run successfully on WiEmu:
  - Blink
  - CntToRfm
  - RfmToCnt
The following experiments were conducted on a computer with the following specifications:

- Intel Core i5 64-bit CPU operating at 2.67 GHz
- 4 GB RAM
- Ubuntu 11.10 amd64 with Linux 3.0.0-12 kernel
## Cycle Accuracy

<table>
<thead>
<tr>
<th>Test</th>
<th>Description</th>
<th>WiEmu</th>
<th>Avrora</th>
<th>Atemu</th>
<th>AVR Studio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Matrix Multiplication</td>
<td>4357</td>
<td>4357</td>
<td>4356</td>
<td>4357</td>
</tr>
<tr>
<td>2</td>
<td>Binary Search</td>
<td>864</td>
<td>864</td>
<td>863</td>
<td>864</td>
</tr>
<tr>
<td>3</td>
<td>MD5</td>
<td>37686</td>
<td>37686</td>
<td>37685</td>
<td>37686</td>
</tr>
<tr>
<td>4</td>
<td>Timer</td>
<td>1361</td>
<td>1379</td>
<td>1354</td>
<td>1361</td>
</tr>
</tbody>
</table>
CPU-bound application (Prime Numbers Counter)
TinyOS-1.x Blink (Single Node)

Time vs Number of Cycles

- "WiEmu.dat"
TinyOS-1.x Blink (Multiple Nodes)

Time vs Number of Nodes

- Time (sec)
- Number of Nodes

Graph showing the relationship between time and number of nodes.
The implementation of a cycle-accurate and scalable software emulator for WSNs with easily extendible architecture
Conclusion

- The implementation of a cycle-accurate and scalable software emulator for WSNs with easily extendible architecture
- Flexibility, a simple TCL interface is developed to support the configuration of the WSNs being emulated
Conclusion

- The implementation of a cycle-accurate and scalable software emulator for WSNs with easily extendible architecture
- Flexibility, a simple TCL interface is developed to support the configuration of the WSNs being emulated
- The support of a scalable emulation using a multi-threaded agent-based distributed architecture
Conclusion

- The implementation of a cycle-accurate and scalable software emulator for WSNs with easily extendible architecture
- Flexibility, a simple TCL interface is developed to support the configuration of the WSNs being emulated
- The support of a scalable emulation using a multi-threaded agent-based distributed architecture
- The proposed emulator was able to run Mica2 applications unmodified
Future Work

- New mote architectures
Future Work

- New mote architectures
- Accurate Battery Model
Future Work

- New mote architectures
- Accurate Battery Model
- Accurate Air Model
Future Work

- New mote architectures
- Accurate Battery Model
- Accurate Air Model
- Mobility Emulation
Future Work

- New mote architectures
- Accurate Battery Model
- Accurate Air Model
- Mobility Emulation
- Dynamic Re-compilation
Thanks for listening ☺