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

Eng. Mohamed A. Aslan

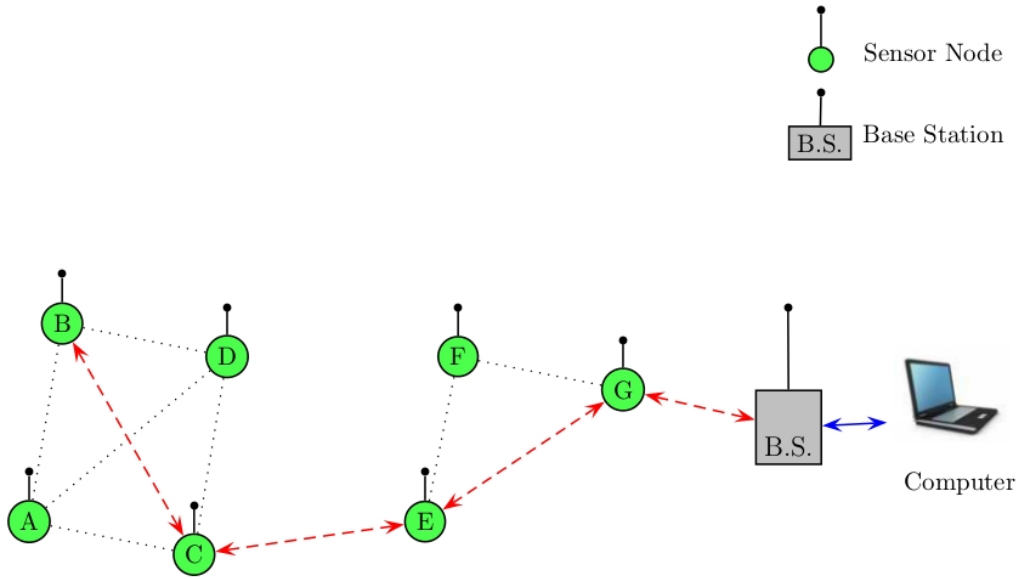
Under supervision of:
Prof. Dr. Mohamad S. Abou El-Nasr
Assoc. Prof. Dr. Sherin M. Youssef

April 12, 2012

Agenda

Wireless Sensor Networks

- 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

Constraints & Challenges of WSN

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

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.

Motivation

- 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.

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.
- Thus a solution is required:

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.
- Thus a solution is required:
  - Simulation
  - Emulation

Simulation vs Emulation

	Simulation	Emulation
Def.	▶ Imitation or modelling of real system ▶ Representing certain key characteristics or behaviours of a real system ▶ Ignoring low-level details	▶ Reproducing an accurate behaviour of a real system ▶ Considering more low-level details
Pros	High Scalability	High Accuracy ▶ Cycle-accurate ▶ OS-accurate
Cons	Limited Accuracy	Limited Scalability

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 [1]

Name	Type	Language	License	Mobility	Power model
J-SIM	D.E.S	Java	OSS	Yes	Yes
NS-2	D.E.S	C++	OSS	Yes	Yes
OMNeT++	D.E.S	C++	OSS	Yes	Yes
OPNet	D.E.S	N/A	Commercial	N/A	N/A

[1] Ana-Beln Garca-Hernando, et al. Problem Solving for Wireless Sensor Networks. Springer Publishing Company, 2008.

Comparison between various network emulators

Name	Type	Language	License	Accuracy	Mobility	Power
sQualNet	DES + Emul.	N/A	OSS	OS	Yes	Yes
TOSSIM	Emulator	C	OSS	OS	No	[1]
Atemu	Emulator	C	OSS	Cycle	No	Yes
Avrora	Emulator	Java	OSS	Cycle	No	Yes
DiSenS[2]	Emulator	N/A	N/A	Cycle	N/A	N/A

[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

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

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 is Java

TOSSIM

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

Atemu

- Atmel Emulator

Atemu

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

Atemu

- 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 *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

System Design

- As for non-functional requirements, the proposed emulator is designed with:

System Design

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

System Design

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

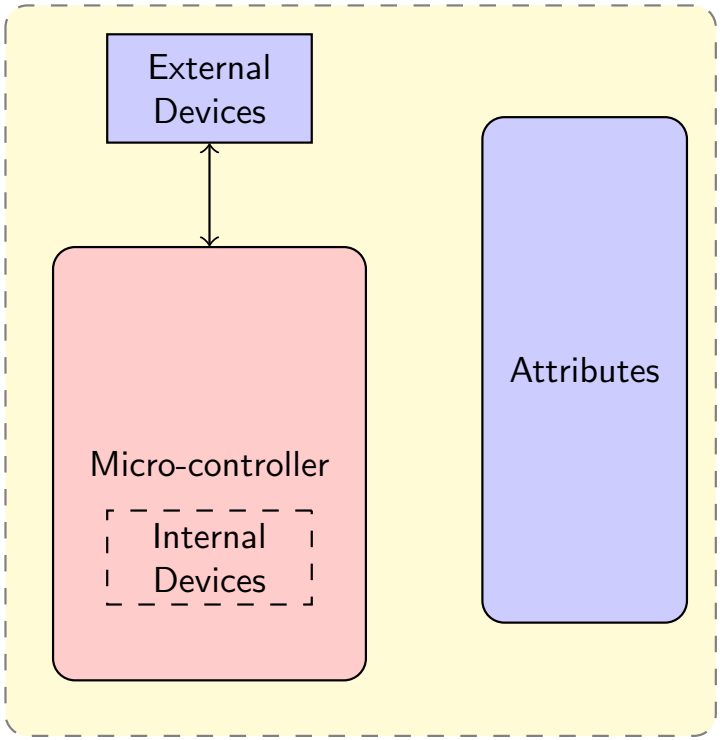
System Design

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

System Design

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

Generic Node Architecture



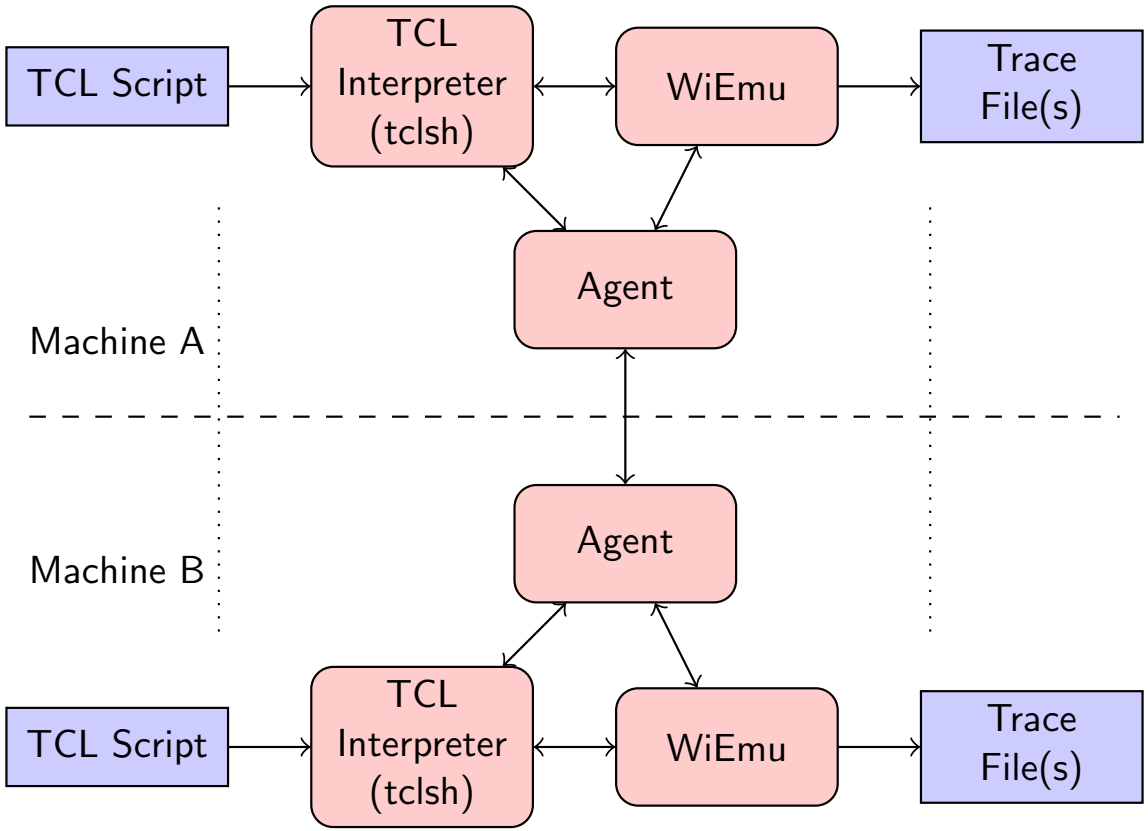
Currently supported mote architecture:

- Mica2 platform:
  - Atmega128 micro-controller
    - Timers
    - SPI
    - ADC
  - LEDs
  - CC1000 radio chip

Flexible TCL Interface Example

```
1  # Node: 0, Runs: TinyOS Blink Application
2  set n0 [Node]
3  $n0 setFlash "Blink.bin"
4  $n0 setID 1
5  $n0 setLocation 10 10 0
6  set d0 [Debugger]
7  $d0 setFile "debug0.dat"
8  $d0 enableDisassembly
9  $n0 setDebugger $d0
10 set t0 [Thread]
11 $t0 setNode $n0
12 # Node: 1, Runs: TinyOS CntToRfm Application
13 set n1 [Node]
14 $n1 setFlash "CntToRfm.bin"
15 $n1 setID 2
16 $n1 setLocation 50 50 0
17 set t1 [Thread]
18 $t1 setNode $n1
19 # Start all threads
20 Thread_start
```

WiEmu's Data Flow Diagram



System Implementation

- WiEmu is written in C++

System Implementation

- WiEmu is written in C++
- WiEmu is compiled as a shared library (module) that is loaded by the TCL interpreter

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:

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

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
  - D'Agents: mobile-agents framework

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
  - D'Agents: mobile-agents framework
- Currently only Mica2 is supported

SWIG

- **S**implified **W**rapper and **I**nterface **G**enerator

SWIG

- **S**implified **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.

SWIG

- **S**implified **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

SWIG Example

The C/C++ code:

```
1  #include <time.h>
2   double My_variable = 3.0;
3
4   int fact(int n) {
5       if (n <= 1) return 1;
6       else return n*fact(n-1);
7   }
8
9   int my_mod(int x, int y) {
10      return (x%y);
11  }
12
13  char *get_time()
14  {
15      time_t ltime;
16      time(&ltime);
17      return ctime(&ltime);
18  }
```

SWIG Example

The interface file:

```
1  %module example
2   %{
3   /* Put header files here or function declarations like below */
4   extern double My_variable;
5   extern int fact(int n);
6   extern int my_mod(int x, int y);
7   extern char *get_time();
8   %}
9
10  extern double My_variable;
11  extern int fact(int n);
12  extern int my_mod(int x, int y);
13  extern char *get_time();
```

SWIG Example

Compilation and Linking:

```
$ swig -tcl example.i
$ gcc -fpic -c example.c example_wrap.c -I/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"

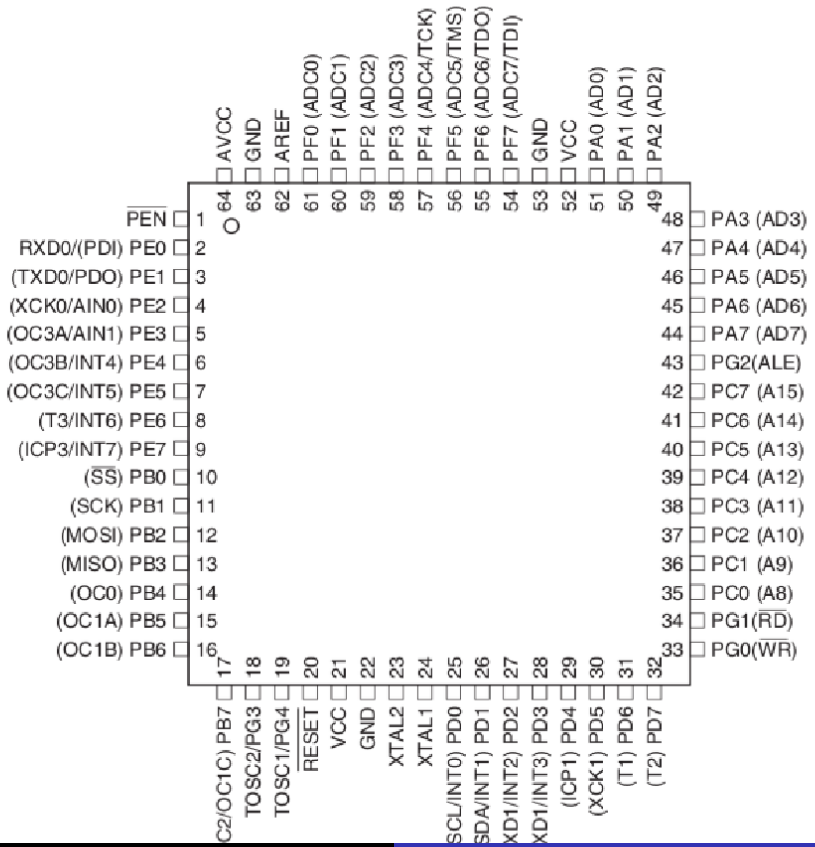
D'Agents

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

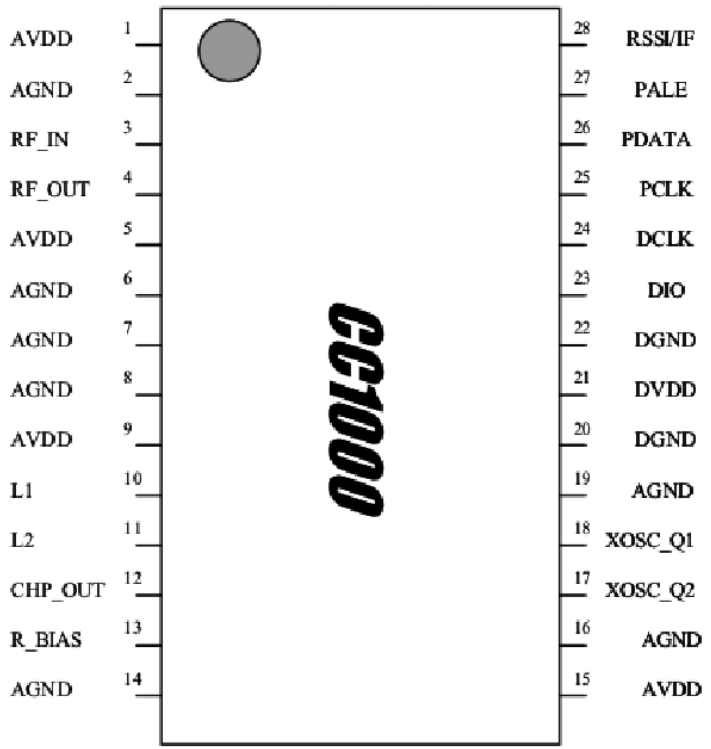
D'Agents

- 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

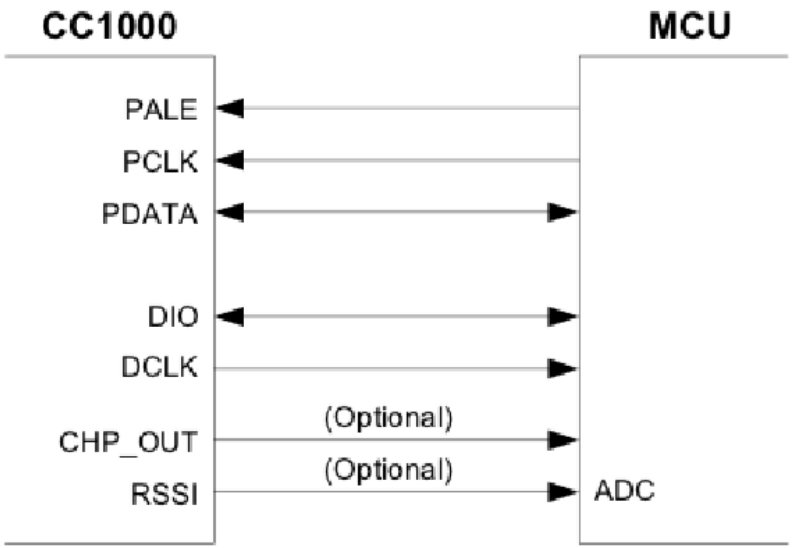
Atmega128L Micro-controller



CC1000 Transceiver



CC1000 Micro-controller Interfacing



Configuration Interface

Chip Line	Micro-controller Pin	
	Name	Number
PCLK	PD6	31
PDATA	PD7	32
PALE	PD4	29

Data Interface

Chip Line	Micro-controller Pin	
	Name	Number
DCLK	SPI_SCK (PB1/SCK)	11
DIO	SPI_MISO (PB2/MOSI), SPI_MISO (PB3/MISO)	12, 13
RSSI	PF0/ADC0	61

Validation

- WiEmu is able to run TinyOS applications unmodified

Validation

- WiEmu is able to run TinyOS applications unmodified
- The following TinyOS-1.x applications where able to run successfully on WiEmu:
  - Blink
  - CntToRfm
  - RfmToCnt

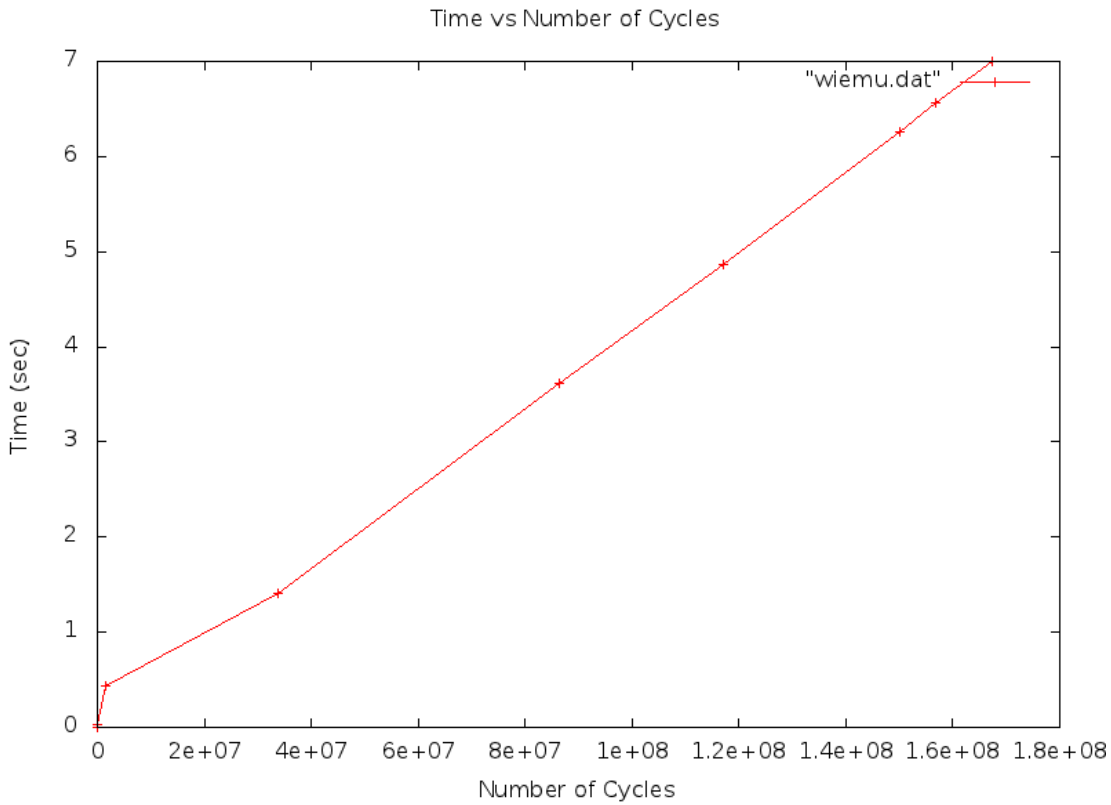
Experimental Results

- 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

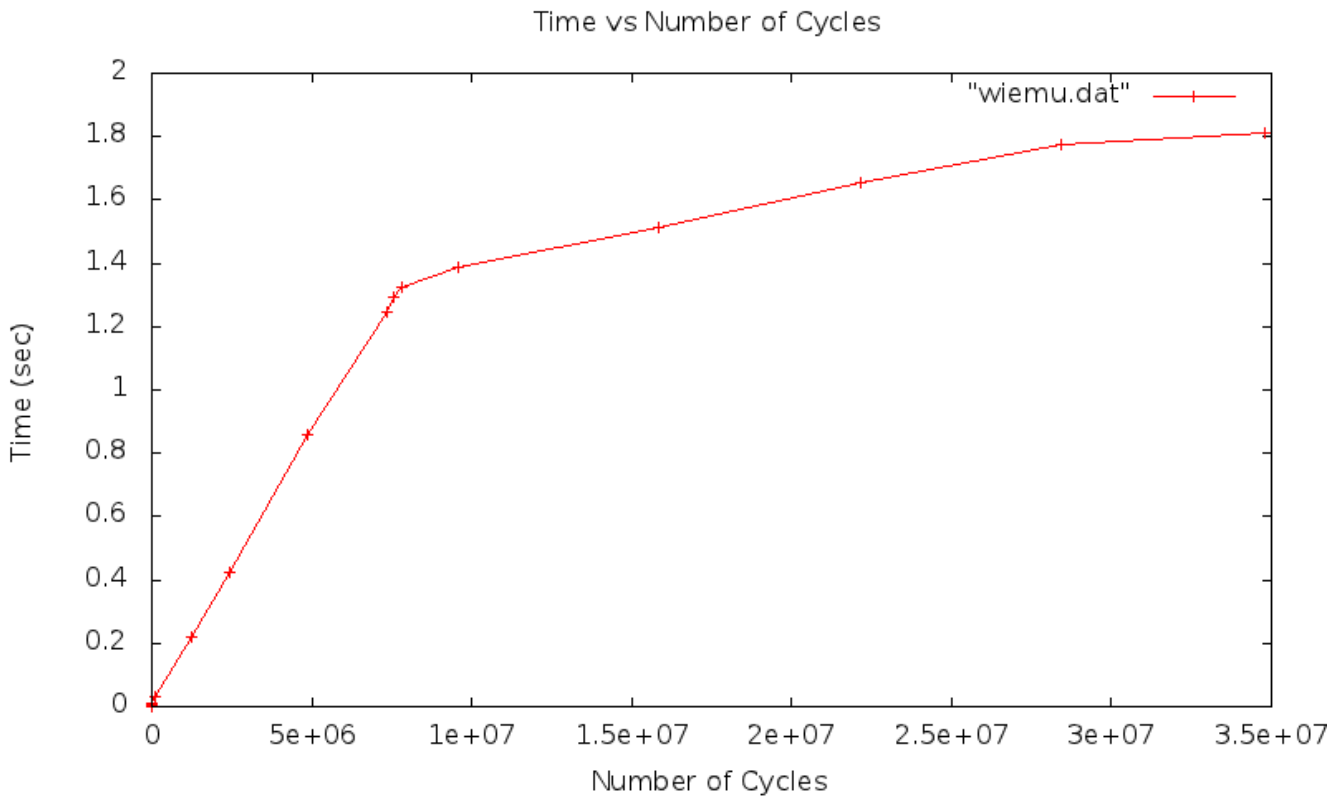
Cycle Accuracy

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

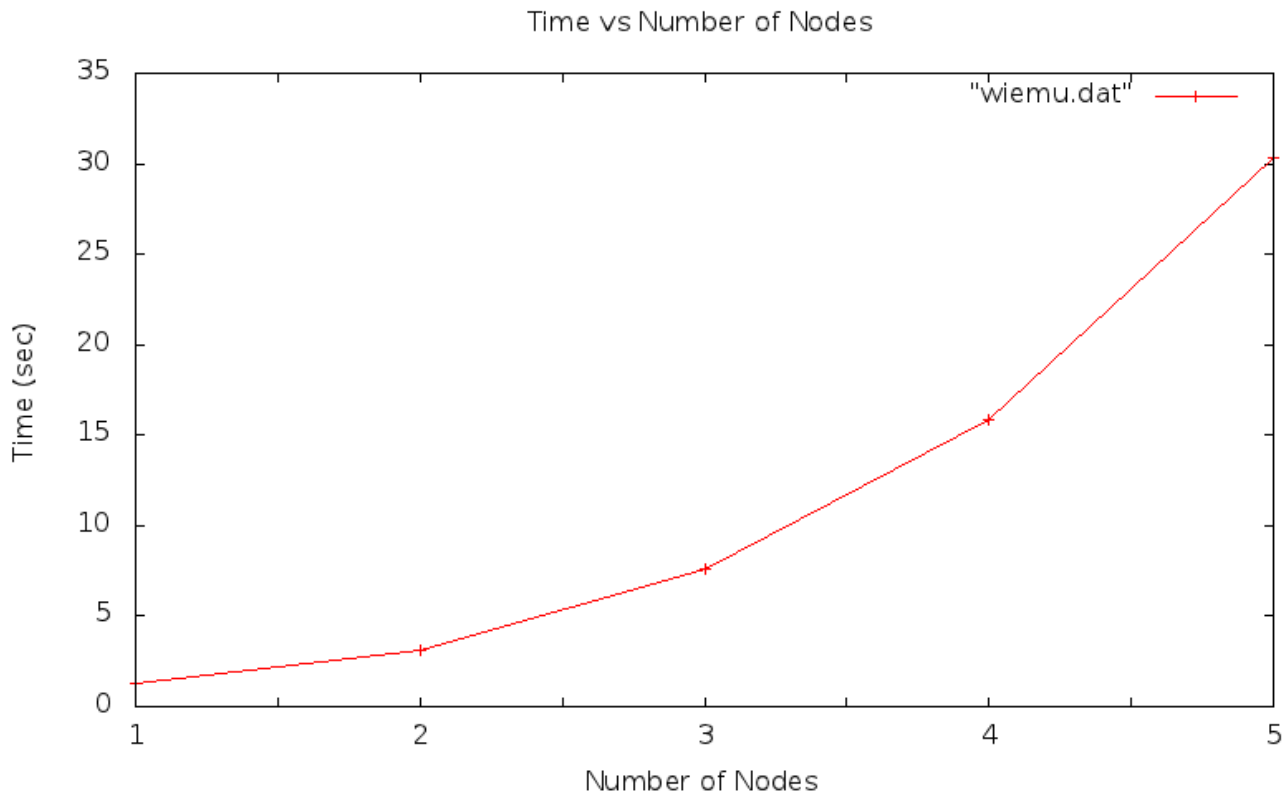
CPU-bound application (Prime Numbers Counter)



TinyOS-1.x Blink (Single Node)



TinyOS-1.x Blink (Multiple Nodes)



Conclusion

- 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

List of Publications

- Sherin M. Youssef, M. Abou El-Nasr and Mohamed Aslan, "WiEmu: The Design and Implementation of a Flexible Agent-based Scalable Network Emulator for Wireless Sensor Networks, " in the 2012 International Conference on Network and Computer Science (ICNCS 2012), Hong Kong, China, March 2012.

Thanks

Thanks for listening ☺