CS649
Sensor Networks
Lectures 30: WSN Simulation

Andreas Terzis
http://hinrg.cs.jhu.edu/wnsn06/
Overview

• The need for simulation
• Different simulation axes
  • High fidelity simulation of TinyOS Apps (TOSSIM)
  • Power consumption simulation
  • Simulation of hierarchical applications and emulation (EmTOS)
TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications

P. Levis, N. Lee, M. Welsh, and D. Culler
UC Berkeley
Presented at SenSys 2003
Outline

• The need for simulation
• TOSSIM Components
• Execution Model
• Hardware Emulation
• Radio Model
• Evaluation
• Discussion
The need for WSN simulators

- WSNs are inherently large scale distributed applications
- Large cost of deploying a network for testing
  - Potentially impossible
- Lowers the barrier for experimentation and research
- Parallel to experience from other disciplines
  - ns-2 experience in networking
Simulator Requirements

- Different levels of simulation
  - Algorithms vs. hardware-level simulation
  - What is the right model?
- TOSSIM requirements
  - Scalability
  - Completeness
  - Fidelity
  - Bridging (same code for real-life app and simulation)
TinyOS Refresher

- TinyOS characteristics
  - Component-based system
  - nesC
  - Component characteristics
    - Private storage
    - Commands and events
    - Tasks
  - Scheduling
  - Integration with hardware
- Challenges of writing TOS apps
  - Each component simple -> many components for complex apps -> large number of interactions
TOSSIM Overview

- TOSSIM Components
  - Compilation to sim code
  - Discrete event queue
  - Hardware abstraction components
  - Radio Model
  - External communications
Compiler Support

- nesC compiler can compile TOS apps to simulation code
- How?
  - Each node runs the same code
    - Code path is the same, need multiple copies of state variables
    - Components variables are replaced with an array, one copy for each simulated mote.
  - Plus additional plumbing for discrete event queue, network model and re-implementation of HW components
Execution Model

- Core simulator loops de-queues and executes next event from the event queue
  - HW Interrupts are modeled by simulator events
- Event is delivered to component that simulated HW component
- HW component posts events, commands to other components on mote it’s running
- All tasks from node’s task queue are executed until completion
  - Interrupt handlers cannot preempt tasks
  - Events, commands and tasks run instantly
  - Implications?
Hardware Emulation

- Simulated HW components
  - ADC
  - Clock
  - Flash
  - Radio
Radio Model (I)

- Model is a directed graph with bit error probabilities
  - Edge \((u, v, p)\): \(\text{Prob}[\text{bit sent by } u \text{ received at } v] = p\)
  - Bit errors are independent, asymmetric links are allowed
- Simulator delivers bit to all nodes within communication range of the sender
- Bit level simulation can provide a wide range of real-life scenarios (hidden-terminal, loss of start symbol, loss of ACK)
Radio Model (II)

- Simulating Radio Noise
  - Empirical model of packet loss as a function of distance
  - Derive Gaussian packet loss pdf for each distance
  - Per packet loss rate for node pair is determined by sampling the Gaussian distribution
  - Translate packet loss rate to bit loss rate

(a) Empirical
(b) Simulated
Communication Services

- Simulator can interact with external apps through socket interface
  - Sensing environment model
  - Visualization (TinyViz)
Evaluation

- Simulator was used to detect bugs in Surge application
- Noted the large number of packets were not delivered to the root
  - Monitoring showed that node send queue was overflowing
  - First bug: message received but ack lost. Surge would retx
  - Second bug: cycles (not including the origin)
Scalability

- Simulated three application types with varying level of network activity
- Performance Metrics
  - Run for ten simulation seconds
  - Memory footprint
    - 10,000 nodes → 4MB
  - Execution Time (10k nodes)
    - Network inactive app: 12.5 sec
    - Network-intensive app: 2.75 hours
Discussion

- Good Points
  - Single code image for simulation, deployment
  - Scalable

- Bad Points
  - Bit-level radio simulation (can also be good)
  - Single application
  - Difficult to change/extend hardware model
  - Instant execution of tasks
  - It’s still nesC 😊
Simulating the Power Consumption of Large-Scale Sensor Network Applications

V. Shnayder, M. Hempstead, B. Chen, G.W. Allen, and M. Welsh
Harvard University
Presented at SenSys 2004
Motivation

- Power consumption is major concern in WSNs
- Difficult to measure consumption in deployed networks
- Existing simulators
  - Do not simulate power consumption
  - Detailed simulation → Non scalable
- Goal: Extend TOSSIM to simulate power consumption
Hardware Power Model

- To simulate power consumption need to create power consumption model (Mica2)
- Approach: Use benchmarks to measure power consumption of actual mote

<table>
<thead>
<tr>
<th>Mode</th>
<th>Current</th>
<th>Mode</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Active</td>
<td>8.0 mA</td>
<td>Radio Rx</td>
<td>7.0 mA</td>
</tr>
<tr>
<td>CPU Idle</td>
<td>3.2 mA</td>
<td>Tx (-20 dBm)</td>
<td>3.7 mA</td>
</tr>
<tr>
<td>ADC Noise Reduce</td>
<td>1.0 mA</td>
<td>Tx (-19 dBm)</td>
<td>5.2 mA</td>
</tr>
<tr>
<td>Power-down</td>
<td>103 μA</td>
<td>Tx (-15 dBm)</td>
<td>5.4 mA</td>
</tr>
<tr>
<td>Power-save</td>
<td>110 μA</td>
<td>Tx (-8 dBm)</td>
<td>6.5 mA</td>
</tr>
<tr>
<td>Standby</td>
<td>216 μA</td>
<td>Tx (-5 dBm)</td>
<td>7.1 mA</td>
</tr>
<tr>
<td>Extended Standby</td>
<td>223 μA</td>
<td>Tx (0 dBm)</td>
<td>8.5 mA</td>
</tr>
<tr>
<td>Internal Oscillator</td>
<td>0.93 mA</td>
<td>Tx (+4 dBm)</td>
<td>11.6 mA</td>
</tr>
<tr>
<td>LEDs</td>
<td>2.2 mA</td>
<td>Tx (+6 dBm)</td>
<td>13.8 mA</td>
</tr>
<tr>
<td>Sensor board</td>
<td>0.7 mA</td>
<td>Tx (+8 dBm)</td>
<td>17.4 mA</td>
</tr>
<tr>
<td>EEPROM access Read</td>
<td>6.2 mA</td>
<td>Tx (+10 dBm)</td>
<td>21.5 mA</td>
</tr>
<tr>
<td>Read Time</td>
<td>565 μs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>18.4 mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Time</td>
<td>12.9 ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PowerTOSSIM Architecture

- Power Transition Messages are used to track the state of each HW component
- New PowerState component
- CPU consumption measured differently
- Derived energy model is used to calculate power consumption
- Processing can be done either off-line or through TinyViz
CPU Profiling

• Calculating PCU power consumption is challenging
  • TOSSIM compiles code into native PC binary
  • All tasks run instantaneously

• General idea
  • Measure the amount of time CPU spends in each power state
  \[ E_{CPU} = \sum_i P_{s_i} t_{s_i} \]
  • Instrument code to obtain execution count of each *basic block*
  • Map each basic block to assembly instructions in AVR binary
  • Determine number of CPU cycles for each basic block
  • Combine execution counts with corresponding cycle counts to obtain time CPU is active
Profiling example

App Code

```java
if(x>0) {
    t = x+42;
    v = t / pi;
} else {
    v = -1;
}
```

Transformed Code

```java
bb[mote][1]++;
if(x>0) {
    bb[mote][2]++;
    t = x+42;
    v = t / pi;
} else {
    bb[mote][3]++;
    v = -1;
}
```

Mote Binary

```java
if(x>0) {
    t = x+42; 2 cycles
    v = t / pi; 21 cycles
} else {
    v = -1; 1 cycle
}
```

Basic Block  Cycles

<table>
<thead>
<tr>
<th>Basic Block</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Method Accuracy

- Compared calculated CPU cycles to instruction-level simulator
  - Micro-benchmarks
- Compared calculated power consumption against actual implementation
  - TinyOS applications

<table>
<thead>
<tr>
<th>Task</th>
<th>PowerTOSSIM</th>
<th>atemul</th>
<th>error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash(char[27])</td>
<td>3002</td>
<td>2966</td>
<td>1.5</td>
</tr>
<tr>
<td>hash(char[404])</td>
<td>45415</td>
<td>49279</td>
<td>11.3</td>
</tr>
<tr>
<td>sort(int[100])</td>
<td>69581</td>
<td>92598</td>
<td>-33.0</td>
</tr>
<tr>
<td>encrypt(char[100])</td>
<td>103756</td>
<td>106096</td>
<td>-2.3</td>
</tr>
<tr>
<td>decrypt(char[100])</td>
<td>105075</td>
<td>107890</td>
<td>-2.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Simulated</th>
<th>Measured</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beacon</td>
<td>92.93</td>
<td>106.73</td>
<td>-12.9</td>
</tr>
<tr>
<td>Blink</td>
<td>940.26</td>
<td>931.72</td>
<td>0.85</td>
</tr>
<tr>
<td>BlinkTask</td>
<td>940.28</td>
<td>917.90</td>
<td>2.5</td>
</tr>
<tr>
<td>CntToLeds</td>
<td>1336.49</td>
<td>1330.00</td>
<td>0.45</td>
</tr>
<tr>
<td>CntToLedsAndRfm</td>
<td>2620.37</td>
<td>2562.00</td>
<td>2.3</td>
</tr>
<tr>
<td>CntToRfm</td>
<td>2028.09</td>
<td>1985.00</td>
<td>2.1</td>
</tr>
<tr>
<td>Oscilloscope</td>
<td>867.94</td>
<td>801.60</td>
<td>8.3</td>
</tr>
<tr>
<td>OscilloscopeRF</td>
<td>2136.45</td>
<td>2021.90</td>
<td>5.7</td>
</tr>
<tr>
<td>Sense</td>
<td>865.59</td>
<td>900.72</td>
<td>3.8</td>
</tr>
<tr>
<td>SenseLightToLog</td>
<td>2133.89</td>
<td>2005.26</td>
<td>6.4</td>
</tr>
<tr>
<td>SenseTask</td>
<td>865.62</td>
<td>944.74</td>
<td>-8.3</td>
</tr>
<tr>
<td>SenseToLeds</td>
<td>868.70</td>
<td>977.73</td>
<td>-11.1</td>
</tr>
<tr>
<td>SenseToRfm</td>
<td>2152.27</td>
<td>2059.16</td>
<td>4.5</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td>4.7</td>
<td></td>
</tr>
</tbody>
</table>

| TinyDB (idle)         | 2001.31     | 2275.55  | -12.1     |
| TinyDB (select light) | 2144.86     | 2465.30  | -13.0     |
| Surge                 | 2089.09     | 2028.40  | 3.0       |
| **Average**           |             | 9.5      |           |

Spring 2006  CS 649  24
Evaluation

- Simulated a number of TinyOS applications
- Findings
  - Idle CPU consumes most energy
  - LEDs are expensive

<table>
<thead>
<tr>
<th>Application</th>
<th>CPU idle</th>
<th>CPU active</th>
<th>Radio</th>
<th>Leds</th>
<th>Sensor</th>
<th>Board</th>
<th>EEPROM</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beacon</td>
<td>35.86</td>
<td>0.58</td>
<td>47.68</td>
<td>8.81</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>92.93</td>
</tr>
<tr>
<td>Blink</td>
<td>742.50</td>
<td>0.25</td>
<td>0.00</td>
<td>197.52</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>940.26</td>
</tr>
<tr>
<td>BlinkTask</td>
<td>742.50</td>
<td>0.27</td>
<td>0.00</td>
<td>197.52</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>940.28</td>
</tr>
<tr>
<td>CntToLeds</td>
<td>743.72</td>
<td>0.57</td>
<td>0.00</td>
<td>592.20</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1336.49</td>
</tr>
<tr>
<td>CntToLedsAndRfm</td>
<td>741.90</td>
<td>1.61</td>
<td>1284.65</td>
<td>592.20</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2620.37</td>
</tr>
<tr>
<td>CntToRfm</td>
<td>741.90</td>
<td>1.54</td>
<td>1284.65</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2028.09</td>
</tr>
<tr>
<td>Oscilloscope</td>
<td>742.65</td>
<td>1.46</td>
<td>0.00</td>
<td>0.00</td>
<td>123.82</td>
<td>0.00</td>
<td>0.00</td>
<td>867.94</td>
</tr>
<tr>
<td>OscilloscopeRF</td>
<td>741.90</td>
<td>1.85</td>
<td>1268.76</td>
<td>0.00</td>
<td>123.95</td>
<td>0.00</td>
<td>0.00</td>
<td>2136.45</td>
</tr>
<tr>
<td>Sense</td>
<td>742.21</td>
<td>0.38</td>
<td>0.00</td>
<td>0.00</td>
<td>123.00</td>
<td>0.00</td>
<td>0.00</td>
<td>865.59</td>
</tr>
<tr>
<td>SenseLightToLog</td>
<td>741.90</td>
<td>0.81</td>
<td>1262.95</td>
<td>0.00</td>
<td>123.95</td>
<td>4.28</td>
<td>0.00</td>
<td>2123.89</td>
</tr>
<tr>
<td>SenseTask</td>
<td>742.21</td>
<td>0.42</td>
<td>0.00</td>
<td>0.00</td>
<td>123.00</td>
<td>0.00</td>
<td>0.00</td>
<td>865.62</td>
</tr>
<tr>
<td>SenseToLeds</td>
<td>743.72</td>
<td>0.73</td>
<td>0.00</td>
<td>0.00</td>
<td>124.25</td>
<td>0.00</td>
<td>0.00</td>
<td>868.70</td>
</tr>
<tr>
<td>SenseToRfm</td>
<td>741.90</td>
<td>1.77</td>
<td>1284.65</td>
<td>0.00</td>
<td>123.95</td>
<td>0.00</td>
<td>0.00</td>
<td>2152.27</td>
</tr>
<tr>
<td>TinyDB (idle)</td>
<td>693.29</td>
<td>10.41</td>
<td>1181.78</td>
<td>0.00</td>
<td>115.83</td>
<td>0.00</td>
<td>0.00</td>
<td>2001.31</td>
</tr>
<tr>
<td>TinyDBApp (select)</td>
<td>742.85</td>
<td>11.20</td>
<td>1266.70</td>
<td>0.00</td>
<td>124.11</td>
<td>0.00</td>
<td>0.00</td>
<td>2144.86</td>
</tr>
<tr>
<td>Surge</td>
<td>727.28</td>
<td>1.50</td>
<td>1239.02</td>
<td>0.00</td>
<td>121.30</td>
<td>0.00</td>
<td>0.00</td>
<td>2089.09</td>
</tr>
</tbody>
</table>

Spring 2006    CS 649    25
Current Traces

(a) Measured

(b) Simulated
Scalability

- Compare PowerTOSSIM, TOSSIM and instruction-level simulator
Case Study: ESS Development

- Development process
  - Use of simulation, emulation, and real modes
  - Real mode used to validate vs. Emulation mode
- Analysis and performance measurement
  - Real mode packet traces with precise timing
  - Offline analysis of collisions and retransmissions
- In deployment
  - EmTOS integrates Multihop tree-building and transport protocol with microserver software
  - EmStar provides visibility and robustness for microserver code
  - Live topology data returned via transport layer piped to visualization tools