Run-time Monitoring of Energy Consumption in Wireless Sensor Networks

Southwest University of Science and Technology
Tao Yang
Nanyang Technological University
Yue Khing Toh, Lihua Xie

2007 IEEE International Conference on Control and Automation
Outline

• Introduction
• System Architecture
• Energy Consumption Estimation of CPU
• Voltage Monitoring and Power Model Adjustment
• Valuation
A wireless sensor network may consist of hundreds to thousands of nodes with finite battery energy as well as limited processing, sensing and communication capabilities.

The lifetime of such a network is determined by the battery capacity.
The energy consumption of an node cannot be determined before deployment because it is influenced by the environment and the interaction with other nodes.

For example, a head node in a cluster tends to consume more energy than other nodes.
Introduction

- TinyOS
  - Development platform and running environment for wireless sensor network applications.
  - Does not provide any built-in support for the nodes’ energy consumption monitoring.
Introduction

- Simulation environments—(predict the energy consumption)

- TOSSIM
  - Does not model all hardware details

- PowerTOSSIM
  - Does not model all interrupts and state transitions of the CPU, which results in a lack of accuracy in energy consumption estimation.

- AEON (has been embedded into AVRORA)
  - Aiding designers in their applications before deployment in real environments and thus lacks the capability of run–time monitoring of power consumption.
In this paper

- Based on the breakdown of the total energy consumption and the tracking of state transitions.

- Focus on monitoring the energy consumption of the MICA2 platform.
System Architecture

- TinyOS system, libraries, and applications are written in NesC.

- One exception is the scheduling module `sched.c`, which is written in the C language to allow direct handling of the pointers to task procedures.

- Each of the mote’s hardware components is driven by a specific module.
  
  - For example, the `CC1000RadioIntM.nc` module handles most of radio communication.
System Architecture

Figure 1. Architecture of run-time energy consumption monitoring
System Architecture

- The total power consumption per node can be decomposed into that of individual hardware components, including CPU, RF transceiver, LEDs and sensors.

- The energy consumption of individual hardware components is estimated using the time-stamped events of their state transitions and different states correspond to different power consumption.
These state transitions may be triggered by low-level events such as timer timeout events or high-level commands such as RadioControl.start.

These state transitions can be captured and logged by modifying the related TinyOS components.
System Architecture

- Upon each state transition, the current and the timestamp are logged and the accumulated energy consumption is calculated according to the power model and the duration of the previous state.
The MICA2 Motes come in three models according to their RF frequency band:

- MPR400 (915 MHz)
- MPR410 (433 MHz)
- MPR420 (315 MHz)

Our study focuses on MPR410 MICA2 motes.
In TinyOS applications, CPU goes to sleep whenever the task queue becomes empty.

Code lines have to be added to the sched.c module to track these transitions.

Example

- Added lines after the nesc_atomic_sleep instruction.
Energy Consumption Estimation of CPU

- POWER MODULE by

- Simulating the power consumption of large-scale sensor network applications—2004
- Accurate prediction of power consumption in sensor networks—2004

<table>
<thead>
<tr>
<th>Device/Mode</th>
<th>Current</th>
<th>Device/Mode</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (7.3728MHz)</td>
<td></td>
<td>Radio (433MHz)</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>7.8mA</td>
<td>Rx (Low Current)</td>
<td>7.4mA</td>
</tr>
<tr>
<td>Idle</td>
<td>3.3mA</td>
<td>Rx (Optimum Sensitivity)</td>
<td>9.3mA</td>
</tr>
<tr>
<td>ADC Noise Reduction</td>
<td>1.0mA</td>
<td>Tx (-20dBm)</td>
<td></td>
</tr>
<tr>
<td>Power-down</td>
<td>110μA</td>
<td>Tx (-16dBm)</td>
<td>7.1mA</td>
</tr>
<tr>
<td>Power-save</td>
<td>117μA</td>
<td>Tx (-12dBm)</td>
<td>7.6mA</td>
</tr>
<tr>
<td>Standby</td>
<td>227μA</td>
<td>Tx (-8dBm)</td>
<td>8.2mA</td>
</tr>
<tr>
<td>Extended Standby</td>
<td>233μA</td>
<td>Tx (-4dBm)</td>
<td>9.6mA</td>
</tr>
<tr>
<td>LEDs</td>
<td>2.2mA</td>
<td>Tx (0dBm)</td>
<td>10.4mA</td>
</tr>
<tr>
<td>Sensor Boards</td>
<td>0.7mA</td>
<td>Tx (2dBm)</td>
<td>12.8mA</td>
</tr>
<tr>
<td>EEPROM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>6.2mA</td>
<td>Tx (6dBm)</td>
<td>15.8mA</td>
</tr>
<tr>
<td>Write</td>
<td>18.4mA</td>
<td>Tx (8dBm)</td>
<td>20.0mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>26.7mA</td>
</tr>
</tbody>
</table>
Voltage Monitoring and Power Model Adjustment

- Supply voltage continues to drop.
- The sampling rate can be as low as once every hour because the supply voltage drops very slowly.

For example

- It takes about 172 hours for the supply voltage of two 2400mA AA batteries to drop from 3V to 2.1V while MICA2 application is running.

- We use ADC readings of battery voltage to adjust the values in the power model.
Voltage Monitoring and Power Model Adjustment

- The battery voltage can be calculated by

- **MPR /MIB – Mote Programming Board User’s Manual**

\[ V_{batt} = V_{ref} \times \frac{ADC\_FS}{ADC\_Count} \]

where:

\( V_{batt} = \text{Battery voltage} \)

\( ADC\_FS = 1024 \)

\( V_{ref} = \text{External voltage reference} = 1.223 \text{ volts} \)

\( ADC\_Count = \text{Data from the ADC measurement of channel 7} \)
Voltage Monitoring and Power Model Adjustment

- The adjustment can be conducted by

\[ \text{Power}_{k} = \text{Power}_{k} \times \frac{V_{\text{att}}}{3V} \]

- \( \text{Power}_{k} \) and \( \text{Power'}_{k} \) are power values for hardware component \( k \) with 3V supply voltage and after adjustment respectively;
**MICA2 motes operate until the supply voltage drops to the operating limit of **2.1V**.

**As shown in Fig. 3, the remaining lifetime can be determined by linear extrapolation as follows**

\[
E_{rem} = P_{cur}t_{rem} - 0.5R_{cur}t_{rem}^2
\]

\[
t_{rem} = \frac{P_{cur} \left(1 \pm \sqrt{1 - 2R_{cur}E_{rem}/P_{cur}^2}\right)}{R_{cur}}
\]

Figure 3. The Lifetime Prediction of Nodes
- The experimental setup includes two MPR410 MICA2 motes.
- Two Gold Peak 1300mAh rechargeable NiMH AA batteries
- Toggle LEDs every 1 second
- Transmit a packet with 20-byte payload every 0.03125 second by radio.
EVALUATION
- The measured lifetime of the MICA mote under test was 56.9 hours.

- Its accumulated energy consumption was 1201.7mAh, lower than the measured capacity of the batteries.

- Each update of accumulated energy consumption takes only 29–74 $\mu$s with an average of 48 $\mu$s, including the time used to get system time.

- Each adjustment of power model takes much longer: 1948–1977 $\mu$s
CONCLUSION

- We break down the total energy into that of individual hardware components.

- Developed a standalone TinyOS component, which captures both low-level events and high level commands and enables accurate estimation of the energy consumption.