

# Selectively Isolating Nodes on a Wireless Sensor Network to Reduce Battery Usage

David Drager  
Department of Computer Science  
West Chester University of Pennsylvania  
dd685492@wcupa.edu

## **Abstract**

Wireless sensor networks have evolved dramatically over the past decade. The decreasing costs of sensor nodes and the increase in the number of uses of these wireless networks has resulted in the numbers of deployed networks increasing dramatically. Although the technology and capabilities of these wireless sensors is always developing, battery capacity has not been able to keep up with the power requirements of these devices. Therefore, to extend the operating lifetime of these sensor networks, we must increase the efficiency of the nodes in relationship to the battery drain. This paper aims to develop a method to assist in increasing the battery efficiency of each node.

## **1. Introduction**

The ultimate goal of this project is to utilize game theory<sup>i</sup> to create the most effective routing mechanisms and incentives to increase the effective operating time of Wireless Sensor Networks. Other methodologies, such as setting a global sleep schedule<sup>ii</sup>, have been tested with some success in the past. Game theory is the practice of mathematically modelling competing parties towards a common goal. In our case, each node is considered a player and the goal of the game is to increase the efficiency of the battery, therefore extending the lifetime of the network but in some cases sacrificing the performing lifetime of an individual mote. We aim to set up the rules of this game and then find the Nash Equilibrium - the results of which maximizes the battery usage of the network as a whole. Our ultimate goal is to develop an algorithm and protocol where the motes will negotiate an optimal strategy for using battery usage while forwarding packets. This paper presents the first step towards this goal.

In order to apply Game theory to our future network, we must be able to isolate and control the operating parameters of each individual node. This is the first step in creating a framework that Game Theory can then utilize to control and operate the Wireless Sensor Network. Once this framework is in place we can then create the algorithm to apply to the network which will allow the nodes to utilize game theory

## **2. Problem Description**

We wish to be able to isolate a node and send out settings to that specific node and not the entire network. Any protocol we use must be able to distinguish between settings meant to be delivered to the entire networks versus ones that should only be applied to a specific node. Currently no existing program exists in this department that has the ability to remotely control the nodes of a Wireless Sensor Network from a root node. We wish to effect settings changes using a Java program running on a host PC, which will then relay commands to a specific node on the Wireless Sensor network. The Wireless Sensor Node may not have direct access to the root node, therefore a routing protocol must be in place to forward packets to these nodes.

## **3. Related Work**

We based our work using the TinyOS 2.x operating system for Wireless Sensor networks. TinyOS has many of the functions that we will use in this and future projects, and is very flexible as far as programming the flash rom on the motes and gathering data. The specific package that most of our code is based off of is the AntiTheft package, since this includes multihop routing and the ability to send out packets of data to the notes. TinyOS 2 uses network protocols<sup>iii</sup> named Dissemination and Collection for this purpose. Dissemination will

take data from a root node (the base station) and send it through the network. The Collection protocol will take data from the network and route it to the root node. We use these two protocols in our example below.

While developing this code, we also developed a VirtualBox<sup>iv</sup> image to facilitate the development of TinyOS code. This novel VirtualBox image includes Ubuntu 9.10 with the latest updates (as of publication date) along with related TinyOS code for 2.1 and 2.x versions, as well as supporting software such as Java, nesc, c++ and python compilers. This VirtualBox image will be made available on the web for public use.

For the implementation we have several options at hand. We chose to use the MicaZ hardware platform from Crossbow<sup>v</sup> to install our programs onto. The MicaZ platform is very stable and flexible and allows us to focus on the programming portion, while the nature of TinyOS makes our program available to compile on many other platforms such as Mica2 and Telosb.

#### **4. Implementation**

We took the code from the example package AntiTheft and added the following features:

- Per-node recognition of settings changes. This currently applies to the sensing interval but we can also change other variables on the node such as whether it will use certain sensors or other methods of detection.
- Java applet was enhanced to display data from each node. For this example, we used the light sensor reading, but we could use any reading which is available via the MTS300 board. We also wrote this data to a per-node log file for analysis.
- Java applet was modified to be able to input node and sensor delay value data, and to write this packet to the Root node which disseminates this to the rest of the network.
- Voltage reading support was added into each Node flash. This is implemented via the VoltageC connector with the following example code:

#### **AntiTheftAppC.nc:**

```
components new VoltageC();
AntiTheftC.ReadVoltage -> VoltageC;
```

#### **AntiTheftC.nc:**

```
interface Read<uint16_t> as
ReadVoltage;
...
call ReadVoltage.read();
...
event                               void
ReadVoltage.readDone( error_t
result, uint16_t val ){
    alert_t *newAlert = call
AlertRoot.getPayload(&alertMs
g, sizeof(alert_t));
    if (result ==
SUCCESS){
        newAlert->voltageData
= val;
    }
}
```

- Small modifications to other areas of code to change the sending of light and voltage readings over a set period of time, not just when a light threshold was reached.

After these changes were made, we installed the “Node” program onto 6 micaz motes, and the “Root” program onto 1 root micaz mote which was attached to the computer via the serial port.

#### **5. Evaluation**

To evaluate the effectiveness of the sensor read delay on each node, we set up six nodes with the modified AntiTheft Node code. The six nodes were allowed to run for 5 minutes to normalize any initial battery usage. We used varying battery manufacturers in the nodes. After 5 minutes, we sent commands to each node which effectively set the delay between reads, and therefore between each packet being sent to the root node. We expected that with increasing delay, the battery lifetime of the node would be extended.

The Java applet was modified to collect this data, both reporting it on-screen and also logging it to a file in a comma separated value format. The  $\Delta$  in Voltage is shown by taking

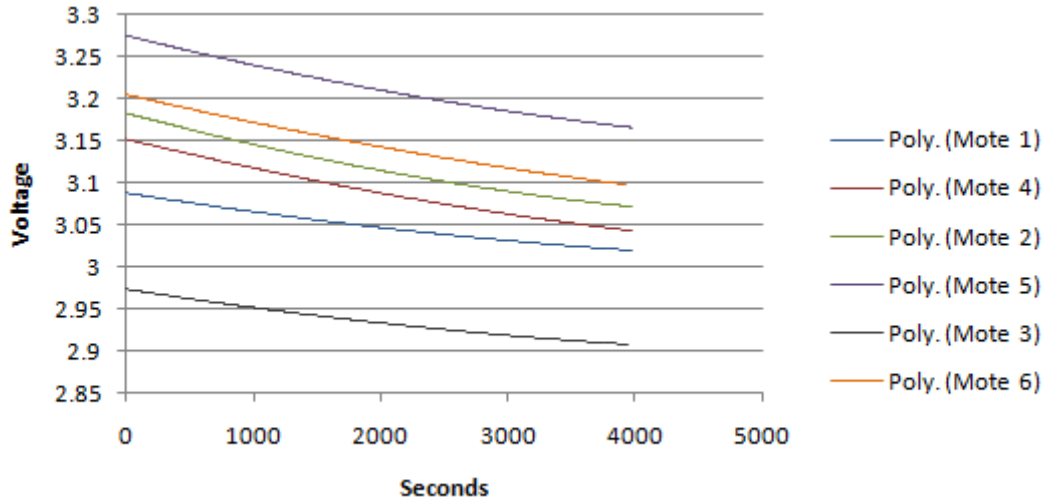


Figure a.

$V_{\text{initial}} - V_{\text{ending}}$ . The results from our experiment are shown below in Figure b.

Node ID	Delay	$\Delta V$	Manufacturer
1	1s	0.074	Energizer
2	10s	0.110	Duracell
3	60s	0.069	Energizer
4	1s	0.107	Duracell
5	10s	0.107	Duracell
6	60s	0.103	Duracell

Figure b.

The results were then imported into a spreadsheet and then graphed over time, as shown at the top of the page in Figure a. The lines represent the polynomial trend line of the data.

As you can see from the above data, we received mixed (and frankly, unexpected) results from running the experiment. In hindsight, the unexpected results were a result in the differences between each brand of battery used. The Energizer brand resulted in much less battery drain than the Duracell brand. When taking into consideration the brand, you can see that each mote with the changing delay has a differing voltage drop over the same period of time. We performed further tests on Panasonic

Alkaline Industrial brands with a similar result, the resulting voltage drops being even greater than the Energizer and Duracell brands. This should be taken into consideration when running future tests on battery drainage to ensure that the results from any tests will be comparable.

Furthermore, when you filter the above results according to battery brand, you can see that we have realized a small energy savings by delaying the time between readings on each sensor node. Being able to selectively turn the delay on motes on or off, and increasing these delay at will, gives us a slight energy advantage over the standard sampling period.

Finally the code that we wrote sets us closer to our goal of being able to apply Game Theory bargaining to each node. The nodes will be able to communicate with a main, coordinating node in order to both read and write settings and energy levels. We were successful in modifying the AntiTheft package to suit our needs for this project.

## **6. Conclusions and Future Work**

We found that adding a delay to the sensor time, does increase the battery savings of the nodes. While this result is not unexpected, it does confirm our belief that being able to control the timers on each node will result in considerable battery savings over time.

Now that we have the ability to configure values on each individual node, we can now create a 'payment' system<sup>vi</sup> where each node is allocated a certain number of fictitious credits in order to negotiate importance of packets that it is sending. We believe that a base node will be required to ensure the safety of the network and to enforce a routing table based on this system.

Further work will involve rewriting the AntiTheft code into a framework in which we can apply the above negotiation tactic onto, with nodes being individually controlled from an authorized root node but also have the ability to negotiate with neighbor nodes.

## **References**

---

<sup>i</sup> A Agah, K Basu, S Das; Security enforcement in wireless sensor networks: A framework based on non-cooperative games.

<sup>ii</sup> Lei Zhang, Somnath Ghosh, Prakash Veeraraghavan, Samar Singh, An Energy Efficient Wireless Sensor MAC Protocol with Global Sleeping Schedule, csa, pp.303-308, International Symposium on Computer Science and its Applications, 2008

<sup>iii</sup> TinyOS, [http://docs.tinyos.net/index.php/Network\\_Protocols](http://docs.tinyos.net/index.php/Network_Protocols)

<sup>iv</sup> VirtualBox, Sun Microsystems.  
<http://www.virtualbox.org/>

<sup>v</sup> CrossBow, <http://www.xbow.com/>

<sup>vi</sup> L Yan, S Hailes; Designing incentive packet relaying strategies for wireless ad hoc networks with game theory.