Measuring and Controlling Energy Consumption of Sensor networks

Bin Lu

08/01/2015

0.Introduction

A wireless sensor network (WSN) is built of two types of nodes: regular sensor nodes and base stations. The regular sensor nodes monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and cooperatively pass their data through the network to base stations. The base stations periodically transfer received messages to server. There are many applications of wireless sensor network, such as Health care monitoring, Environmental sensing, and Forest fire detection.

Since sensors are powered by batteries, energy efficiency is a key issue for wireless sensor network. Researchers have proposed lots of complicated energy-efficient MAC or routing protocols. However, the effectiveness of these protocols is usually validated through simulations but not real experiments: this is due to a lack of precise measurements of energy consumptions. Previously, researchers measure energy consumption by mapping voltage to energy. However, voltage to energy mapping is not injective, which means measuring the energy consumption precisely is impossible in this way.

In order to overcome the energy measurement problem, one project aims to develop a system which provides plug-and-play energy measuring hardware boards, easy-to-use local software, and user-friendly data management system on the server. As a sub-project, the purpose of this project is to implement the local software (server script and client script) and data management system (a website for displaying data). Researchers in sensor networks are able to easily use our system to carry out real experiments for re-evaluating many energy-efficient protocols.

Technical details of implementation will be discussed in this report. In the first section, we introduce simulation specifics of the wireless sensor network and how to run a simulation with our code. Second section presents the structure and deployment of website. The last section gives several simulation examples to demonstrate the efficiency of moving base stations routing algorithm.

1.Simulation of Wireless Sensor Network

A real WSN consists of sensors (regular sensors and base stations) and a remote server.

- Regular sensors generate messages periodically and transfer messages from other sensors.
- Activated base stations receive messages from regular sensors and send them to the remote server. Inactivated base stations act the same as regular sensors. All base stations should have synchronized clocks and retrieve routing information from server at fixed time points. The routing information will then be broadcast to all sensors.
- Remote server receives messages from base stations and send back routing information. The communication protocol between base stations and server is HTTP.

1.1 Global Parameters

All global parameters are set in setting.py module. They are:

host, port: IP address of remote server
sqladd, userid, userpsw, database: database address, user name, password, database name
initEnergyBS: initial energy(or cumulative current) of base station
initEnergyRS: initial energy(or cumulative current) of regular sensor
bsnum : number of base stations
rsnum: number of regular sensors
rst: time interval for regular sensor to send a message(e.g. 10 minutes)
bst: time interval for base station to send messages to remote server(e.g. 30 minutes)
rt : time interval for a network to update its routing strategy(e.g. 60 minutes)
cc, cst, csr, clc, clt, r: constant cost of sensor, short range sending cost, short range receiving cost, constant cost of activated base station, long range sending cost, message sending rate

1.2 Sensor Module (Client module)

Sensor behaviors are implemented in client.py module. For simplicity, we assume all actions are carried out at the same time and in sequence, i.e. every several minutes (say 10 minutes), regular sensors will send messages in turn and then they will transfer their messages in turn. Though in reality, the actions of regular sensors are independent, which means sending action are independent between sensors and transferring action is carried out as soon as message is received. But it will not cause much inaccuracy. Another assumption is that base stations will send messages to server and retrieve routing information from server at fixed time points. Routing information is then broadcast to all connected sensors. This can be achieved in a real network. In real WSN, every base station should have a synchronized clock and a routing round counter.

- Sensor Class: This class simulates the behaviors of regular sensor. It have three main functions, sending message, receiving message and transferring message. There is another function which simulates the action of receiving routing information.
- BaseStation Class: This class is inherited from Sensor class. It has two extra functions: sending messages to remote server and retrieving routing information from remote server.

1.3 Routing Module

Routing module generates routing table in each round. We have implemented four routing strategies.

- Random Routing: Randomly choose base stations to be activated and all the other sensors send messages along a random path.
- Use only one activated base station. Messages are transferred along the shortest path
- Use all base stations. Messages are transferred along the shortest path
- Moving base stations. Algorithm from the paper “Virtually Moving Base Stations for Energy Efficiency in Wireless Sensor Networks”

The only public method which will be called in the server module is *getRoutingTable* function. It returns a string containing routing table.

1.4 Server Module

This module implements a multi thread HTTP server.

1.5 Program Running and Extension

1.5.1 Package Installation and Database Initialization

To run the program, Python, MySQL and mySQLdb should be installed on the machine. The code is tested with Python 2.6, MySQL 5.1.73 on Ubuntu 10.04.1. Before simulation, we need to create four tables in our database with following schemas:

- *EnergyMonitor*[id(int, primary key), NodeID(int), Time(DateTime), Voltage(double), CumCurr(double), Data(varchar(40)), Experiment(int)]
- *ActiveBS*[id(int, primary key), Experiment(int), NodeID(int), StartTime(DateTime), EndTime(DateTime)]
- *monitor_sensor*[(int, primary key), last_connected_time(DateTime), initial_energy(double), remaining_energy(double), is_base_station(int), last_voltage(double), x(int), y(int)]
- *monitor_topology*[id(int, primary key), sensorID1(int), sensorID2(int)]

We also need to specify the parameters in setting.py. By convention, time intervals (bst, rst, rt) should be integers. Executing the command “python sqlwsn.py” will generate a random WSN.

1.5.2 Running a Test

Step1: open a command window, and redirect to project folder
Step2: execute “python server.py [*Test-id*]”. *Test-id* can 0,1,2,3, corresponding to the four routing strategies respectively.
Step3: open another command window, and redirect to project folder
Step4: execute “python client.py”

1.5.3 Extension

If we want to test other routing algorithms, server.py and route.py need to be modified. Concretely, in server.py, a unique test-id need to be assigned to the new algorithm. In route.py, method *getRoutingTable* needs to be recoded in order to return new routing tables according to test-id.

1.6 Other Specifications

Routing table sent by server has the format *[Node:NextNode,]*, for example “0:1,1:3,2:0,3:4,”, which means sensor 1 sends messages to sensor 3 , sensor 3 sends to sensor 4 and so on. From routing table, we can determine whether a base station is activated or not. If sensor *id* is included in the *NextNode* list but not included in the *Node* list, this sensor is an activated base station.

Message sent by sensors contains sensor id(1st byte), voltage(2nd and 3rd bytes), cumulative current(4th and 5th byte) and time stamp(remaining bytes, in the format “*yyyy-mm-dd hh:mm:ss”*).

2. Data Visualization

2.1 Website Structure

We design a website to display the data uploaded by sensors. The website consists of two independent applications: account management application and data monitoring application. Account management application is nothing special but a standard template, which is modified according to the requirement.

Monitoring application can be regarded as two parts: skeleton frame and content container. Skeleton frame is a modified template. Content containers are *home_iframe.html* and *linewithfocuschart.html*.

- *home_iframe.html* contains
  -- a table of sensor information
  -- a map of WSN network, which is implemented by exploiting a JavaScript package *Sigma*
  -- a form to add or delete sensors
- *linewithfocuschart.html* consists
  -- a form to choose start time, end time, sensors' id and test id
  -- a table displaying the activated time slots for all base stations. Package *Timeline* is exploited.
  -- two charts displaying voltage and cumulative current respectively during the chosen period. Package *NVD3* is exploited.
  -- tables to compare cumulative current decreasing rates between different tests

All data, which is required for rendering these pages, is retrieved and pre-processed in /energysense/monitor/sqlweb.py module.

2.2 Website Deployment

To deploy our website, Django, Apache server and django-nvd3 need to be initialized on the machine. In the project, we use Django 1.6 and Apache 2.2.14. Database needs to be specified in file /path/to/energysense/energysense/setting.py.

First, we need to generate tables in our database by executing "python manage.py syncdb". *manage.py* locates in the project folder.

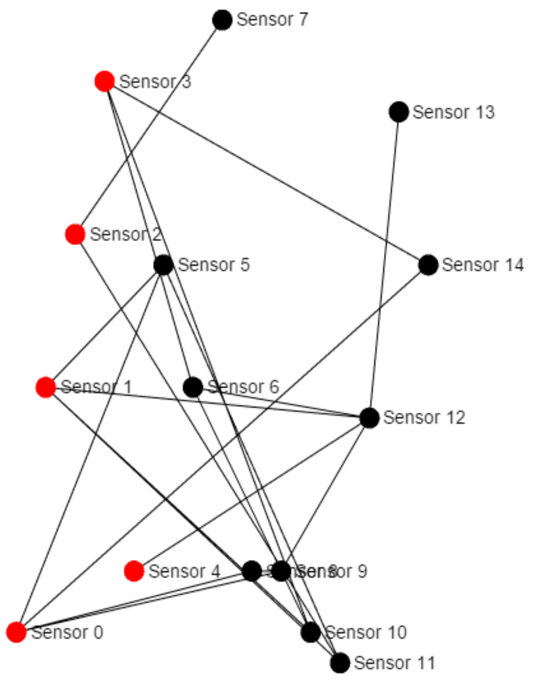
Second, to run the website on apache server, file /etc/apache2/httpd.conf should be modified as follow:

```
WSGIScriptAlias / /path/to/energysense/energysense/wsgi.py
WSGIDaemonProcess 179.43.141.181 python-path=/path/to/energysense
WSGIProcessGroup 179.43.141.181

Alias /static/ /path/to/energysense/static/
<Directory /path/to/energysense/static>
```

```
Order deny,allow
Allow from all
</Directory>

<Directory /path/to/energysense/energysense>
<Files wsgi.py>
Order deny,allow
Allow from all
</Files>
</Directory>
```

After modifying the file, we need to restart server by executing “sudo restart apache2”(or “restart apache2” if one logins as root). Now, our website can be visited on the browser through IP address 179.43.141.181.

3. Example



Wireless Sensor Network with 5 base stations and 10 regular sensors. Parameters are assigned with following value:

initEnergyBS = 30000
initEnergyRS = 18000
rst = 10
bst = 30
rt = 60
cc,cst,csr,clc,clt,r = 6,1,1,60,1,6
alpha = 10000

Test 1: Randomly Choose Activated Base Stations, Shortest Path Routing

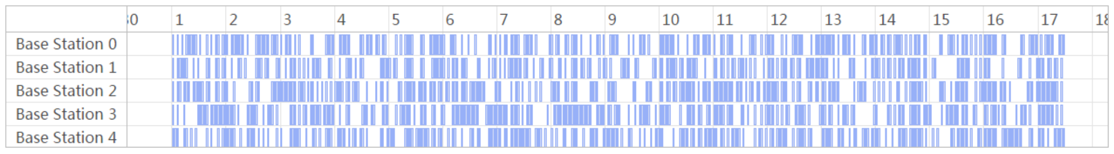


Fig 3-1 Activated Time Slots of Base Stations in Test 1

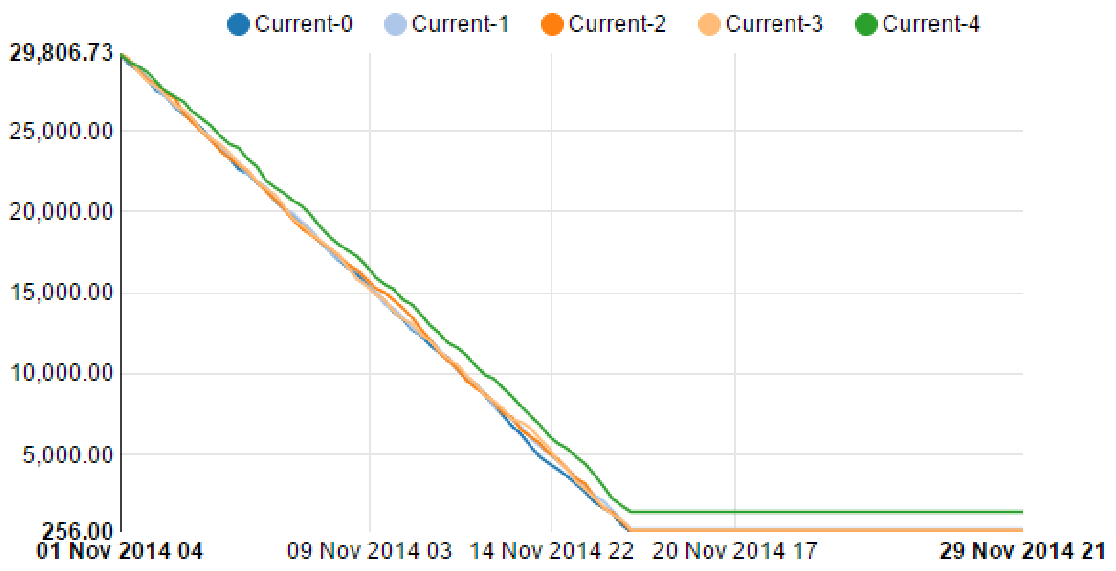


Fig 3-2 Cumulative Current of Base Stations in Test 1. Base station 3 runs out of energy after around 395 hours

Test 2: Active One Base Station, Shortest Path Routing

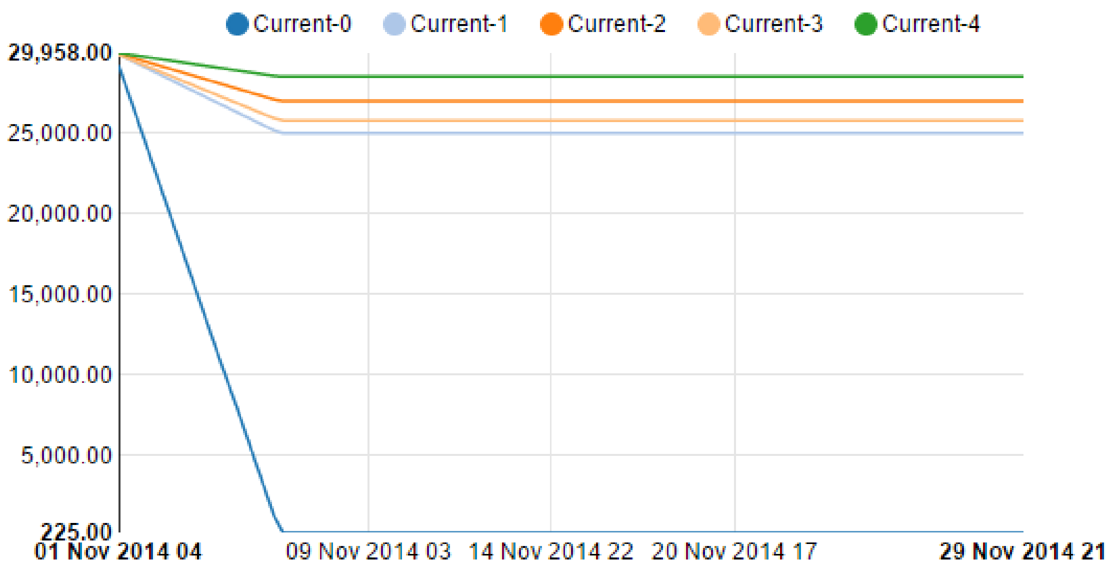


Fig 3-3 Cumulative Current of Base Stations in Test 2. In this test, we choose base station 0 to be activated. After 127 hours, base station 0 runs out of energy.

Test 3: Activate All Base Stations, Shortest Path Routing

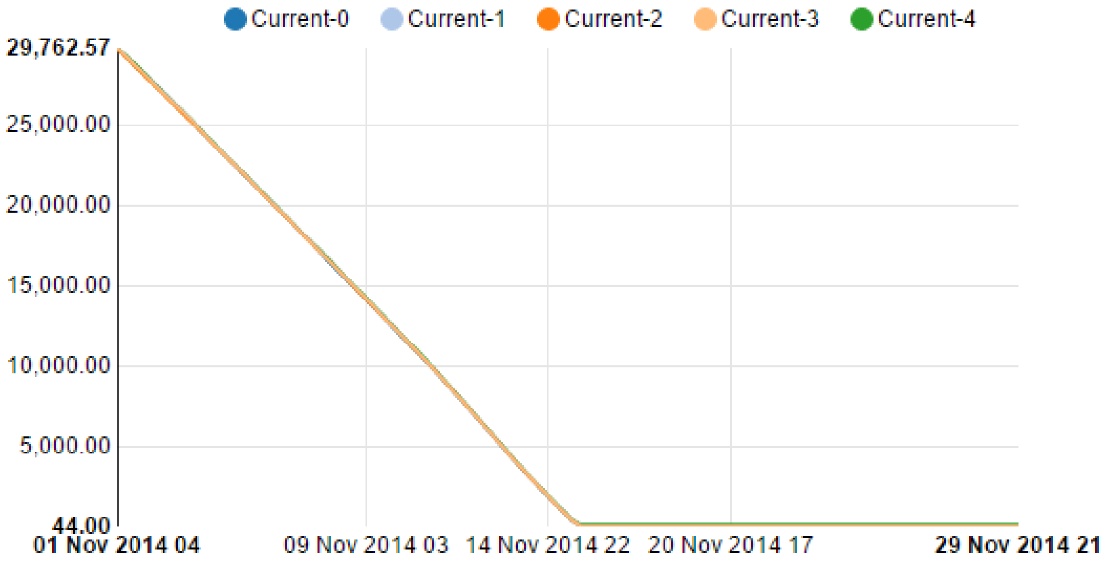


Fig 3-4 Cumulative Current of Base Stations in Test 3.

All base stations runs out of energy after around 356 hours.

Test 4: Moving Base Stations, Shortest Path Routing

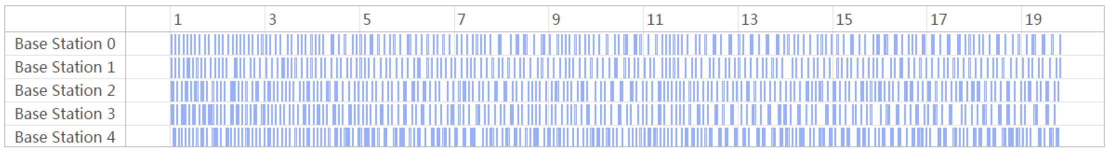


Fig 3-5 Activated Time Slots of Base Stations in Test 4

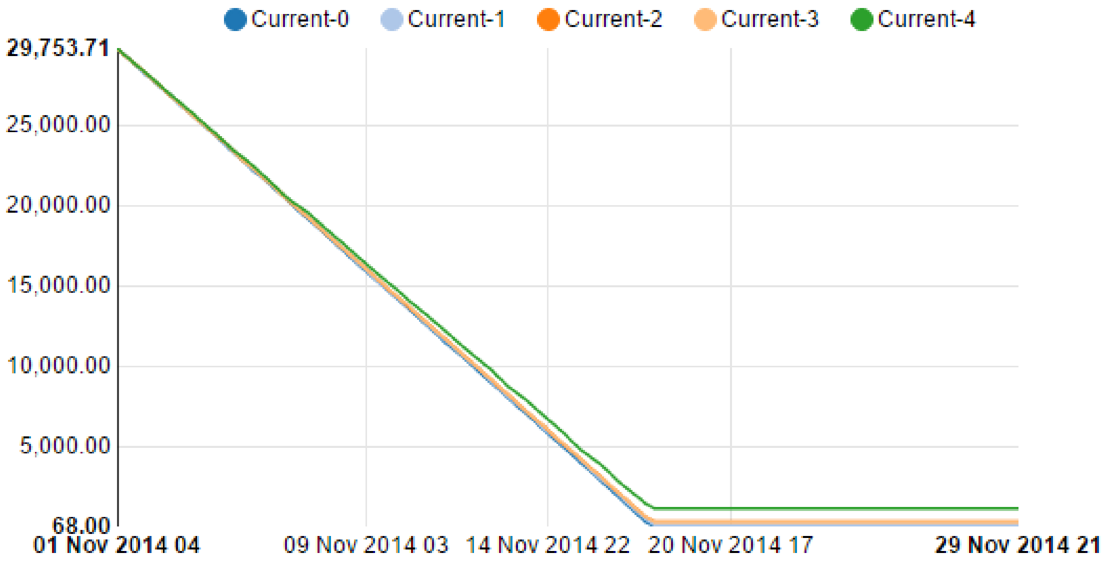


Fig 3-6 Cumulative Current of Base Stations in Test 4. Base Station 0 runs out of energy after 413 hours.

Result Analysis:

Routing Algorithm	total hours before some node(s) run out of power
Random Routing	395 hours
One Base Station, Shortest Path Routing	127 hours
All Base Stations, Shortest Path Routing	356 hours
Moving Base Stations	413 hours

In test 4, the remaining energy of all base stations is at the same level. Compared with other strategies, moving base stations strategy evenly distributes working load over all base stations.

Reference

[1] Virtually Moving Base Stations for Energy Efficiency in Wireless Sensor Networks
[2] Dargie, W. and Poellabauer, C., "Fundamentals of wireless sensor networks: theory and practice", John Wiley and Sons, 2010 ISBN 978-0-470-99765-9, pp. 168–183, 191–192