Volume 49, Number 1, 2008 **ACTA TECHNICA NAPOCENSIS**
Electronics and Telecommunications

A MULE BASED STRATEGY FOR WIRELESS SENSOR MOTES TO REDUCE POWER CONSUMPTION

Adina ȚOPA, Anghel-Vasile CONȚIU, Vasile Teodor DĂDÂRLAT
Technical University of Cluj-Napoca, Str. C-tin Daicoviciu nr. 15, 400027 Cluj-Napoca, Romania
E-mail: Vasile.Teodor.Dadarlat@cs.utcluj.ro

Abstract: The paper presents an application for wireless sensors, which focuses on reducing power consumption of the wireless sensor devices. The project uses a new technology, having a very big potential: motes called Tmote developed at Moteiv. The goal of this application is to investigate the use of the Tmote nodes as mules, to implement a power consuming strategy, so that the sensors deployed through the network will consume as little power as possible.

Key words: *wireless sensor motes, power consumption strategy, data collection*

I. INTRODUCTION

The research area of the wireless sensor network has witnessed an increased attention in last few years, making it one of the most important technologies of the 21st century. The wireless microsensor comes with a cheap and smart idea: now one can create a wireless network using the Internet and these small devices with multiple onboard sensors on them. Because they are low-cost, low power and multifunctional, one can deploy a large number of devices, creating a network big enough to be able to control the environment [1]. The position of the sensors nodes does not need to be engineered or predetermined, so these devices can be placed in inaccessible terrains or disaster relief operations. Sensor nodes are fitted with an onboard processor, so they are capable to process locally simple computations and transmit only the required and partially processed data. [2]. The sensor boards, called motes can gather large range information, from temperature or light to velocity.

In our project we used motes developed at Moteiv and called Tmote [10]. They have a special operation system called TinyOS. The applications that can be deployed on them are developed in NesC [6], which was especially built for these motes and which is an extension of the well-known programming language, C/C++. The idea behind the project is to create a power consumption strategy, so that the sensors deployed through the network will consume as low power as possible. Because the sensors can be spread all over the area that we want to investigate, some sensors can be placed in rather inaccessible terrains, in which case only the batteries will generate the sensor's power and also the gathering of data directly from them, e.g. using your PC, is unpractical. The problem of collecting data was resolved using a mobile mote, called mule.

The paper is devoted to the investigation of Tmote nodes and is constructed as follows: Section 2 presents the architecture of the application, Section 3 the power reduction strategy. Next some experimental results are discussed in section 4 and in the last section conclusions are drawn.

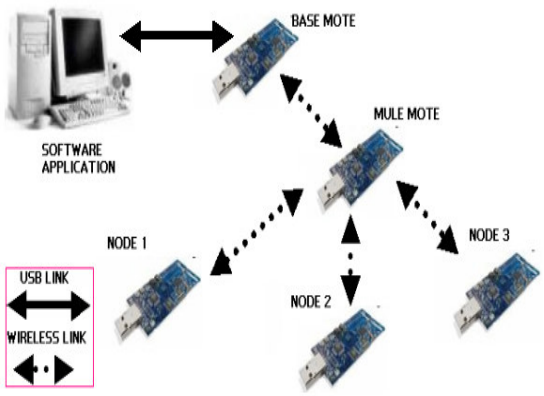

Figure 1. The architecture of the Mule Project

II. THE ARCHITECTURE OF THE PROJECT

The architecture of the project is depicted in Fig. 1. The mule mote moves around the network, so that every mote can be reached once every round. Basically, the mule's job is to make a logical connection between a base mote and all the sensors in the network. All the information gathered by the mule will be dumped on the base, which is a mote directly connected to a PC. The computer runs an application that makes the user view all the information transmitted through the network. The base will be the one

Manuscript received February 19, 2008; revised April 17, 2008

issuing the requests that the sensor motes must interpret and execute, but it will also be responsible for receiving the information from the mule and forwarding it to a PC.

In order to fulfill the communication task, we developed different applications for the motes, corresponding to the three types of motes: base, mule and sensor. The project consists of the following applications:

- The UsbUserbuttonDeluge application, written in NesC;
- The PC application – it is an intermediate between the user and the motes, written in Java;
- The applications for the sensors written in NesC– they make the transition between the wireless sensor motes and Java.

II.1 THE USBUSERBUTTONDELUGE APPLICATION

This application has been developed for giving the user the ability to switch as easily as possible from running one of the three applications (the base, the mule and the sensor) to another one, because at one point only one application can run. It uses the Deluge component, which is part of the NesC components and which permits the deployment of more than one application on a mote. The transition between different applications called images is done by the USBDetectExtendedC component.

When a mote is running the UsbUserbuttonDeluge application, the component will verify whether the mote is connected to USB or whether it is working on batteries. By default, the mote will upload the image for the base if the mote is connected to the USB; otherwise, the mote will upload the image for the sensor. We have also given the possibility to change the default images by pressing the user button which resides on the mote:

- For red light, the image of the base application will be loaded on the mote;
- For blue/yellow light, the image of the mule application will be loaded on the mote;
- For green light is on, the image of the sensor mote application will be loaded on the mote.

On every mote there is a 4th image uploaded, which is used for installation and debugging purposes. After the user has selected the application that he wants to run on the mote, a timer will start. As long as the timer does not expire, the user can change the image to be uploaded on the mote.

II.2. THE PC APPLICATION

The main purpose of the PC application is to be an intermediate between the user and the motes. It is a Java application, installed on the PC, which must be directly connected to the base mote via USB. This application has a graphical interface, which permits the user to send new requests for different nodes from our network and also visualize the data received by the mule from them (*see Fig 2*).

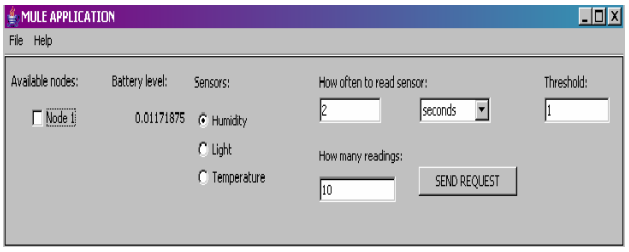

Figure 2. Screenshot of the Java Application

II.3. THE APPLICATIONS FOR THE SENSORS

The base application will make the data transfer between the wireless sensor motes and the Java application (see Fig. 1). The functions of a base are:

- Forwarding to the mule the requests created by the user in the Java application;
- Forwarding to the Java application all the information received from the mule, like:
  - Information about the state of the sensor motes from our network;
  - Data collected by the sensor motes from our network.

The mule application is the application which makes the communication between the base and the sensor motes possible, it logically connects them. It will store all the requests received from the base and also all the data received from the sensors, in order to forward it to the corresponding devices.

The sensor motes are the only motes that are gathering information from their sensors, so in fact they are the only motes that need to have sensors on them.

III. POWER CONSUMPTION STRATEGIES

One major issue for the sensor application is the power consumption. In order to reduce it as much as possible, we have taken into consideration two components: the processor and the radio.

III.1. PROCESSOR APPROACH

The processor of the Tmote device, MSP430, was designed in such a manner that it will automatically go in low power mode when no tasks are in the queue. MSP430 has defined five low power modes for the processor: LMP0 – LMP4 and one active, normal power mode. The operating modes take into account three different needs:

- Ultralow-power;
- Speed and data throughput;

- Minimization of individual peripheral current consumption [3].

The MSP430 typical current consumption is shown in Figure 3.

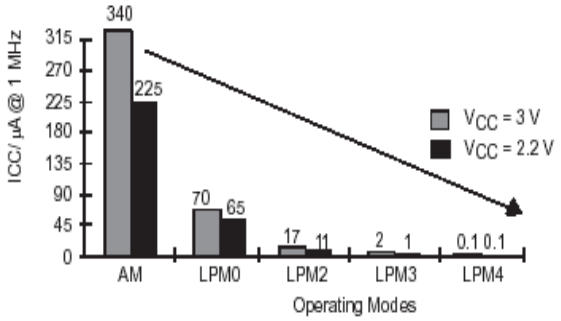

Figure 3. Typical Current Consumption of 13x and 14x Devices vs Operating Modes

Often, the most important factor for reducing power consumption is using the MSP430's clock system to maximize the time in LPM3. LPM3 current consumption is less than 2 µA typical with both a real-time clock function and all interrupts active. A 32-kHz watch crystal is used for the ACLK and the CPU is clocked from the DCO (normally off) which has a 6-µs wake-up.

III.2. RADIO APPROACH

The radio uses most of the power of all the Tmote device's components. Typically, the current consumption in receive mode is 19.7mA and in transmit mode it can go from 8.5mA (P=-25dBm) to 17.4 mA (P=0dBm) [4].

The sensor mote will have to use the radio as little as possible, so we have developed our sensor application that it turns off the radio as long as no mule is visiting him; the sensor mote will only turn on the radio when it knows that the mule should be coming. This information is received from the mule in the message which encapsulated SetWhen2WakeUpMsg_t and which is now a fixed value.

III.3 INTEROPERABILITY BETWEEN APPLICATIONS

The communication between the user and the sensor motes is done in two major steps:

- Communication between the PC and the base mote;
- Communication between the different types of motes.

The user does not know how the network looks like; the only interaction with the network is done using the PC. The PC is the tool used to visualize the data exchanged in the network and also to create new requests for the sensor motes to interpret.

For the implementation of the communication between the three types of devices, we have made the following assumptions:

- The user must press the user button of the mule, in order to send all the information back to the base;
- The base application will continuously send the requests to the mule, so that the mule receives all the requests issued by the user;
- The mule can store only a limited number of requests;
- The sensor can deal only with a limited number of requests;
- The mule has a fixed route;
- The mule will tell the sensor when it will visit it again, which is an invariable value.

We have developed an algorithm for data compression so that the sensor sends as few information as possible. For achieving this, we have used two arrays: valueArray (contains the values of the readings) and bitArray (is a mask that encrypts the values read from the sensor).

These arrays are created using the following steps:

- We take the first (r1) and the second (r2) values from the readingArray:
  - If r1 is equal to r2, then in the bitArray, in the first and second position we place the same binary value (1/0);
  - If r1 is not equal to r2, then inside the bitArray, in the first and second position we place different binary values;
- We take next look at the third value read from the sensor (r3):
  - If r1 is equal to r2, then we compare r3 with r1 and apply the same algorithm as before;
  - If r1 is not equal to r2, then we compare r3 with r2 and apply the same algorithm as before;
- We continue this algorithm for all the values read from the sensor;
- Next, we will take all the consecutive values from the readingArray that have the same values in the bitArray and compute their average. This value will be added in the valueArray. This way, we will not send approximately the same values, making the packets sent from the radio smaller.

We consider that two readings have the same value, if the absolute difference between them is below a threshold!

Example: We have 10 readings and the threshold is 2.
→ The list of bits will look as follows:

Reading	10	11	9	20	19	20	21	23	23	22
bitArray	1	1	1	0	0	0	0	1	1	1

→ readingArray: 10, 20, 23

IV. EXPERIMENTAL RESULTS

The tests have been done on tmote devices [5] and on a PC with the following configuration: AMD Athlon 64 Processor, 2.1 GHz and 512 RAM.

For the measurement of the temperature, humidity and light, we have done the following tests:

A. The sensors (temperature, humidity and light) must be read every 2 seconds and the minimum number of samples that must be sent back to the base is 3;
B. The sensors (temperature, humidity and light) must be read every 2 seconds and the minimum number of samples that must be sent back to the base is 5.

		CASE	
		A	B
Results	Temperature	26.1 26.1 150.1	26.1 26.3 26.5 26.9 155.5
	Light	5.5 5.3 1549.3	5.5 5.5 5.5 5.3 1543.0
	Humidity	13.6 13.3 -6581.9	12.5 12.5 12.5 12.6 -6441.0

Table1. Test results

As we can see from the results shown in Table 2, the last data from every package is corrupted.

V. CONCLUSIONS AND FUTURE WORK

This project is a very good approach of the mule problem. It can be used for monitoring different environments, even if the allocated budget is not very big. The project uses a new technology, which is not very known, but which has a very big potential. It also deals with some issues that are very important for this field: power consumption. There are papers that dealt with the mule problem, but they are all theoretical and from our knowledge nobody has tried to implement them.

One of the biggest advantages of the whole project is that the user of this application does not need to have any knowledge about NesC [6], TinyOS [7] or wireless sensor networking when using it. Everything is transparent to him. He just needs to know how to run the graphical interface (from the cygwin[8] bash shell) and how to deal when the mule arrives in the range of the base, by pressing the user button. The only person that needs to have some basic knowledge about TinyOS or NesC is the person that has to install the applications on the wireless sensor motes.

For the future work we would like to implement an algorithm so that the sensor learns when its data might change and only then, we should begin to read data from his sensors. This way the data will only be transmitted when new information is being read from the sensors and so the radio will be used even fewer than before. We would also want to implement a security algorithm, so that the transmission between our entities will be as safe as possible.

REFERENCES

[1] Chee-Yee Chong; Kumar, S.P., *Sensor networks: Evolution, opportunities, and challenges*, Proc IEEE, August 2003, p.1-10.
[2] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci - *A Survey on Sensor Networks*
[3] *MSP430x1xx Family, User's Guide* (available at http://focus.ti.com/lit/ug/slau049f/slau049f.pdf)
[4] *CC40 Data Sheet*
[5] *CC2420 Data Sheet* (available at http://www.btnode.ethz.ch/pub/uploads/Projects/CC2420_datasheet.pdf)
[6] *Moteiv documents* - /cygwin/opt/moteiv/doc/nesdoc
[7] *NesC: A Programming Language for Deeply Networked Systems* (available at http://nescc.sourceforge.net/)
[8] www.tinyos.net
[9] www.cygwin.com
[10] Tmote-sky-datasheet (available at www.moteiv.com)