Spotlight: Focusing on the Individual

Zainul Charbiwala, Younghun Kim, Akhilesh Singhania

***Abstract*— We present an application that can monitor the electrical energy consumption of an individual in an office setting with sharable and non-sharable appliances. A sharable appliance is one that multiple personnel can use at the same time like lights in a lab, television, etc. A non-sharable appliance is one that only one person uses at a time like microwave, coffee-maker, etc. All non-sharable appliances are instrumented to consist of a power-meter and a RFID reader. The user carries a RFID tag which is read by the instrument when it is in use by the user. Each sharable appliance has a power meter and a defining border with controlled entry and exit points. Personnel entering and exiting the range are detected. The power consumption is equally divided between the people sharing the appliance. Our results show interesting individual behavior that could be used for various energy optimizations.**

I. Introduction

Presently all consumption monitoring is at the granularity of when the energy was used and where it was used. This measurement profile is very coarse and there is no notion of who used this energy. As energy cost increases, it is becoming important to monitor the consumption at a finer granularity like at the level of individuals. This will allow organizations to optimize energy consumption in various ways. Employees can be assigned "green" points based on the amount of energy they consumption and can offer incentives to decrease their consumption. When this information is combined with the information of where and when, organizations can better pinpoint areas of energy wastage and can optimize the consumption in certain places.

We attempt a basic attempt at monitoring individual consumption profile and offer some analysis of issues in solving this problem. Our results show that it is possible to carry out such profiling at an individual level with reasonable assumptions. Using our results a more sophisticated and detailed application can be constructed which would carry out such profiling for more appliances and also collect other data.

This document is structured as follows. Section II describes our introductory application and some of its limitations. Section III discusses the methodology and device used for power measurement. Section IV describes the RFID system used for identifying users and Section V details the back-end system used to collect and post data to a central repository. Section VI describes the R algorithms used to analyze the data to construct a consumption profile of a user. Section VII illustrates the final deployment scenario and we offer some results in Section VIII. Finally we detail some related work in Section IX and conclude in section X.

II. Application

We discuss our introductory application and some of the limitations of the application.

A. Application Details

We had a number of constraints in developing our application. We had only two power-meters are our disposal and we were short of the number of appliances that we can use. Therefore, we instrumented the coffee-maker and the soldering station in our lab. These two will serve as the non-sharable appliances as only one person can use it at any time. Both consist of a RFID reader and a power-meter. The reader sits on top of the appliance and is connected to a mote. The power-meter monitors the energy flowing into the appliance and is also connected to another mote. Both appliances have two motes each.

We monitor the lights in our lab as the sharable appliance. We are unable to hook a power-meter to the lights fixture so we will estimate the energy consumption of them. The range of the lights appliance is defined to be the entire lab. So anyone present in the lab is sharing its consumption. The entry and exit points of the lab are the doors. We established one RFID reader at each door. This collects the RFID reading on people whenever they enter or exit the lab. Then using a algorithm on the back-end, we will determine when a person is entering or leaving the lab. Our lab has two doors so two RFID readers were required to instrument the doors.

The motes on the appliances collect the RFID or power-meter data and transmit to the base-station. Since the deployment for this lab spanned a vast area, we had to use an additional mote to serve as a hop for all the motes to be able to send data to the base-station. We used an additional mote to serve as a base-station. The base-station is connected to a Gumstix which sends all the collected data to SensorBase using an Ethernet Internet connection.

We use SensorBase to aggregate all our data. The data is sent to it through the base-station. Then we can probe SensorBase for the collected data on the back-end where we run different algorithms written in R to compute the individual power consumption. The total amount of hardware used in the application is eight motes, two power-meters, four RFID readers, one Gumstix and numerous RFID tags.

B. Limitations

There are certain limitations with our application which will have to be addressed for it to be used successfully in a real world deployment. One fundamental limitation steams from our division of appliances into sharable and non-sharable. There are certain appliances that do not necessarily fall in either category. A good example of this is servers and computers. Servers have a complicated consumption profile which to a certain extend tends to be independent of the usage profile of it. In order to properly evaluate the individual profile for a server, we need to monitor a log of which user is actively

Fig. 1. Veris Hawkeye 8035

Fig. 2. Watts Up? Pro

running tasks on the system. Presently, we are monitoring the printer by assuming that the user performing the print job will pick up the work shortly afterward. If another person initiates another print job before the previous one is picked up, the profile will be incorrect. One way of improving this is to use print logs to determine the profile instead of using RFID tags. The printer in our lab does not maintain any such information but in a general setting, it will be possible to maintain such profiling.

Presently, there is no policing in the usage profiling. Anyone can use the appliance and as long as they do not register their RFID tags, it will not be counted against them. Some ways of improving this is to instrument the appliances such that a RFID registry is required in order to activate them. This would require significant effort and is beyond the scope of this document but easily implemented in a general setting. Finally, we did not instrument the light fixtures in out lab to retrieve consumption data. We simply profile the the details of the lab occupancy to model our sharable appliances.

III. Power Measurement

A. Power Meter Selection

Two units for measuring power consumption have been studied - Veris Hawkeye 8035/8036 and Watts Up? Pro.

The Veris H8035/8036 device shown in Figure 1 is a CT based power meter with an RS-485 interface featuring a MODBUS Networked kW/kWH Transducer. The H8035 and H8036, three-phase power transducers monitor energy parameters from Aggregate kW (demand) and kWH (consumption) to power factor per phase. The H8035/36 Series combines a microprocessor based power meter and split-core instrument grade current transformers (CTs) in a single unit. The meter has the ability to sense up to 100 Amps of current with 100mW precision at a 6 Hz sampling rate.

The Watts Up? Pro device shown in Figure 2 is a consumer friendly power meter that has a USB interface and Windows based software to read the meter. One simply requires to plug any device into Watts up?, and the meter instantaneously displays the wattage being used. The meter can be used in internal logging mode where it records all the data into non-volatile memory. The sample rate is user selectable or can be automatically set (resolution decreases over time as memory fills up). In addition, the meter is capable of real time external logging to a host interfaced with the meter. The communication protocol is well documented, including a list of commands required for real-time logging. The maximum sampling rate of the Watts Up? Pro is once per second, with a 100mW precision.

The device chosen for current implementation of Spotlight is the Watts Up? Pro. There are a number of reasons for this. Firstly, the Veris device needs to strip off the cable cladding to get the clamps on the wires whereas the Watts Up? is plug and play. Using the Veris device leads to installation hassles especially when one would like to characterize energy consumption of consumer end appliances. Second, the Veris device does not have any display, whereas the Watts Up? has a simple built-in LCD display. Usage of the LCD reduced development time of the communication protocol and drivers considerably. Thirdly, the Veris device has an RS485 interface for which an independent hardware interface circuit would have to be built and tested before any other development could begin. The interface for Watts Up? is USB, with PC software provided by the manufacturer. This feature expedited the development of the project since it was possible to parallelize various parts of the implementation. Also, no extra hardware interface circuit would be required, further reducing development time. In addition, the protocol and code for the Watts Up? Pro are both available from the manufacturer. Last but not the least, the Veris device is over 6 times more expensive than the Watts Up? Pro.

B. Hardware Interfacing

The only initial concern in using the Watts Up? Pro for Spotlight was the USB interface. The USB interface poses a problem for the project since the mote being used, the MicaZ, does not have a USB interface. To circumvent this, one idea was to use a mote with a USB host interface, the iMote2. However, a quick review revealed that the pins exposed for the USB host are present on the advanced 40-pin connector but to use it would require to build a small custom board. To avoid the delay associated with the custom board design, we sought another solution.

When we studied the Watts Up? Pro device internals, we discovered that the USB interface was implemented using a common UART to USB part from Future Technology Devices International (FTDI) - FT232RL. It seemed possible to bypass the USB host problem entirely by intercepting the UART signals directly on the circuit board. Further, the voltage levels of the UART on the Watts Up? Pro are 3.3V LVTTL rendering

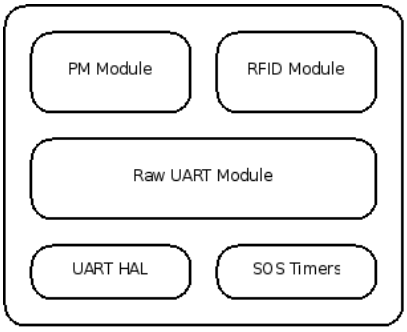

Fig. 3. Raw UART Driver Software Stack

it straightforward to interface to a MicaZ mote, which uses the same levels.

For power isolation, the Watts Up? Pro uses an Analog Devices ADuM1201 Dual Channel Digital Isolator just prior to feeding the UART signals into the FT232RL chip. Intercepting the UART signals meant connecting the mote between the ADuM1201 and the FT232RL, requiring to disconnect the tracks between the devices. In addition, it was found that the ADUM1201 was powered through the USB interface. Since the USB port is now disabled, the ADuM1201 has to be powered from the mote a drain of about 2mA. In summary, the hardware interface from the Watts Up? Pro to the MicaZ mote was performed by disconnecting four signals (Vdd, Rx, Tx, Gnd) between the ADuM1201 and the FT232RL and connecting those signals to the mote. A small custom board is used to connect the signals to the appropriate pins on the 51-pin mote connector.

C. Software Interfacing

In order to interface the Watts Up? Pro with SOS running on the MicaZ mote, a new UART driver was needed. The current UART driver in the SOS kernel implements a framing based protocol architecture assuming that SOS messages flow on the UART channel. The Watts Up? Pro on the other hand has its own framing and message structure as does the RFID reader described in Section IV. Figure 3 illustrates the software components developed to implement a custom UART driver. The driver is implemented as an SOS module, and reuses the UART HAL and timers provided within SOS. The RFID and PM (power measurement) modules implement device specific protocol communication such as device initialization and message parsing. The raw UART module provides a messaging interface to the PM and RFID. Each frame received on the UART link is forwarded as an SOS message to the higher layer module. However, since a raw channel does not provide any explicit framing mechanism, some method must be devised to find frame delimiters. Figure 4 shows how frames typically appear on the UART channel. It is noted that frames usually arrive with some minimum delay between frames called the inter frame delay or IFD. By exploiting this fact, the raw UART module can derive the size of frame.

The UART HAL implements an interrupt service routine that is called on reception of every byte on the UART channel. The raw UART module restarts a timer with the minimum IFD on every receive interrupt. The key idea is that if a new byte arrived before the IFD passed, the timer would get restarted without firing. Thus, when the timer does fire, it implies that a new byte has not been received for IFD time, indicating the end of the current frame. At this point, the raw UART module bundles up the data it has already received for that frame into an SOS message and posts it to one of the higher layer modules. This is the scheme for data reception, the case for data transmission is much simpler. The higher layer PM (or RFID) module posts a message to the raw UART module indicating the payload should be transmitted on the UART, handled by the transmit interrupt service routine. This IFD is typically device specific and depends on the baud rate as well and thus the raw UART module requires to provide an API to the PM and RFID module to access the UART channel correctly.

An important issue that comes up with very short IFD at high baud rates is that the timer API in SOS has a 500+ cycle latency, requiring $62.5\mu s$ to complete on an $8MHz$ ATMega128 AVR processor. The highest baud rate of $115.2kbps$ interrupts the processor once every $86\mu s$ leaving a small margin for the processor to complete other tasks. To circumvent this issue, a couple of methods have been proposed. The first mechanism uses the SOS counter API, which has a smaller (50 cycle) latency, to measure the time difference between byte interrupts. When the time difference between two bytes is found to be greater than the IFD, it indicates that a new frame has just begun and the latter byte is the first byte of a new frame. Note that this method posts a message only after the first byte of a new frame and thus the message delivery latency could be unbounded. In other words, this mechanism works well when the device connected to the UART is generating a constant stream of data frames. If, however, the data frames arrive at sporadic intervals a frame is not delivered until the next frame has just begun. The other issue with this form of system is that since the processor spends time posting an SOS message as a new frame is arriving, the message post latency has to be less than the interrupt rate. The second idea is to exploit some knowledge of the frames, such as header or trailer bytes or size (for fixed length frames). This last idea was implemented in the Spotlight project and has served its purpose well.

IV. RFID

A. RFID Reader Hardware

Since we found SkyeTek RFID reader in the backroom, we chose it. There were two choices, SkyeTek M1 and SkyeTek M1-mini. The former one is a bigger one and the latter one is smaller. One key difference between SkyeTek M1 and SkyeTek M1-mini is that the mini accepts variable Vcc(From 1.8V to 5V) which allows a MicaZ mote to supply power.

SkyeTek RFID reader supports various TAG reading modes as well as other functions such as writing and/or selecting TAGs. And it is compatible to ISO15693 passive RFID TAG families such as Tag-It HF-I(Texas Instrument), I-Code SLI(Phillips), my-d SRF55VxxP(Infineon), and LRI512(ST-Microelectronics), which means there's no concern with TAG compatibility issues.

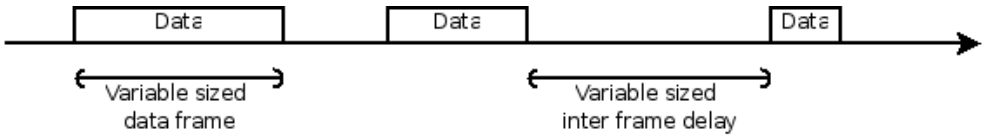

Fig. 4. Frames on the raw UART channel

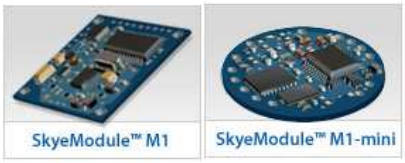

Fig. 5. SkyeTek RFID Reader Modules

Since SkyeTek M1-mini supports 3.3V TTL level UART interface, it can be directly connected to a MicaZ mote. Similar to the Watts Up Pro case, a small custom board is used to connect the (Vcc, Rx, Tx, GND) signals to the appropriate pins on the 51-pin mote connector.

B. Software Interfacing

The only concern for the RFID mote was the same as for the Watts Up Pro. It also requires RAW UART communication. To cope with it, we used the same RAW UART communication module for the RFID mote as for the Watts Up Pro mote.

Although the RFID reader supports various reading modes, for our system we only need the continuous reading mode where the RFID reader sends TAG ID readings when TAGs are within the range, otherwise no data signal is generated. The RFID mote consists of the four SOS modules called “Neighbor”, “Tree Routing”, “UART Module”, and “RFID Module”. The “Neighbor” and “Tree Routing” modules are used to enable multi-hop communication because our Spotlight system aims to be extensible to environment where the multi-hop communication is needed. As shown in III, UART module is used to strip headers of standard SOS message and make them RAW UART communication packets and vice versa. In other words, the UART module accepts and strips SOS_UART_send message and transmits RAW UART data strings to USARTx of the MicaZ mote. The RFID module is the main software interface for the RFID module. When the module starts, it initializes the SkyeTek M1-mini RFID reader and set it into the “Continuous Reading” mode. As soon as it receives any TAG ID readings, it forwards the data to the “Tree Routing module.”

V. Back-end System

A. Hardware Architecture

To meet the second project requirement, we decided to set up a back-end side with a Gumstix. The Gumstix is a finger size, full-function Linux computer which can expand its functionality by stacking various expansion boards. Firstly, we stacked NetCFstix to support Ethernet connectivity which enables the back-end to send sensor readings to the SensorBase on the fly. For interfacing the back-end and the infrastructure

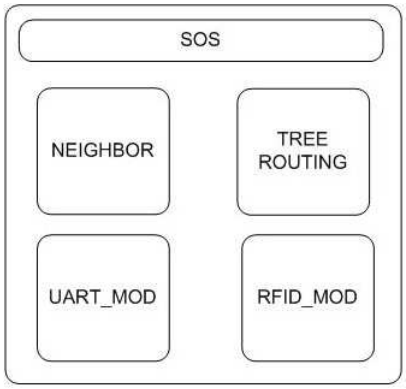

Fig. 6. RFID mote SOS S/W

sensor nodes, we used BreakoutGs which provides most of all pin-outs of the Gumstix. The base sensor node is connected to the Gumstix over the serial line through the BreakoutGS board.

By introducing a small sized back-end computer, we are envisioning that the Spotlight system will be embedded to various environment such as smart home, smart building, and smart office environments as illustrated in II. Advantages of the backbone system are obvious. It doesn’t require a big size server or a desktop computer, but provides relatively high computing power (Intel PXA255 400Mhz) which allows us to do more sophisticated data pre-processing than the mote class device can do. On one hand, it provides the SSH service and has the Internet connectivity, remote maintenance is also possible through the Internet connection.

B. Software Architecture

On the back-end, we run two applications, which are the SOS-server and the back-end application. The SOS-server basically opens a serial port to receive and send messages from/to a base sensor node. It also opens a network socket on the localhost so that other applications can communicate with the base node.

On the other hand, the back-end application, which consists of several python modules such as pySOSmodule, PowerMeasurementParser, RFIDParser, XMLmodule and spltSlogmodule, translates data from the infrastructure nodes and sends the results to the SensorBase. The pySOS provides several APIs for the SOSmessage communication. One can initialize pySOS module using “pysos.sossrv()” which makes a socket connection to the SOS-server through the localhost. Then “pysos.post()” can be used for posting SOSmessages and “pysos.listen()” allows us to receive SOSmessages. More detailed tutorial can be found at [2].

Since PowerMeasurement and RFID motes have their own protocol, we needed to write translator modules which are PMparser and RFIDparser, respectively. The reason why we use the back-end for translating those information is because they are strings and string processes are not well supported by SOS.

The protocol for the Watts Up Pro is an ASCII string "#d,-,18,<W>,<V>,<A>,...;<CR><LF>" which implies (Command, NumberofArgument, Watt, Voltage, Ampere, ...). PM mote forwards its power measurement reading with its network ID every one second. PMparser basically translates this information into the 5 tuples (TimeStamp, ApplianceID, Ampere, Watt, Voltage).

The protocol of the RFID reader is simpler than that of the PM. The packet from the RFID motes consists of their network address and TAG ID reading. Similarly, it translates these information into three tuples (TimeStamp, ApplianceID, TAG ID).

After both of translator modules make the data in appropriate form, XML module generates appropriate SensorBase XML scripts from the data which will eventually be sent to the SensorBase. Then it puts these XML packets into a queue where spltSlog module, which allows us to slog on the fly, grabs the data and sends it to the SensorBase. Once sets of all the tuples are stored at the predefined PowerMeasurement table and RFID table in the SensorBase, it can be used for off-line analysis as described in VI.

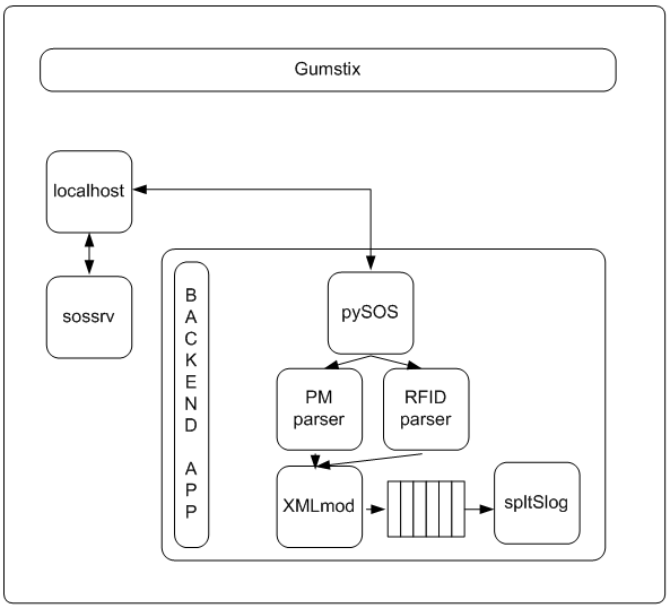

Fig. 7. Back-end Software Architecture

VI. R Algorithms

We constructed two usage profile in our application: one for the sharable appliances and one for the non-sharable.

A. Sharable Appliances

For the sharable appliances, we need to construct an occupancy profile of the users. We do this by collecting RFID tag information on when the users enter and leave the lab. We assume that the user diligently tag him/herself when entering and leaving the lab. Our data collection profile is robust enough to ensure no lost information. We start by assuming that no user is present in the lab. Then whenever a RFID tag is received, it is believed that the user has entered the lab and on the next occurrence of the tag, it is believed that the user has left the lab. Whenever anyone enters or leaves the lab, we calculate the additional power consumed by each user present in the lab during that duration and add it to his/her total power consumption.

B. Non-sharable Appliances

For non-sharable appliances, we simply detect usage by RFID readings. Whenever we get a RFID reading on a particular appliance, we determine that the appliance is used between a given interval before and after the reading. We will detect any appliance usage in this interval and allocate that consumption to the particular RFID reading.

Before we can determine an appliance use, we need to filter out the transients in the usage profile. From the data collected from the power-meter, we have power readings of the appliance at regular intervals. We smooth out these readings by applying a exponentially weighted moving average filter to the stream. The equation of the filter is the following:

$$y(n) = \alpha x(n) + (1 - \alpha)y(n - 1)$$

α has to be smaller than 1. Picking a big value for α emphasis the actual stream over the history and a smaller value emphasis the history over stream. Therefore, a small value will filter out more transients from the streams. We pick a modest value of 0.1 for removing the transients from our data stream. We will be using this filtered stream for all our data processing in this section.

We use a much smaller value of 0.001 for calculating the idle power of the device. This value will filter out most transients from the stream to the point that even when the device is in use, it will still have only a slight change in the filtered stream. Averaging the stream computes roughly the idle power of the device.

After we have computed the idle power and the filtered stream, we are ready to locate the usages of the devices based on the RFID readings. Whenever we detect a tag, we look for the usages on the filtered stream. We define the device to be in use whenever the filtered stream is outputting a value greater than some threshold determines from the idle power. We look for the closest such edge in our filtered stream to find the usage. We locate the low-to-high and the high-to-low edge of the usage, integrate it and add it to the user's consumption profile.

VII. Final deployment and Project requirement

A. Deployment Scenario

Figure 8 describes our deployment of Spotlight system in the NESL. There are two PM motes on the printer and the coffee machine where PM motes send power measurement packets to tree routing relay nodes once per a second. Also

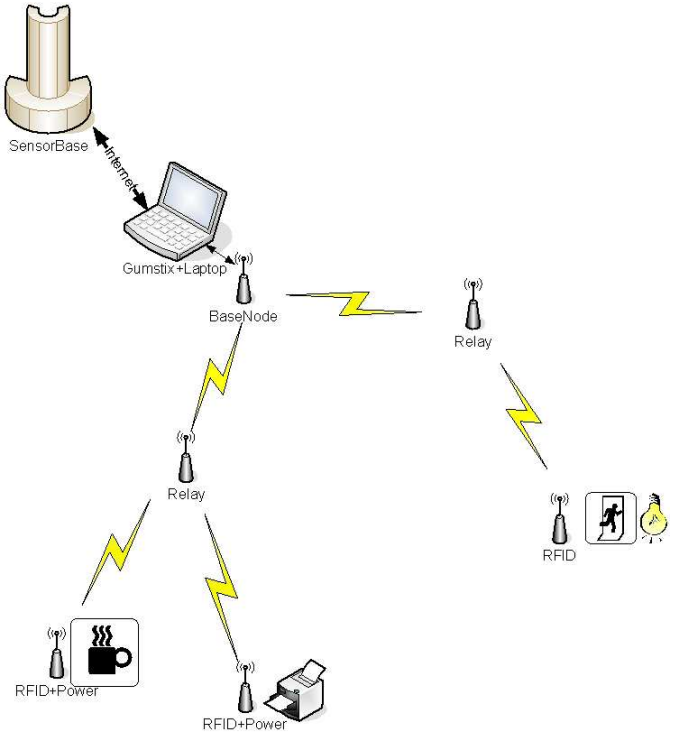

Fig. 8. Final deployment topology

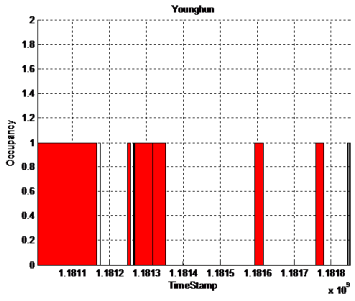

Fig. 9. Younghun's occupancy data

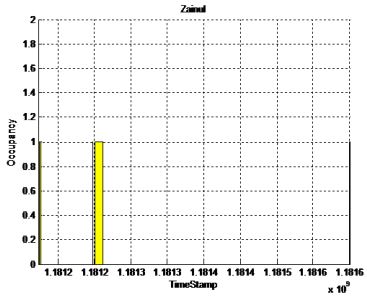

Fig. 10. Zainul's occupancy data

we placed RFID motes at the backdoor, the printer, and the coffee machine where they read TAG IDs as RFID TAGs are within the range. As soon as a RFID TAG is detected, they forward it to the tree routing relay so that it can reach the base node. The base node just sends data packets back to the back-end Gumstix on which SOS messages are translated and slogged to the SensorBase as described in V

In the sense of the project requirement, we used 8 MicaZ motes: 1 for the base node, 2 for the relay nodes, 3 for the RFID motes, and 2 for the PM motes as well as we used the Gumstix for the back-end. And the current deployment shows that the maximum hop count is two. As described in V, the back-end sends data to the sensor base on the fly as well. And of course, two sensing modalities RFID and Power Measurement (current, watts, voltage) are involved.

Some interesting observations are found during 3 weeks of intermittent deployment. First, the SensorBase returns "Fatal Error" when the number of rows of a table exceeds 100s of thousands of lines. This case will be reported to the administrator. In addition, we frequently encountered SOS "kernel panic" during the long-term deployment which is not supposed to happen.

VIII. Results

A. RFID with appliances

For three days deployment, we collected users' appliance usage profile. As described in VII, we placed two RFID motes at the coffee machine and the printer. A simple python retrieving module was used to get and interpret data.

Table I,II illustrate that the Spotlight captures users' appliance usage profile. But, in table II we can see in the first and the second rows that two continuous time stamps are stored, which makes off-line analysis tricky. However, one can be smart to cope with it by assuming that users can only use an appliance at once per 10 seconds or more. But still, there are a couple of factors which make the analysis difficult. For example, what happens if one user tags at the coffee machine and the other one actually uses it. And what happens if someone forgets tagging his or her TAG ID when they are using appliances. All those issues will be remaining works.

B. RFID for determining occupancy

In oppose to the appliance case, we also deployed a RFID mote at the door to determine who are in the lab. The simple scheme to determine occupancy is the following. Once one tags her/his ID, the system assumes she/he enters. Similarly, she/he tags the ID again, it assumes she/he is leaving. Although this scheme is simple and sensitive to malicious attacks such as "forget tagging" and "multiple tagging at a moment." We could find some interesting occupancy profile by analyzing those information.

Figures 9,10,11 and 12 give participants' occupancy profile, *x*-axis is time-stamp and 1 implies she/he is in and 0 implies that she/he is not in.

C. Idle/Usage Power measurement

Figure 13 shows the raw power measurement readings taken over a few days and illustrates how the readings vary. Using the algorithms described in Section VI, the data is filtered for processing using two values of α. A higher value of α provides a version without as many transients as shown in Figure 14, and is useful for determining usage transition edges. A much

UserName	TID	TimeStamp	Date/Time
Younghun	14E0040100004288B9	1181758272	2007.June.13/11:11.12
Younghun	14E0040100004288B9	1181846901	2007.June.14/11:48.21
Younghun	14E0040100004288B9	1181863507	2007.June.14/16:25.07
Akhi	14E004010000429DF7	1181858135	2007.June.14/14:55.35
Akhi	14E004010000429DF7	1181868116	2007.June.14/17:41.56

TABLE I
PRINTER USAGE PROFILE

UserName	TID	TimeStamp	Date/Time
Younghun	14E0040100004288B9	1181779440	2007.June.13/17:04.00
Younghun	14E0040100004288B9	1181779441	2007.June.13/17:04.01
Younghun	14E0040100004288B9	1181925106	2007.June.15/09:31.46

TABLE II
COFFEE MACHINE USAGE PROFILE

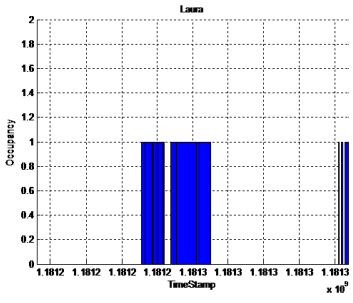

Fig. 11. Laura's occupancy data

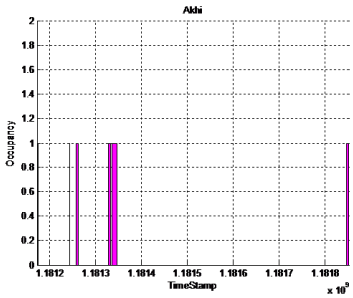

Fig. 12. Akhi's occupancy data

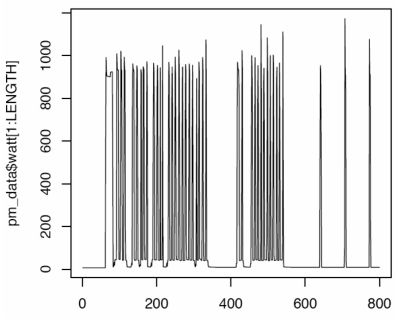

Fig. 13. Raw power measurement data

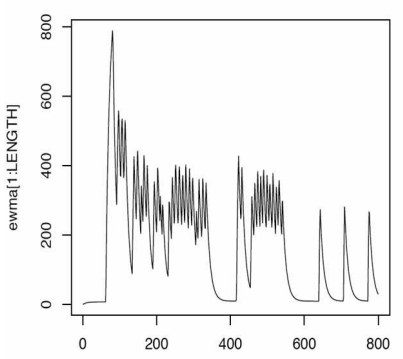

Fig. 14. Filtered power measurement data

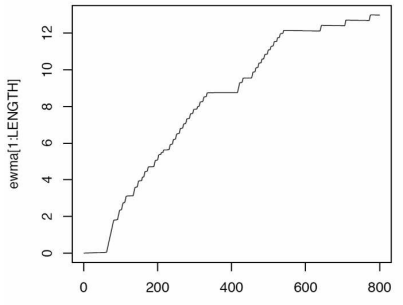

Fig. 15. Idle power estimation

lower value of α is used to estimate the idle power as shown in Figure 15.

IX. RELATED WORK

A. Demand Response Enabling Technology Development - UC Berkeley

Demand Response [1] refers to the mechanism by which utility providers can reduce the demand for electricity during shortages by setting variable rates based on the load with a mechanism for customers to respond to those rates. This means that if a customer can defer usage of an appliance to a time at which the rate is lower, it would result in a lower cost to the customer as well as the utility provider. New DR-enabled

thermostats are proposed that have metering, communications and automatic response mechanisms. The new thermostats require the homeowner to program the system once to express her preferences for cost versus comfort. In addition, appliances include price indicators as familiar red-orange-green LEDs to indicate the current price scenario. The Berkeley system is deemed fairly close to our measurement of electrical energy, but is coarser grained. Their system is ?active? in its use of a feedback mechanism, whereas our system is designed to be a ?passive? profiling tool for an individual or a household. The Berkeley system requires communication with the utility provider and the provider has access to the energy consumption profile at all times. This may lead to privacy concerns.

B. Samsung Electronic Company's Smart Building

Energy measurement and control system based upon occupancy is not just being explored in research but is also being adopted in real life. The new building at Samsung Electronic Company Digital Media R&D center, opened in Sep. 2005, is equipped with an RFID based systems for air conditioning and light level control. Each employee and visitor is required to carry an RFID tag. The RFID receivers and back-end systems identify the occupancy level in a room based on which the building controls its air conditioning and lighting systems. This system also helps the authorities track visitors? movements and avoid information theft.

In this work, we see that they measure and control the energy consumption of sharable electronic appliances. The ideas that we would like to explore in this project are similar but extend to all appliances that a person may come across.

X. Conclusion

In conclusion, we have shown that energy consumption monitoring at a finer granularity of individuals is possible. We have carried out such profiling by dividing the commonly found appliances into two different classes and described our algorithms for profiling such devices. Most commonly found appliances fit in this classes of devices while some don't. An interesting area of future work will be identifying in further detail other appliances that do not fall in either category and developing profiling algorithms for them and also addressing some of the limitations of our application discussed in section II-B.

References

[1] E. Arens et al. Demand response enabling technology development. *Report to CEC Public Interest Energy Research (PIER) Program, Center for the Built Environment, University of California, Berkeley*, 2006.

[2] PySOS. http://enaweb.eng.yale.edu/drupal/pysos, 2007.