Optimizing Trade-offs among Performance, Accuracy and Energy Consumption in Wireless Sensor Nodes

by

Inkeun Cho

University of Maryland, College Park
Electrical and Computer Engineering

The Final Research Paper for Austrian Marshall Plan Scholarship, 2012 Springs

1. INTRODUCTION

A *wireless sensor network*(*WSN*) is distributed, wireless communication network system, where the nodes of the network (*sensor nodes*) contain groups of sensor devices. In addition to their associated sensors, sensor nodes typically consists of microprocessors, wireless communication interfaces, and energy sources. The small size and low cost of many modern sensor devices allow designers to deploy WSNs for a wide variety of applications.

Major research issues for WSNs include efficient hardware implementation and software stacks for energy-optimized wireless communication. Various low power processing units for sensor nodes have been developed with objectives of application-specific energy optimization. Development of new sensor technologies is also an important research area that gains motivation from the rapid evolution of WSN applications.

Research on design methods for WSN applications has received relatively little attention compared to the aforementioned WSN-related research areas. Design of WSN applications requires careful attention to operating system, data acquisition, and communication protocol aspects and their interactions. The complexity of these interactions and the need to manage this complexity under strict energy constraints contribute significantly to the difficulty of WSN application development. In this

paper, we address this difficulty by developing new design methods for WSN systems based on the formal application modeling framework of signal processing dataflow graphs. Our methods help to ease the task of implementing WSN applications, and to optimize their energy efficiency so that they can operate for longer periods before they need to be serviced or replaced.

More specifically, we develop a number of contributions to energy analysis and optimization of sensor nodes in WSN application. First, we develop a new energy analysis method for this class of sensor nodes at the application level and network level. Then we apply this energy analysis method to develop a new energy management scheme that is targeted to maximizing end node lifetime in building energy monitoring system(*BEMs*). To enhance the efficiency and reliability with which this management scheme can be implemented, we develop a new application-level design approaches of two representative application in each group that uses dataflow to model the application-level interfacing behavior between the processor and sensors on an individual sensor node. At the network level, we analyze the energy consumption in the 802.15.4 Zigbee network medium access control (MAC) layer based on the state of the sensor node radios. We validate our proposed methods for energy analysis and optimization through experiments on a fully functional building energy monitoring system in which the sensor nodes are equipped with Texas Instruments CC2530 Zigbee network-enabled micro controllers [24].

2. RELATED WORKS

2.1 Wireless Sensor Network Applications

WSN systems are widely used in real world applications. Akyildiz et al. present a survey on various application areas of WSNs, and categorize WSN applications as military, environmental, health, home, and other commercial areas depending on the kinds of sensing, processing and communication functionality provided [1]. In military applications, WSN systems can be used for applications such as equipment monitoring or detection of chemical attacks. WSNs can be applied for flood or mountain fire detection in environmental applications, and used to track physical data for patients in healthcare applications.

WSNs can be categorized as monitoring systems and signal processing systems. For example, flood detection, physical data tracking, and building energy monitoring involve periodic reporting of sensed data, which is characteristic of monitoring systems. On the other hand, intruder detection and home automation typically involve intensive image, acoustic, or speech signal processing, and are thus representative of signal processing oriented WSNs. In monitoring systems, the behavior of a sensor node exhibits periodic, alternating mode changes from active to sleep states depending on the reporting schedule. In signal processing oriented WSNs,

the system continually acquires and processes sensed data for event detection, and classification.

In this paper, we develop new design methods for monitoring and signal processing oriented WSN systems. Our methods apply the recently introduced lightweight dataflow programming model [21] [16], which provides a flexible and retargetable approach for developing monitoring and signal processing applications based on formal dataflow graph semantics. Using lightweight dataflow programming and dataflow graph analysis, we have developed systematic methods for measuring and optimizing energy consumption in sensor nodes. We have demonstrated and refined these methods using a representative application in the monitoring domain, and in our future work, we propose to address also the signal processing domain. Specific applications that we are focusing on include building energy monitoring (for the monitoring domain), and distributed speech recognition (for the signal processing domain).

2.2 Wireless Sensor Network Building Energy Monitoring System

2.2.1 Wireless Sensor Network Existing Researches on Network Lifetime

Zhaohua and Mingjun [22] present a survey on network lifetime research for wireless sensor networks. Zhang, Jia, and Yuan [6] and Padmavathy [8] have proposed network protocols and assocaited algorithms for improving energy efficiency. Wang and Kulkarni [7] consider scenarios in which sensor nodes are over-deployed in certain regions, and explore trade-offs between network lifetime and coverage

through application of partial coverage transmission. Madan et al. [12] investigate cross layer design for the physical, MAC, and routing layers to maximize lifetime in wireless sensor networks.

These are largely general-purpose approaches, which are formulated without taking into account detailed characteristics of the targeted wireless sensor network or monitoring application.

Various bodies of research have targeted WSNBEMS technology (e.g., see[2] [5] [20]). Jang, Healy and Skibniewski [5] develop a building monitoring system with associated hardware, a web-based user interface, and monitoring software; evaluate this system on a wireless sensor network; and demonstrate that such a network can be employed to provide low cost monitoring. Chintalapudi et al. [20] show how wireless sensor networks can be used in structural health monitoring, and describe actual monitoring systems that are deployed to monitor real structures. These efforts have emphasized the development and demonstration of practical wireless sensor network based monitoring systems, and have not placed significant emphasis on optimizing energy consumption or maximizing lifetime for the network nodes.

Our research differs from these approaches in that we develop methods to exploit specific characteristics of our targeted class of WSNBEMSs, and streamline the design to maximize network lifetime based on these characteristics. Also, whereas an important body of related work focuses on network level considerations, the approach developed focuses on trade-off analysis and optimizations that take into account application level quality of service (reporting accuracy).

We therefore provide an additional layer of application-driven optimization

that is geared towards the important domain of WSNBEMS and goes beyond what can be provided by more generic methods. As part of this application-driven approach, we model the target application in terms of core functional dataflow [15], which allows us to formally capture the signal processing functionality in a form that can be efficiently analyzed and mapped into hardware and software. Additionally, we apply new techniques for MAC layer energy analysis and we apply these techniques to optimize transceiver energy consumption in sensor nodes.

3. ENERGY ANALYSIS FOR MAC LAYER

3.1 IEEE 802.15.4 MAC layer overview

The 802.15.4 Zigbee network MAC layer protocol is a hierarchical protocol in which at any given time, a single *coordinator* node controls a set of one or more *end nodes* in the network. This protocol has two operation modes, which are referred to as the *beacon-enabled mode* and *non-beacon mode*. The non-beacon mode works with carrier sense multiple access with collision avoidance (CSMACA) communication.

In the non-beacon mode, data communication from end nodes is allowed at arbitrary times. To support such flexibility, the coordinator continually monitors the communication channel for messages from the associated end nodes. Such monitoring is expensive in terms of energy consumption, and furthermore the flexibility it provides is not needed in our targeted WSNBEMS applications, where data communication from end nodes can be restricted to occur at pre-specified, periodic times. Thus, we are able to perform all of the intra-node communication in our wireless sensor network architecture using only the beacon-enabled mode, which is more energy efficient.

In the beacon-enabled mode, communication is governed by a *superframe* structure, which is illustrated in Figure 3.1. The superframe structure is com-

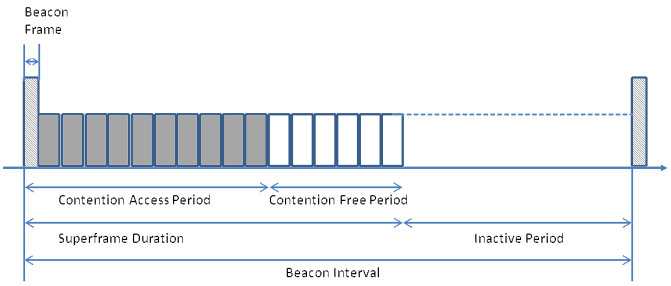

Fig. 3.1: 802.15.4 Zigbee network beacon interval. The superframe structure is illustrated as the component of the beacon interval that lies within the *superframe duration*.

posed of four parts, which are called the beacon frame, contention access period (CAP), contention free period (CFP) with guaranteed time slot(GTS), and inactive period. The superframe structure (indicated by the interval in Figure 3.1 labeled *superframe duration*) is composed of 16 time slots, where each slot corresponds to communication from at most one end node to the coordinator. The beacon frame is used to synchronize all of the nodes in the network. In the CAP, the end nodes that are ready to transmit data (ready nodes) check their back-off timers (the values of these timers are randomly assigned), and wait until their respective timers expire. After appropriate back-off, a ready node sends data if the channel is clear; otherwise, its backoff timer is set to another random value, and its backoff process is repeated. Through the CFP, a coordinator node can selectively assign guaranteed time slots to specific nodes, and nodes that obtain such permissions can send data during the CFP without channel sensing.

3.2 Energy modeling of the beacon-enabled mode

End node lifetime can be estimated by dividing the battery capacity (in Joules) with the average power consumption of an end node (in Watts). In the WSN system, sensor node micro-controller have variable power states and the dynamic power scheduling in the node is based on the transition of power states. The power states in sensor node platform can be composed of *ActiveState* and *SleepState*. The *ActiveState* can be divided into three sub states, which are reception state, transmission state and processing state. The required power can be varied with states. In our target application, the average power consumption of an end node can be divided into application layer and MAC layer power consumption. The application layer power consumption includes the power expended for acquiring sensed values from the attached sensors, such as temperature, light, and humidity sensors. The MAC layer power consumption includes the power expended for handling MAC events, the transceiver power consumption for transmission of sensed data, and sensor node sleep mode power consumption during the inactive periods of network beacon intervals. For simplicity, we take into account only the MAC layer data transmission, and ignore the effects of communication that are due to other factors. Although this is a significant simplification, we show in our experiments (Section 6) that it does not have a major affect on the energy consumption *trend* in our experiments; that is, we can compare alternative design configurations with reasonable accuracy under this assumption.

We start by analysis of 802.15.4 MAC energy with the different energy con-

sumption levels during beacon interval. The transmission in 802.15.4 MAC can be divided into two groups which are contention access and contention free based transmission. In the contention access case, the data is transmitted after carried sensing and the energy consumption during this beacon interval can be varied with the number of fail in carrier sensing. In the papers [13] and [14], they assumed the maximum number of trying for carrier sensing as 5, and showed that this assumption is relevant. We followed this assumption in this paper, and the energy consumption levels in CAP based transmission can be five states. In detail, the $CSMA1$ means that the transmission is succeeded right after the first carrier sensing and $CSMA5$ is the failure of transmission after 5 times carrier sensing contention in the figure 3.2. On the other hand, the CFP based transmission has no contention in the transmission because of transmission in allocated slots. The request for the GTS should be prior to the GTS transmission and the transmission can be done after getting the beacon message which is included the information of GTS slot. GTS request message is also based on the carrier sense, there are five energy states.

In the paper [13] and [14], they assumed the contention probability is followed the geometric distribution. In this paper, we brought the assumption in these papers. We define the P_{NC} as the probability for success of carrier sensing without contention in random back-off time

If we assume X as the number of trial for carrier sensing in a beacon interval, the probability of random variable X is

$$P(X = k) = (1 - P_{NC})^{k-1} P_{NC} \tag{3.1}$$

The P_C which is the probability of contention in carrier sensing and the P_{NC} is

$$P_{NC} = 1 - P_C \tag{3.2}$$

The P_C is closely related to the number of nodes in the network and the periodic transmission interval of each end node. In our application, the data size of one transmission interval is 32bytes and transmission required time for the data is 1ms under 256Kbps of 802.15.4 Zigbee MAC data rate. In this case, a time slot in beacon interval is 1ms, so the contention only happen the node's random backoff time is same with other's backoff time. The probability of random backoff is uniformly distributed and the probability q is

$$q = \frac{1}{2^{BE}} \tag{3.3}$$

The probability of contention for the node is sum of probabilities that one or more nodes have same backoff period in case that k nodes participate in contention period, so it is

$${P_C}^b(k) = q \sum_{m=1}^{k-1} \binom{k-1}{m} q^m (1-q)^{k-1-m} = q(1-(1-q)^{k-1}) \tag{3.4}$$

In our application, the sensor node have periodical report of sensed data and the sensor node sleeps during subsequent beacon intervals. There are two kinds of beacon interval, which are transmission beacon interval and sleep beacon interval. The probability that transmission happen in this beacon interval r is related to data

transmission interval T_{int} and it is

$$\begin{aligned} r &= \frac{BI}{T_{int}} \\ T_s &= aBaseSuperframeDuration \\ SD &= T_s 2^{SO} \\ BI &= T_s 2^{BO} \end{aligned} \tag{3.5}$$

If the number of nodes n is in the network, the probability of k nodes participate to transmit data in this beacon interval is

$$P_C{}^T(n,k) = r\binom{n-1}{k-1} r^{k-1}(1-r)^{n-k} \tag{3.6}$$

If we say that n and k are the number of nodes in the network and the number of nodes that try to transmit data in this beacon interval, respectively, the $P_C(n,k)$ is defined as

$$P_C(n,k) = P[contention,(n,k)] = P[contention|(n,k)]P[(n,k)] = P_C{}^b(k)P_C{}^T(n,k) \tag{3.7}$$

If we want to make a Markov Chain for energy state of beacon interval, we should assume that the state transition for recurring non-transmission beacon interval is followed the geometric distribution. To adjust the transmission interval as T_{int}, the probability of P_{TR} is decided depending on T_{int}. The figure 3.2 shows the Markov Chain for the energy state of beacon interval.

The energy model for a beacon interval can be the mean value of energy in Markov chain and it is

$$E_b = BI \sum_{i_{states}} \pi_i E_b^i \tag{3.8}$$

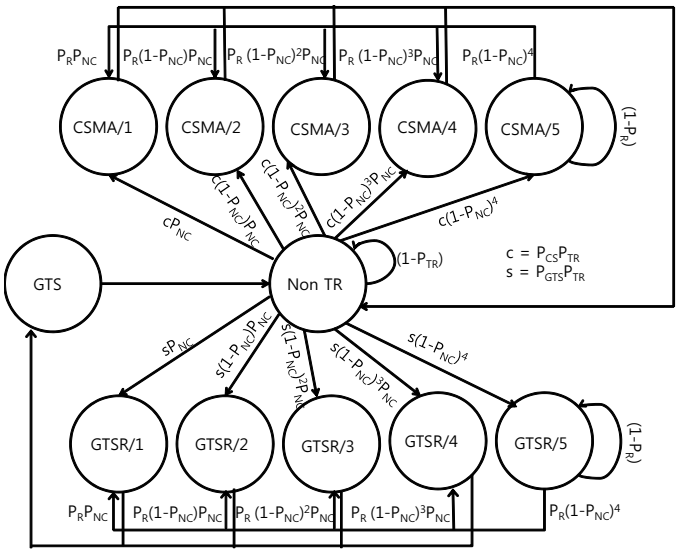

Fig. 3.2: Markov Chain for energy state for beacon interval

The π_i is the steady state probability of Markov Chain and it can be calculated from the transition matrix of Markov Chain. In E_b^i, energy consumption for i state, all state include the beacon message receiving, so the the energy for receiving beacon message E_{rb} is

$$E_{rb} = P_R T_s \tag{3.9}$$

The energy for k times transmission of carrier sensing, $E_{cs}(k)$, is

$$E_{cs}(k) = \begin{cases} kP_T T_s + P_R T_s & 0 \leq k \leq 4 \\ kP_T T_s & k = 5 \end{cases} \tag{3.10}$$

The energy for transmission of m bits data, E_t, is regardless of transmission methods, and we assumed that it takes a T_s. It is

$$E_t = P_T T_s \tag{3.11}$$

The energy for non transmission beacon interval, E_b^{NT}, is composed of receiving transmission beacon and the sleep energy for remaining beacon interval during sleep

duration(SD), and it is

$$E_b^{NT} = E_{rb} + (SD - T_s)P_{SL} \tag{3.12}$$

The energy for CSMA based transmission beacon interval, $E_b^{CSMA}(k)$, is

$$E_b^{CSMA}(k) = \begin{cases} E_{rb} + E_{cs}(k) + E_t + (SD - (k+2)T_s)P_{SL} & 0 \leq k \leq 4 \\ E_{rb} + E_{cs}(k) + E_t + (SD - (k+1)T_s)P_{SL} & k = 5 \end{cases} \tag{3.13}$$

The energy for GTS request beacon interval, $E_b^{GTSR}(k)$, is

$$E_b^{GTSR}(k) = \begin{cases} E_{rb} + E_{cs}(k) + (SD - (k+2)T_s)P_{SL} & 0 \leq k \leq 4 \\ E_{rb} + E_{cs}(k) + (SD - (k+1)T_s)P_{SL} & k = 5 \end{cases} \tag{3.14}$$

The energy for GTS transmission beacon interval, E_b^{GTST}, is

$$E_b^{GTST} = E_{rb} + E_t + (SD - 2T_s)P_{SL} \tag{3.15}$$

4. BUILDING ENERGY MONITORING APPLICATION

ENERGY MONITORING

In this section, we describe our approach to energy analysis for BEMS applications. We start by describing the *lightweight dataflow* programming approach, which we use for design and implementation of embedded software in terms of formal dataflow models of computation.

4.1 Lightweight dataflow

Lightweight dataflow (LWDF) is a programming methodology for implementation of signal processing systems based on the core functional dataflow (CFDF) model of computation [21]. In CFDF, as in other forms of dataflow, an application is represented as a directed graph in which vertices, called *actors*, represent computational modules, and edges represent first in first out buffers between communicating pairs of modules [15]. Dataflow actors execute as sequences of discrete computational units, which are called actor *firings*.

Dataflow graphs are useful in the design and implementation of signal processing systems, including many kinds of wireless sensor network applications, because they expose valuable high level signal processing flowgraph structure that can be

exploited for application analysis and implementation optimization (e.g., see [18]).

By avoiding dependence on specialized tools and facilitating retargetability to arbitrary host languages (languages for implementing individual actors), LWDF provides a minimally intrusive methodology for implementing applications based on the CFDF model, and can be incorprated efficiently into existing design processes [21]. In our approach to design and implementation of WSNBEMS systems, we employ LWDF-C, which is the integration of the LWDF programming methodology with the C programming language. C is commonly used in developing embedded software for sensor nodes, and LWDF-C provides a framework for implementing such software through rigorous dataflow principles.

In LWDF, each actor has an operational context (OC), which encapsulates all parameters, local variables, and state variables of the actor, as well as references to the ports of the input and output FIFOs. Actor design in LWDF involves four interface functions called the *construct*, *execute* (also called *invoke*), *terminate*, and *enable* functions. The construct function performs one-time memory allocation and initialization associated with setting up the actor, and is typically called once at the beginning of the application. Similarly, the terminate function performs memory deallocation and other actor-level wrapup tasks, and is called when the actor is no longer needed, typically after the application is shut down or otherwise terminated.

Each call to the execute function performs a single firing of the associated actor provided that the input ports of the actor have sufficient input data. If the execute function is called when there is not sufficient data, the results are in general unpredictable. For each call to the execute function, the designer or enclosing

tool can ensure sufficient data through appropriate static or quasi-static scheduling techniques or by first checking the status of the input buffers using the LWDF enable funciton. The enable function is a function that returns a Boolean value. This return value is `true` if and only if the actor has sufficient data on in its input edges and sufficient empty space on its output edges to carry out the next firing of the actor, including all of the required data consumption and production.

In CFDF and LWDF, the firing of actors and the associated tests that are performed by the enable function are formulated in terms of distinct *modes* of the actor. In general, an actor has one or more modes. A mode can be viewed as a "firing template" (functionality for a well defined subset of firings) in which the number of data values produced and consumed from the actor ports is constant. Thus, each mode has fixed data transfer (production and consumption behavior), but data transfer behavior can vary across different modes. Such decomposition of actor firing behavior in terms of fixed data transfer units is useful since data transfer characteristics generally have a strong influence on techniques for dataflow graph scheduling and other forms of dataflow graph analysis (e.g., see [18]).

4.2 WSNBEMS system model

In the model of WSNBEMSs that we apply, end nodes, which are equipped with heterogeneous sets of measurement sensors, are deployed in fixed positions, and these end nodes periodically send sensor measurements to a central network node, which we refer to as the *coordinator node*. The coordinator node collects data

from all of the end nodes, and processes the data to determine properties of overall building energy consumption.

In the WSNBEMS testbed that we are experimenting with, the wireless communication range for end nodes is approximately 30 meters, and router nodes are used for multi-hop networking and clustering.

Each end node in our testbed is composed of a microcontroller for data processing and peripheral control, and a wireless transceiver subsystem for communication with other nodes. The microcontroller on an end node communicates with its associated sensors and acquires environmental measurement data using peripheral communication protocols.

We encapsulate the functionality for sensor interfaces in LWDF actor implementations so that sensor interfaces are integrated into the overall design using the same general module design approach as all other functional components. Thus, regardless of whether the sensors support I2C, SPI or some other protocol, the sensor interfaces communicate with other actors in the same way, which makes it easy to integrate sensors into the end nodes using a standard, protocol-independent approach. Moreover, the standardized interfacing and encapsulation as dataflow components facilitates more comprehensive model-based analysis, including energy estimation, for the overall application. This feature, for example, allows designers to derive useful estimates of energy consumption for alternative designs before deploying and testing the design in real environments.

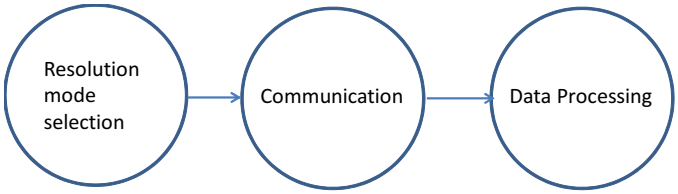

Fig. 4.1: LWDF application model for the targeted WSNBEMS application.

4.2.1 Application energy modeling

Commonly, environmental sensors support multiple sensing modes having different resolutions. Such sensing mode alternatives provide for trade-offs among the resolution of the data arriving from the sensor interfaces, number of bits required to communicate the sensed data across the interface and network, and time required to obtain the sensor readings. Resolution modes for sensors are generally configured by sending appropriate instructions to the sensors.

The mode based decomposition of LWDF firings provides a natural method for specifying functionality associated with different resolution modes within LWDF actors. In our WSNBEMS end node design, we provide for dynamic selection of sensor resolution modes through a *resolution mode selection* actor. Such an actor can, for example, be connected to a bank of switches or process configuration data sent by the coordinator node to dynamically adapt the resolution mode based on user input or coordinator node decisions. On each firing, the resolution mode selection actor reads the resolution settings from the appropriate configuration interface, packages the settings as a sequence of dataflow tokens (data packets), and passes these tokens to one or more subsequent actors for controlling how sensor data is read and transmitted.

```
boolean resolution_mode_selection_enable() {
boolean result = FALSE;
switch(context->mode) {
        case HIGH_resolution:
       result = TRUE;
           break;
        case LOW_resolution:
       result = TRUE;
           break;
        default:
           wsnbems_exception(context, INVALID_ACTOR_MODE);
           break;
}
    return result;
}

void resolution_mode_selection_invoke() {
switch(context-> mode) {
        case HIGH_resolution:
            value=generate_inst(HIGH_resol);
            lwdfc_fifo_write(&value);
            context->mode=decide_next_mode();
            break;
```

```
        case LOW_resolution :
            value=generate_inst(LOW_resol);
            lwdfc_fifo_write(&value);
            context->mode=decide_next_mode();
            break;
        default:
            wsnbems_exception(context, INVALID_ACTOR_MODE);
            break;
    }
}
```

Figure 4.1 illustrates our LWDF-based application model for our experimental WSNBEMS application. As an example of actor programming in our design, selected code is sketched below for the resolution mode selection actor.

We analyze the energy consumption of the end node functionality in terms of the LWDF application model of Figure 4.1.

The overall application energy is estimated as the sum of energy consumption estimates that are derived for the individual actors. In the estimation process, each component of energy consumption (E_{en}, E_{fifow}, etc.) is derived by instrumenting the application to determine the amount of time spent in the associated processing state, and multiplying by an estimate of the power consumption in each state. The instrumented values are obtained through actual application execution on the targeted hardware (not by simulation).

The overall energy estimation approach is summarized by the following equations.

$$E_{\text{app}} = E_{\text{res}} + E_{\text{com}} + E_{\text{dp}}$$

$$E_{\text{res}} = E_{\text{en}}(res) + E_{\text{fifow}} + E_{\text{mdch}}(res) + E_{\text{inst}}$$

$$E_{\text{com}} = E_{\text{en}}(com) + E_{\text{fifor}} + E_{\text{intfmode}} + E_{\text{fifow}}$$

$$E_{\text{dp}} = E_{\text{en}}(dp) + E_{\text{fifor}} + E_{\text{dps}}$$

Table 4.1 provides a legend of the symbols that are used in these energy modeling equations.

E_{app}	Application Energy
E_{res}	Resolution Selection Actor Energy
E_{com}	Communication Actor Energy
E_{dp}	Data Processing Actor Energy
E_{en}	Actor Enabling Energy
E_{fifow}	FIFO Write Energy
E_{fifor}	FIFO Read Energy
E_{mdch}	Energy for Mode Change
E_{inst}	Energy for Instruction Generation
$E_{intfmode}$	Sensor Interface Communication Energy
E_{dps}	Data Processing Energy

Tab. 4.1: Terminology for Energy Analysis

5. ENERGY EFFICIENT MANAGEMENT SCHEME FOR WSNBEMS

In the previous sections, we have developed models for analyzing the energy consumption of our targeted WSNBEMS, both in terms of application- and MAC-related energy consumption. To derive energy estimates from these models, we apply appropriate hardware platform parameters. For example, Table 5.1 shows relevant platform parameter values for the Texas Instruments CC2530 microcontroller, which supports 802.15.4 Zigbee networks, and is the platform used in our experiments. However, we note that through its basis in dataflow-based application representations, our energy analysis methodology is easily retargeted to other platforms, and is not specific to the hardware used in our implementation.

As an example of how this kind of energy modeling can be applied, Figure 5.1 shows how end node lifetime varies as we increase the time interval for transmission, and apply the parameters in Table 5.1 and a battery with an energy capacity of 1250mAh. As we would expect, the lifetime has a tendency to increase with increasing intervals for transmission, and our detailed energy analysis approach helps to quantify this trend so that we can identify a suitable trade-off based on application requirements.

Parameter	Value
$\text{Current}_{\text{TX}}$	39.6 mA
$\text{Current}_{\text{Active}}$	20.5 mA
$\text{Current}_{\text{Sleep}}$	1 uA
t_{timeslot}	320 us
V_{op}	3.3 V

Tab. 5.1: Hardware platform parameters for the Texas Instruments CC2530.

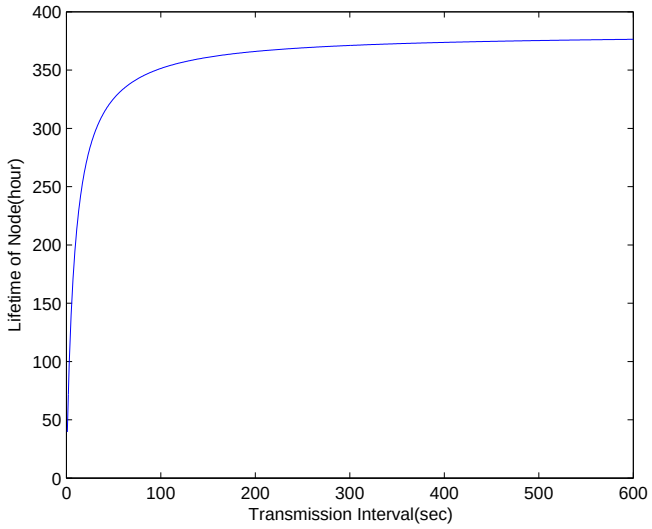

Fig. 5.1: Node lifetime versus transmission interval.

5.1 End Node Based Energy Saving Scheme

In this section, we present the *end node based network lifetime* (*ENBESS*) scheme, which is an energy saving scheme that helps to extend end node lifetime by

adjusting the transmission interval based on relevant operating conditions. This approach is motivated by the trade-off between transmission interval and node lifetime, which was discussed above.

ENBESS is motivated by the observation that environmental data of interest in our application, such as temperature and light, is relatively stable in building environments. ENBESS exploits this stability by maintaining a moving average of sensed data for each monitoring modality (temperature, humidity, light, etc.) of interest, and transmitting at increasingly long intervals when the current sensed values are sufficiently close to their associated moving average values.

When each modality is sensed, the moving average is updated. Then a weighted sum is used to determine the deviation of the current set of sensed values from the associated moving averages. This deviation is compared to a pre-specified threshold, and if the difference is within the threshold then the data transmission time is increased; on the other hand, if the deviation is moderately outside the threshold, then the transmission time is decreased. If the deviation from the threshold is outside of a pre-specified *normal range*, then a much shorter *exception interval* is used to notify the controller node. This notification alerts the controller node to an exceptional change in sensed value status, which could indicate, for example, a device failure or dangerous event in the environment.

The ENBESS method is sketched by the pseudocode shown in Algorithm 1. By strategically adjusting between shorter and longer transmission intervals, the method provides for both accurate data reporting to the coordinator node, as well as energy efficient operation on the end nodes.

In Algorithm 1, a provides a weight for updating the moving average value, which is described above. F_{int} is a weight that is applied to the difference between the newly updated moving average and stored moving average, and W_{sen} is a coefficient applied to each sensed value measurement. The value of W_{sen} should be determined based on the units in which sensor value measurements are provided and the associated magnitude ranges. For example, in our measurement scenario, temperature sensor values typically range from 20 to 35 degree Celcius, while light sensor values range from 0 to 800 degree Lux. In our experiments, we used coefficient values of 0.2 and 0.01, for temperature and light readings, respectively, to help normalize the values into similar ranges. t_{Base} provides a time offset that we used to update successive transmission intervals. In our experiments, we used 12 seconds as the value for t_{Base}.

Algorithm 1 End Node Based Energy Saving Scheme.

$Fint = 0$

for $i = 1$ to $Nsen$ **do**

$Mavg[i] = a * Msen[i] + (1 - a) * Mpavg[i]$

$Dsen[i] = Mavg[i] - Mpavg[i]$

$Mpavg[i] = Mavg[i]$

end for

for $i = 1$ to $Nsen$ **do**

$Fint = Fint + Dsen[i] * Wsen[i]$

end for

if $Fint > Lbase$ **then**

$NexInterval = PreInterval + tBase * Fint$

else if $Fint \leq Lbase$ **then**

$NexInterval = PreInterval - tBase * Fint$

end if

for $i = 1$ to $Nsen$ **do**

if $Msen[i]$ is out of normal range **then**

$NexInterval = ExceptionInterval$

end if

end for

$PreInterval = NexInterval$

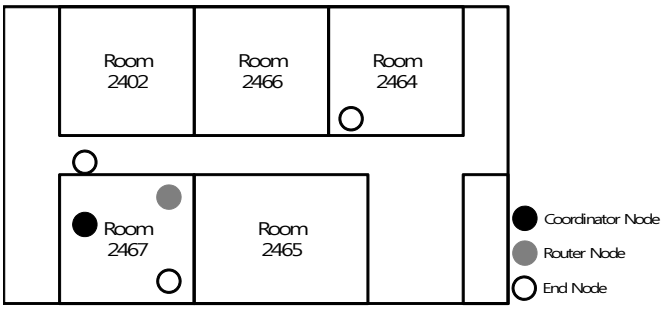

Fig. 6.1: Network for WSNBEMS testbed.

6. SIMULATION AND EXPERIMENT

6.1 WSNBEMS Testbed

To experiment with our methods for design and energy analysis, we have developed a fully functional, small scale WSNBEMS testbed. The network nodes in this testbed employ Texas Instruments CC2530 802.15.4 Zigbee network enabled (Z-Stack 1.4.0) [25] microcontrollers.

Figure 6.1 shows an overview of the network in our testbed. The network includes one coordinator node, one router node, and three end nodes. Each end node is equipped with two environmental sensors — a temperature sensor and a light sensor. Both of these sensors communicate with the associated microcontroller using the I2C peripheral communication protocol [26].

The network is deployed in the A. V. Williams Building on the University

of Maryland, College Park campus. End nodes periodically send sensed values for temperature and light to the coordinator node, and the router node relays information to the coordinator node in case end nodes cannot reach the coordinator node directly.

6.2 Node Lifetime

In our testbed, the Texas Instruments Z-stack is used to implement the 802.15.4 Zigbee protocol, and the Operating System Abstraction Layer (OSAL) is used for node scheduling. The Z-stack is composed of 7 main layers — the MAC layer, network layer, hardware abstraction layer, application support sub-layer, monitoring task, interface for Zigbee application layer, and application layer. In our energy analysis, we consider in detail the MAC layer and application layer, so our energy analysis needs some model for differentiating results measured from our testbed with results obtained from our analysis. To help differentiate between testbed and analysis results, we define the term E_{layers} to represent the combined energy consumption of the network layer, hardware abstraction layer, application support sub-layer, and interface for Zigbee application layer.

To estimate the energy consumption for these layers (i.e., all layers apart from the application and MAC layers), we measure the required numbers of clock cycles for processing the layers with increasing data transmission intervals. From these clock cycle counts, we can estimate the associated energy consumption values. The clock cycle counts are measured 15 times and the average across these 15 trials

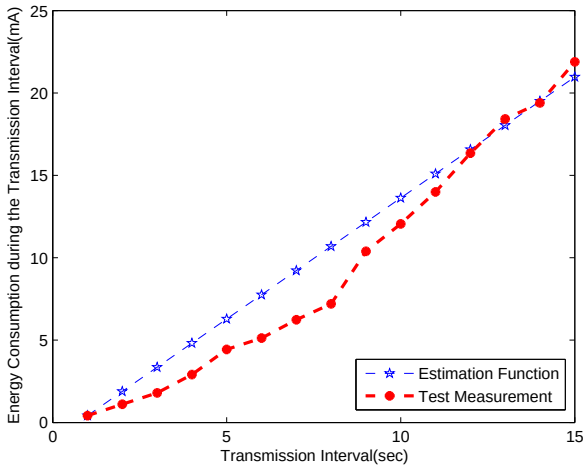

Fig. 6.2: Energy consumption for all Z-stack layers except for the MAC and application layers.

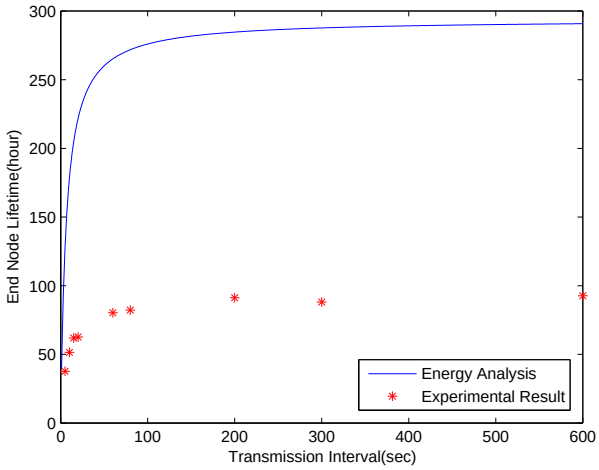

Fig. 6.3: Test results for end node lifetime.

is used to derive the energy consumption estimates. Figure 6.2 shows our energy estimation results for the different layers.

Based on the results shown in Figure 6.2, we use a linear estimation function to provide the term E_{layers}.

Figure 6.3 presents experimental results based on the end nodes employed in

our testbed. From these results, we can verify that end node lifetime increases with increases in the data reporting interval. However, measurements from the actual testbed give shorter end node lifetimes compared to the lifetimes estimated from our energy model analysis. Even if we revise the energy analysis by incorporating the E_{layers} term, the actual measured lifetimes are shorter than what the estimates predict. This indicates that there are other overheads that contribute to overall energy consumption but are not included or not adequately covered by our energy model.

However, the results also demonstrate that the measurement- and analysis-based energy consumption profiles follow similar trends, and thus the results from our analysis can be used to compare alternative design points with reasonable accuracy. This confirms the utility of our energy analysis approach in rapid prototyping, design exploration, and overall design methodology assessment.

6.3 Energy Savings Using ENBESS

To apply ENBESS, we must first ensure that the method is accurate — i.e., that the technique is effective at detecting exceptional conditions when they occur. An energy monitoring setup that has high lifetime but low accuracy will usually be of little use. In this context, we define accuracy under a given application scenario (e.g., based on a given experiment) as the ratio D/A, where D and A represent the number of detected exception cases, and the number of actual exception cases, respectively. If $A = 0$ for a given scenario, then the accuracy for that scenario is

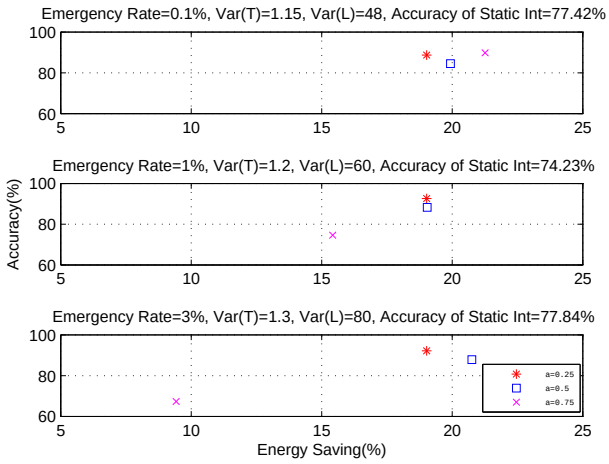

Fig. 6.4: Simulation results for accuracy and energy savings.

undefined.

The accuracy result can be affected by the weighted sum calculation in Algorithm 1 (i.e., the update of the `Fint` variable). Because of the difficulty involved in rigorous testing of exception conditions in an actual environment (e.g., the need to cause sudden swings in temperature, humidity, etc.), we performed accuracy assessment throught simulation. For this purpose, we modeled sensed values as having Gaussian distributions, and as a result, exception conditions were configured to occur during simulation with relatively low, but nonzero probability.

Figure 6.4 shows the accuracy and energy savings for various simulations. This is based on comparison with a static data transmission time interval of 40s, which is selected because it is the largest (most energy efficient) static interval that achieves a target accuracy of 75%.

The tests are held with different emergency rate value. As shown in Figure 6.4, ENBESS usually saves 9% or more energy and increases reporting accuracy com-

End Node Life Time	Hours	Average Interval
ENBESS with a=0.3	76h 15m	527s
ENBESS with a=0.5	77h 12m	512s
ENBESS with a=0.7	74h 23m	507s
Static Transmission Interval(40s)	60h 12m	

Tab. 6.1: Energy saving of ENBESS as determined from the WSNBEMS testbed.

pared to the use of static data transmission intervals. Thus, this value should be examined carefully through simulation to determine an effective setting based on expected operating conditions. The value of the weight values (in the update equation for `Fint`) in general will also affect the results. Analysis of this effect, as well as development of more systematic methods for determining a and the weighting factors for `Fint`, are useful directions for further investigation.

We have also experimented with the ENBESS scheme on real hardware in our WSNBEMS testbed. Table 6.1 shows the energy savings determined from these experiments. These savings are determined in comparison to the same 40s static interval case that was used in Figure 6.4. The results in Table 6.1 demonstrate significant improvement compared to the static interval case.

7. CONCLUSION

In this paper, we have developed methods for system design and analysis of wireless sensor network building energy monitoring systems (WSNBEMs). We have developed methods for application modeling and energy analysis that help to quickly assess the energy efficiency of alternative design configurations. Motivated by results from our energy analysis, we have developed a method, called the End Node Based Energy Saving Scheme (ENBESS), for dynamically adjusting data transmission intervals in a WSNBEMS based on the degree to which sensed data deviates from corresponding moving average values. We have validated ENBESS along with our energy analysis methods in a fully functional, small scale WSNBEMS testbed deployed across multiple rooms in an actual building. Results from experiments on our testbed show that our energy analysis methods are useful in comparing alternative WSNBEMS design configurations, and that ENBESS can achieve significant improvements in energy efficiency and node lifetime through strategic adaptation of transmission intervals.

These developments provide a strong foundation for our proposed future research in WSN design and implementation, which we will discuss in the following section 8.

8. PROPOSED FUTURE WORK

In this section, we present directions for our proposed future work, which will build on the foundations developed in the previous section 7. As mentioned in section 1, WSN applications can be categorized into two groups — monitoring and signal processing applications — and in our previous work, we have focused primarily on design methods for monitoring applications. In our future work, we will explore design methods for signal processing oriented WSN applications.

Signal processing oriented WSN applications continually acquire and process sensed data to detect the starting points associated with relevant events. Such continual operation places heavy demands in terms of energy consumption, as it requires sensor nodes to remain in their active modes all of the time.

To address this energy consumption challenge, we propose to explore multiprocessor signal processing architectures, and associated dataflow graph based design techniques for implementation of embedded multiprocessor software. Rather than employing a high performance signal processor to perform both event detection and signal processing, as is conventionally done for signal processing oriented WSN applications, we propose to employ a low performance, ultra low power microcontroller for event detection in conjunction with a higher performance (but less power efficient) processor for post-detection signal processing.

In a distributed speech recognition application, for example, the low power microcontroller can be employed to continually monitor data from the microphone sensor, and initiate processing on the higher performance processor only when the signal is found to exceed a pre-determined threshold. The higher performance processor can then be kept in its sleep mode during periods of low signal level so that significant power is expended only when required for intensive processing of microphone sensor data.

Such a multi-processor approach for signal processing oriented WSN implementation provides a number of important research challenges, which we propose to address in our future work. These challenges include the following.

- High level application modeling and mapping for reliable coordination of sensor node processing across the low power and high performance processors.

- Power optimization of embedded software for both the low power and high performance processors.

- Optimizing the distribution of computational tasks between the low power and high performance processors. For example, by adding some pre-processing to the low power processor after event detection, we may be able to significantly reduce average energy consumption in the high performance processor. Such optimization, however, must be done carefully to avoid overloading the low power processor and avoid violating real-time performance constraints.

BIBLIOGRAPHY

[1] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E., Wireless Sensor Networks: A Survey, Computer Networks Elsevier Journal ,*38*, 393-422, 2002.

[2] M. Kintner-Meyer, M. Brambley, T.A.Carlon and N.N. Bauman, "Wireless sensors: technology and cost-savings for commercial buildings, Teaming for Efficiency", 2002 ACEEE Summer Study on Energy Efficiency in Buildings ,11, 54-61, 2002.

[3] D.B. Belzer and T.L. Marsh and R.D. Marsh, "Analysis of US commercial building energy use trends", 1972-1991, 1994.

[4] T. A. Reddy, J. K. Kissock, S. Katipamula and D. E. Claridge, "An Energy Delivery Efficiency Index to Evaluate Simultaneous Heating and Cooling Effects in Large Commercial Buildings", Journal of Solar Energy Engineering, 116, 79-87, 1994.

[5] W. Jang, W. M. Healy and M. J. Skibniewski, "Wireless sensor networks as part of a web-based building environmental monitoring system", Automation in Construction ,*17*, 729-736, 2008.

[6] R. Zhang, Z. Jia and D. Yuan, "Analysis of Lifetime of Large Wireless Sensor Networks Based on Multiple Battery Levels", International Journal of Communications, Network and System Sciences ,*2*, 136-143, 2008.

[7] L. Wang and Kulkarni S.S., "Sacrificing a Little Coverage Can Substantially Increase Network Lifetime", Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on ,*1*, 326-335, 2006.

[8] T. V. Padmavathy, "Extending The Network Lifetime Using Optimized Energy Efficient Cross Layer Module (OEEXLM) In Wireless Sensor Networks", Wireless Sensor Network ,*1*, 1-60, 2009.

[9] L. Wang and S. S. Kulkarni, "Sacrificing a Little Coverage Can Substantially Increase Network Lifetime",Sensor and Ad Hoc Communications and Networks,

2006. SECON '06. 2006 3rd Annual IEEE Communications Society on , ,*1*,1281-1300 , 2006.

[10] X. Tang and J. Xu, "Extending Network Lifetime for Precision-Constrained Data Aggregation in Wireless Sensor Networks",25th IEEE International Conference on Computer Communications , 1-12 , 2006.

[11] Chang, Jae-Hwan and Tassiulas, Leandros, "Maximum lifetime routing in wireless sensor networks",IEEE/ACM Trans. Netw. ,,*12*, 609-619, 2004.

[12] R. Mada, S. Cui, S. Lall and A. Goldsmith, "Cross-layer design for lifetime maximization in interference-limited wireless sensor networks", 24th IEEE International Conference on Computer Communications , ,*3*, 1964-1975 , 2005.

[13] Ramachandran, Iyappan and Das, Arindam K. and Roy, Sumit, "Analysis of the contention access period of IEEE 802.15.4 MAC", ACM Transaction on Sensor Network, March 2007, vol. 3.

[14] Kohvakka, Mikko and Kuorilehto, Mauri and Hännikäinen, Marko and Hämäläinen, Timo D., "Performance analysis of IEEE 802.15.4 and ZigBee for large-scale wireless sensor network applications", Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks, 48–57, 2006.

[15] W. Plishker, N. Sane, M. Kiemb, K. Anand and S. S. Bhattacharyya , "Functional DIF for Rapid Prototyping", Proceedings of the International Symposium on Rapid System Prototyping, 17-23 , 2008.

[16] C. Shen, W. Plishker and S. S. Bhattacharyya, "Dataflow-based Design and Implementation of Image Processing",Techreport UMIACS-TR-2011-11, Institute for Advanced Computer Studies, University of Maryland at College Park, *http://drum.lib.umd.edu/handle/1903/11403*, 2011.

[17] C. Shen, W. Plishker, H. Wu and S. S. Bhattacharyya, "A Lightweight Dataflow Approach for Design and Implementation of (SDR) Systems",Proceedings of the Wireless Innovation Conference and Product Exposition , 640–645, 2010.

[18] S. S. Bhattacharyya, E. Deprettere, R. Leupers and J. Takala, "Handbook of Signal Processing Systems", ,*Springer*, 2010.

[19] C. Shen, C. Badr, K. Kordari, S. S. Bhattacharyya, G. L. Blankenship and N. Goldsman, "A Rapid Prototyping Methodology for Application-Specific Sensor Networks", Proceedings of the IEEE International Workshop on Computer Architecture for Machine Perception and Sensing, 130-135 , 2006.

[20] K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson and S. Masri, "Monitoring civil structures with a wireless sensor network",Internet Computing, IEEE , ,*10*, 26-34, 2006.

[21] C. Shen, W. Plishker, H. Wu and S. S. Bhattacharyya, "A Lightweight Dataflow Approach for Design and Implementation of SDR Systems", Proceedings of the Wireless Innovation Conference and Product Exposition, 640-645 , 2010.

[22] Long Zhaohua and Gao Mingjun, "Survey on network lifetime research for wireless sensor networks", Broadband Network Multimedia Technology, IC-BNMT '09. 2nd IEEE International Conference on, 899-902 , 2009.

[23] C. Savarese, J.M. Rabaey and J. Beutel, "Location in distributed ad-hoc wireless sensor networks", Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on, ,*4*, 2037-2040, 2001.

[24] Texas Instruments, "DataSheet: Z-Stack - ZigBee Protocol Stack v.2.4.0", 2010.

[25] Texas Instruments, "Datasheet: CC2530-Second Generation System-on-Chip Solution for 2.4 GHz IEEE 802.15.4 / RF4CE / ZigBee", 2010.

[26] Philips, "Datasheet: I2C-bus specification and user manual",Philips, 2007.