Static Worst-Case Energy and Lifetime Estimation of Wireless Sensor Networks

Yu Liu and Wei Zhang
Department of Electrical and Computer Engineering,
Southern Illinois University Carbondale,
Carbondale, Illinois, USA
{liu,zhang}@engr.siu.edu

Kemal Akkaya
Department of Computer Science, Southern Illinois University Carbondale,
Carbondale, Illinois, USA
kemal@cs.siu.edu

Received 25 February 2010; Revised 24 May 2010; Accepted 8 June 2010

With the advance of computer and communication technologies, wireless sensor networks (WSNs) are increasingly used in many aspects of our daily life. However, since the battery lifetime of WSN nodes is restricted, the WSN lifetime is also limited. Therefore, it is crucial to determine this limited lifetime in advance for preventing service interruptions in critical applications. This paper proposes a feasible static analysis approach to estimating the worst-case lifetime of a WSN. Assuming known routes with a given sensor network topology and S-MAC as the underlying MAC protocol, we statically estimate the lifetime of each sensor node with a fixed initial energy budget. These estimations are then compared with the results obtained through simulation which run with the same energy budget on each node. Experimental results of our research on TinyOS applications indicate that our approach can safely and accurately estimate worst-case lifetime of the WSN. To the best of our knowledge, our work is the first one to estimate the worst-case lifetime of WSNs through a static analysis method.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design, Wireless Communication; C3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Real-time and Embedded Systems

General Terms: Performance, Reliability

Additional Key Words and Phrases: Static Timing Analysis, WSNs, Worst-case Execution Time

---

1. INTRODUCTION

Wireless sensor networks (WSNs) are increasingly used in many aspects of our daily life, such as environmental monitoring, health-care systems, structure health checking, military applications, etc. Such networks normally consist of a large number of tiny sensor nodes that self-organize themselves into a multi-hop wireless network in order to collect ambient information. A sensor node is battery-powered which has a limited lifetime. When the battery depletes, the node basically will not be able to function anymore. Therefore, there has been a significant amount of research in the past in order to extend the lifetime of sensor nodes with intelligent, energy-aware algorithms at different layers of the protocol stack [Akyildiz et al. 2002]. Despite improvements, such algorithms cannot guarantee an unlimited running time for sensors and thus the service of the WSN can be easily interrupted/terminated with the failure of sensors due to battery exhaustion. Therefore, a mechanism to determine the lifetime of individual sensors and thus the whole WSN in advance is of paramount importance for WSN application designers. Such knowledge would be crucial for critical real-time applications such as forest monitoring, military surveillance, target tracking, etc. In addition, it will enable better asset planning before the WSN is deployed.

In the past decade, there has been many research on the predication of energy consumption of sensors [Salhieh et al. 2001; Wang et al. 2006; Barberis et al. 2007; Agnihotri and Nuggehalli 2007; Jayaseelan et al. 2006]. In addition, plenty of radio energy models were proposed in order to estimate the lifetime of the WSNs through simulation [Salhieh et al. 2001; Wang et al. 2006; Barberis et al. 2007; Agnihotri and Nuggehalli 2007]. However, all these previous works can only guarantee estimation of the average-case WSN lifetime, which still makes users face the high risk of unexpected service termination of the whole network if the network design does not base on the worst-case analysis.

Mounier et al. studied the worst-case lifetime of a WSN by using model checking [Mounier et al. 2007]. In general, their approach finds an exhaustive way to provide information about the worst-case. However, their goal is essentially to compare protocols, not to have an absolute value of the network lifetime. Therefore, their estimated results can not be used to guide the design of WSNs. Furthermore, this model-checking approach requests a large amount of computation time. For instance, the computation lasts more than two days for a WSN with eleven nodes. Therefore, an effective static analysis method to bound the absolute value of worst-case lifetime of a WSN is necessary to direct the design of WSNs.

In this paper, we propose an approach to estimating the worst-case lifetime of a WSN, which considers the worst-case energy consumption of each sensor, including its CPU, radio, circuit, and EEPROM. The energy models are chosen based on the widely used MICA2 motes [MPR-MIB 2007]. We estimate the CPU energy consumption by an Integer Linear Programming (ILP) based method on the events occurring in the lifetime of a node. Also, the TinyOS operation system [TinyOS 2010] overhead for the CPU active state energy consumption is considered in our work. We then analyze the worst-case radio energy consumption scenarios for the S-MAC protocol [Ye et al. 2002], and compute the results according to our analysis. Finally, we add the worst-case CPU, radio, sensor circuit and EEPROM energy consumption to calculate the

total energy consumption of a sensor node for a particular S-MAC frame. The lifetime of a node can then be calculated by finding the total number of S-MAC frames for the initial energy budget of the node. Once we know the lifetime of each node, we can determine the lifetime of the network by different strategies, such as the average lifetime of all nodes based, a certain percentage of alive network nodes based, the network connectivity based, the sensor coverage based one, etc. More information about the strategies of determining the WSN lifetime can be found in [Sha et al. 2008; Dietrich and Dressler 2009]. Our experimental results on TinyOS applications based on S-MAC indicate that this approach can safely and accurately estimate the worst-case lifetime of WSNs. Besides, we also study the way of extending our static analysis method to the energy-efficient routing protocols of WSNs.

2. SYSTEM MODEL AND ASSUMPTIONS

In this section, we first provide basic hardware information regarding the sensors we use and then summarize our network assumptions.

In the past years, several hardware platforms for sensors were developed and commercially used in the field, such as Rockwell WINS & Hidra Nodes [Teledyne 2010], Berkeley Motes [Berkeley 2010], UCLA iBadge [iBadge 2010], Crossbow MICA Series Motes (MICAz, MICA2, Mica2Dot) [MPR-MIB 2007], etc. These hardware platforms have a similar architecture to match general requirements of WSNs, including a basic controller, a low power communication device, a memory, a sensing circuit, and a power supply. In this work, we focus on the most commonly used battery-powered platform, named MICA2 mote from Crossbow Inc [MPR-MIB 2007].

WSNs are formed by networking of a large number of sensors (e.g., MICA2 motes).

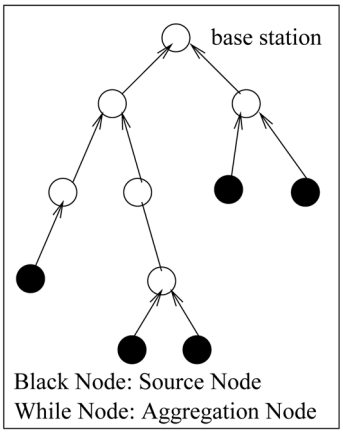


Figure 1. The topology of a data gathering tree for a sample WSN.

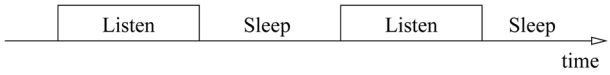


Figure 2. Periodic listen and sleep scheme of S-MAC.

The data sensed at the sensors are collected at a node called base-station (BS) by forming a data gathering tree [Lu et al. 2004] rooted at the BS. In such a data gathering tree, leaf nodes are referred to as source nodes. In addition to these source nodes, there are relay nodes which are responsible for processing/aggregating the data and forwarding it towards the BS. Such nodes are referred to as aggregation nodes. A sample data gathering tree is shown in Figure 1.

In our setup, source nodes generate packets periodically. Aggregation nodes on the other hand do not sense the environment but rather collect data from their child nodes and perform some CPU processing (e.g., averaging the readings, calculating histogram, etc) before relaying the data upwards. Finally, we assume that the peer communication is reliable once pair nodes reserve the wireless medium. In other words, no re-transmission is needed once the peer communication is set up. Accordingly, we disable such re-transmission during simulation. However, our analysis can be easily adapted to remove this assumption by considering the maximum re-transmission configuration.

3. S-MAC AND ITS ANALYSIS

3.1 Overview of S-MAC

In this paper, we use the S-MAC protocol as our media access layer. S-MAC is one of the first energy-efficient MAC protocols used in WSNs. A full software implementation on TinyOS is freely available [TinyOS 2010]. S-MAC is designed to reduce energy consumption by introducing four major components: periodic listening, collision avoidance, overhearing avoidance, and message passing.

Periodic listen and sleep is the major component of S-MAC, which greatly addresses the issue of energy waste due to idle listening. Each node goes to sleep for some time, and then wakes up and listens to see if any other node wants to talk to it. During sleeping, the node turns off its radio, and sets a timer to awake itself later [Ye et al. 2002]. Periodic listen and sleep of S-MAC is illustrated in Figure 2. Normally, S-MAC synchronizes all the nodes to ensure that they can listen and sleep at the same time. Therefore, nodes periodically exchange synchronization packets to maintain their synchronization. Collision avoidance is achieved by following similar procedures in IEEE 802.11 specification, (Network Allocation Vector, Carrier Sense and RTS/CTS mechanism). Forcing interfering nodes sleep after they hear a RTS/CTS packet can avoid overhearing efficiently. Finally, message passing is introduced to reduce the control packet overhead. The approach is to fragment the long message into many small fragments, and transmit them in burst, and only one RTS and one CTS packet are used [Ye et al. 2002].

The S-MAC protocol utilizes frames and periods [Mchenry and Heidemann 2007] to organize the sleep and synchronization schedules. A *frame* is the combination of listen and sleep intervals. *Listen interval* is the duty cycles of S-MAC communication, and divided into two segments namely SYNC and DATA. The *SYNC segment* is used for sending/receiving synchronization packets. The *DATA segment*, on the other hand, is used for RTS/CTS exchange. Furthermore, each segment is composed of some number of *slots*, where each slot represents an opportunity for one node to capture the channel

for sending a control packet, and several slots are provided to allow randomization to reduce the likelihood of collision [Mchenry and Heidemann 2007]. A *period* is the interval of synchronization packets transmission to neighboring nodes.

S-MAC frames can be categorized into Tx, Rx and idle frames according to the status of DATA packets communication. The S-MAC frame with DATA packet transmission is named the *Tx frame*, while the frame with DATA packet receiving is the *Rx frame*. Thus, the S-MAC frame without any DATA packet operation is called the *idle frame*. In addition, the *DATA packet transmission interval* is the number of S-MAC frames between two adjacent Tx frames.

Our worst-case energy consumption scenarios analysis regards that all nodes in the designed WSN can work correctly. In other words, all nodes can be synchronized correctly which is requested by the S-MAC. If nodes cannot be synchronized, they will not be able to follow listen/sleep scheme, which is the basis for our analysis. In addition, desynchronization will force radio into greedy neighbor searching to consume much more energy. Normally, such assumption can be guaranteed in the run-time, since it is a basic requirement of a good WSN design.

3.2 Analysis of S-MAC

In this section, we will identify the scenarios for S-MAC which will result in the worst-case energy consumption.

3.2.1 *Radio Transmission.* In S-MAC, three different types of packets are transmitted through radio: SYNC packets, control packets (RTS, CTS and ACK), and DATA packets. We now explain worst-case scenarios for each of these packet types:

- **SYNC Packets:** Each node periodically broadcasts SYNC packets to its neighbors even if it has no followers [Ye et al. 2002]. Although SYNC packets may encounter collision or interference problems at the receiver side, no re-transmission is done for SYNC packets in S-MAC. Therefore, for a sensor node at most one SYNC packet will be transmitted in each synchronization period.
- **RTS/CTS packets:** S-MAC adopts the RTS/CTS exchange mechanism to reserve medium for collision avoidance. The node which first sends out the RTS packet wins the medium, and the receiver will reply with a CTS packet [Ye et al. 2002]. It is important to note that when a sensor node starts transmitting its data and there is no time left in the DATA segment according to its schedule, this does not mean that the node immediately stops transmitting and goes to sleep mode. In other words, the node will not be forced into sleep until its DATA packet transmission is finished. Therefore, the worst-case energy consumption scenario occurs when a node wins the medium at the very end of its regular DATA segment, which requires the node to extend its radio duty cycle maximally.
- **DATA/ACK packets:** As mentioned earlier, we assume that the stable peer communication cannot be interfered once the medium is reserved by RTS/CTS exchange. Therefore, we do not consider the re-transmission of DATA packets. As a result, the ACK packet can be successfully received after each DATA packet transmission.

3.2.2 *Radio Receiving.* When the radio wakes up, it will firstly enter the idle state. It enters the receiving state once it detects carrier in the wireless medium. However, it is impossible for us to identify the exact radio receiving time, since all types of interfering transmission and even noise can drive radio into the receiving state at run-time. Therefore, all radio duty time in both the listen and sleep intervals excluding transmission time is regarded as receiving time. This scenario may seldom occur, but it is the worst-case one. In addition, RTS packet receiving could occur at the end of duty cycle which causes a break in the listen/sleep scheme, and thus prolong the duty cycle to the fullest extent possible.

3.2.3 *Topology Constraints.* Worst-case energy consumption scenarios will differ based on the types of nodes in the data gathering tree. For instance, the leaf nodes only transmit DATA packets. Thus, they will only have Tx and idle frames. While source nodes are trying to transmit DATA packets to their parent nodes during the Tx frames, they will not do such an action in idle frames. Aggregation nodes, on the other hand, have both child nodes and parent nodes. Thus, their frames are categorized into Tx, Rx, and idle frames. The number of Rx frames and idle frames depend on both the number of child nodes as well as DATA packet transmission intervals. Finally, the base station does not have any Tx frames, because it is the center of the network for gathering data.

4. WORST-CASE ENERGY AND LIFETIME ANALYSIS

4.1 Energy Models

4.1.1 *The Energy Model of The Radio.* The energy consumed in the radio depends on the number of message bits received/transmitted and the per bit energy cost required [Agnihotri and Nuggehalli 2007]. The number of message bits received/transmitted can be transformed into the radio receiving/transmission duration on the specific hardware platform. Therefore, we can use Equation 1 to calculate the energy consumption of radio, where U_{radio} is the voltage of radio component, I_{radio} is the current of radio. *Time* is the radio receiving/transmission duration, which is computed through dividing the number of transmitted/received message bits by the baud rate.

Table I. Current notations and values on a MICA2 mote.

State	Notation	Current (A)
CPU Active	I_{cpu_active}	0.0075667
CPU Idle	I_{cpu_idle}	0.0033433
Radio Tx (15M)	I_{radio_tx}	0.01134
Radio Rx	I_{radio_rx}	0.0096
Sensor Circuit	I_{sensor}	0.0007
EEPROM Standby	$I_{eeprom\_standby}$	0.000002

$$E_{radio} = U_{radio} \times I_{radio} \times Time \quad (1)$$

4.1.2 *The Energy Model of The CPU.* Our work focuses on the popular MICA2 mote that uses AVR Atmega128L CPU. The AVR Atmega128L is an 8-bit CPU with 128k in-system programmable flash. It is a simple CPU without employing advanced microprocessor features, such as multi-stages pipeline, cache, out-of-order execution, etc. Thus, instruction-level energy estimation techniques are quite accurate for simple processor architectures [Jayaseelan et al. 2006]. The energy consumption of each cycle is fixed. In addition, we can know the maximum needed cycles of each instruction from the datasheet of AVR Atmega128L CPU [Atmega128L 2007].

Some AVR instructions request different number of cycles depending on the exact instruction. In order to bound the worst-case energy consumption through static analysis, we select the maximum requested cycles of these instructions in the active state. By adding up these maximum requested cycles, we can calculate the total worst-case number of cycles needed in the active state. Then, we multiply this number with the time per cycle to calculate the total time the CPU staying in the active state. Excluding the active state time, all the remaining time is in the idle state. Therefore, we can compute the CPU energy consumption based on Equation (2), where U_{cpu} is the voltage of CPU, I_{cpu} is the current of CPU, and *Time* is the duration the CPU staying in the active or idle state.

$$E_{cpu} = U_{cpu} \times I_{cpu} \times Time \quad (2)$$

4.2 Overview of Our Approach

The majority of the energy-efficient MAC protocols specifically designed for WSNs are based on a periodic listen/sleep scheme. Therefore, our basic idea of bounding the worst-case lifetime of WSN nodes is to statically estimate the worst-case energy consumption in each S-MAC frame, and then compute the number of frames per a specific battery energy budget based on the worst-case energy consumption in each frame. Since we know the duration of one S-MAC frame, we can acquire the worst-case lifetime through the minimal number of frames.

Our proposed static approach can be summarized into three steps. First, our approach estimates the maximum number of CPU cycles of each event occurring in one S-MAC frame through the ILP method based on the control flow graph (CFG). By adding up all maximum CPU active state time of all events, we can bound the worst-case CPU time in the active state, and then estimate the worst-case CPU energy consumption. The detailed method and algorithm about utilizing ILP is described in Subsection 4.3. Second, our approach predicts the worst-case radio energy consumption in each frame based on our worst-case energy consumption scenarios analyzed in the previous sections. The details will be presented in Subsection 4.4. Finally, our approach computes the minimal lifetime of each node based on the worst-case energy consumption in each S-MAC frame as detailed in Subsection 4.5. Having the information of worst-case lifetime of all nodes, the network lifetime can be bounded by different strategies.

We also note that we need to consider the energy consumption from both sensor circuit and EEPROM. The sensor circuit is requested to be always powered on by the simulator, and we do not have EEPROM read and write operations in our applications. Thus, EEPROM is only in the standby state.

4.3 CPU Energy Consumption Estimation

The CPU has two states including the active and the idle one. S-MAC utilizes an event-driven WSN operation system called TinyOS and thus our analysis will focus on this operation system. The CPU will turn into the idle state from the active one if no TinyOS task requested by events is waiting for executing in the task queue. Since the energy consumption in the idle state is less than that in the active one, we need to obtain an upper bound of the CPU active time through static analysis.

We need to analyze all events occurring in one S-MAC frame to find the maximum CPU active time through an ILP based method on the CFG information. Also, the overhead of TinyOS system should be considered. TinyOS system has an infinite loop running in the whole lifetime of a node to process the tasks requested by the events. Since only one task can be handled during each iteration, the number of iterations of this loop can be bounded by the number of requested tasks in the worst-case energy consumption scenarios. Moreover, the number of these tasks can be calculated from the weight of basic blocks in the codes of each event, which is the result of solving the ILP formulas to obtain the WCET.

4.3.1 *Basic Categorization of CPU Energy Consumption.* We can categorize all events occurring in one S-MAC frame into several groups, and find the theoretical worst-case energy consumption for each. By adding up these data, we can get the worst-case energy consumption of one frame. Events occurring within one S-MAC frame is illustrated in Figure 3.

(1) At the beginning of system execution on the node, it has an *initialization event* prior to the start of S-MAC frames. Its WCET is denoted as T_{init}.
(2) After the system initialization, the node will begin its periodic events in S-MAC frames. One of them is the *sample event* to sample environmental data by the sensor circuit, whose WCET is denoted as T_{sample}.
(3) After getting the data from the sensor, The *tx application event* is needed to process the sensed data. The WCET of this event is denoted as T_{app_tx}. Similarly, in case of a reception, a sensor node needs to process the received data, and this event is called the *rx application event.* Also, the WCET of this event is denoted as T_{app_rx}.
(4) The *byte rx event* and *byte tx event* are the byte ready interrupt services for the

Figure 3. The events in the S-MAC frame (I Event: the initialization event, S Event: the sample event, A1 Event: the tx application event, A2 Event: the rx application event, T Events: a series of tx byte events and rx byte events, C Events: a series of S-MAC clock events).

communication, and their WCET are denoted as T_{byte_rx} and T_{byte_tx}. Also, the maximum number of occurrences of these events are bounded by the maximum number of bytes received and transmitted within one S-MAC frame.

(5) S-MAC has an important timer fired event, called the *s-mac clock event*. This event occurs every 1 ms to serve all features of S-MAC, including the periodical sleep scheme, the control message exchange, etc. The WCET of this event is denoted as T_{smac_clock}.

(6) The total worst-case CPU active time in one S-MAC frame can be obtain by Equation (3) with considering the TinyOS overhead denoted as $T_{tinyos}^{overhead}$, where $n1$ is the maximum number of bytes received in one S-MAC frame, $n2$ is the maximum number of bytes transmitted in one S-MAC frame, and $n3$ is the number of S-MAC clock timer fired.

(7) Besides all these events above and TinyOS overhead, CPU will be in the idle state during the rest of time in one S-MAC frame. The CPU idle time in one S-MAC frame is called T_{idle}, T_{smac} is the duration of one S-MAC frame. Thus, we can compute T_{idle} by Equation (4). In addition, all notations used for CPU active time analysis are summarized in Table II.

$$T_{active} = T_{app\_rx} + T_{app\_tx} + T_{sample} + n1 \times T_{byte\_rx} + n2 \times T_{byte\_tx} + n3 \times T_{smac\_clock} + T_{tinyos}^{overhead} \quad (3)$$

$$T_{idle} = T_{smac} - T_{active} \quad (4)$$

4.3.2 *Bounding Worst-Case Execution Time for The Events.* To bound the WCET for all events requested for the worst-case CPU energy consumption analysis, we need to set up an ILP formulation including the objective function and constraints [Jayaseelan et al. 2006; Li and Malik 1995] following the CFG of this program which can be automatically generated in our approach. Equation 5 is the objective function for estimating the WCET, where ci is the execution time of each basic block, and b_i

Table II. Notations used for CPU active time analysis.

Time	Notation
initialization event	T_{init}
tx application	T_{app_tx}
rx application	T_{app_rx}
tx byte event	T_{byte_tx}
rx byte event	T_{byte_rx}
s-mac clock timer event	T_{smac_clock}
sample event	T_{sample}
TinyOS overhead	$T_{tinyos}^{overhead}$
CPU idle time	T_{idle}
CPU active time	T_{active}
S-MAC frame time	T_{smac}

is the weight of each basic block.

$$\max \sum_{i=1}^{n} \{c_i \times b_i\} \tag{5}$$

$$b_i - \sum_{j=1}^{n} \{d_j^{i\_in}\} = b_i - \sum_{j=1}^{n} \{d_j^{i\_out}\} = 0 \tag{6}$$

$$b_i - \sum_{j=1}^{n} \{f_j^{i\_in}\} = 0 \tag{7}$$

$$b_i - f_j^{i\_out} = 0 \tag{8}$$

The ILP constrains are derived from the CFG of the program. The relationship between the caller and callee functions is presented as function call edges [AVRORA 2010]. The ILP constraints are based on the conclusion that the execution count of each basic block must be equal to both the sum of the control flow going into it and the sum of the control flow going out of it [Li and Malik 1995], which are presented as equality equations. Equations (6), (7) and (8) are the equality ones, where $d_j^{i\_in}$ is the weight of the in-edge, $d_j^{i\_out}$ is the weight of the out-edge, $f_j^{i\_in}$ is the weight of the function call in-edge, and $f_j^{i\_out}$ is the weight of the function call out-edge of basic block i. Equation (8) is needed for each function call out-edge of basic block i separately. In addition, the range of loop bounds should be denoted as ILP constraints, which are presented as inequality equations. Equation (9) is the inequality equation, where e_i^{entry} is the entry edge of the header block h_i of loop i, and m is the upper bound of the number of iterations of this loop.

$$m \times e_i^{entry} - h_i \geq 0 \tag{9}$$

The algorithm for generating the ILP objective function and constraints is summarized in Figure 4. First, our algorithm generates the objective function, which indicates that our objective is to obtain maximum execution cycles of the target program (Lines 1-7). Second, our algorithm exhausts all basic blocks to generate structural constraints between basic blocks and their entry edges except the entry basic block of each function in the program (Lines 8-14), as well as equality structural constraints between basic blocks and their exit edges except the exit basic block of each function in the program (Lines 15-18), and structural constraints between basic blocks and their function call edges (Lines 19-24). Third, our algorithm exhausts all loops to generate the inequality constraints based on the relation between the entry edge of this loop and the header basic block (Lines 25-32). Last, our algorithm specifies that the weight of all basic blocks in a target program should be equal or greater than zero (Lines 33-38), and the weight of entry basic block of the whole program should be one (Line 39). It should be noted that this algorithm can be computed efficiently. Suppose a loop has N basic blocks and M Loops, the complexity of our algorithm is $O(3 \times N + M)$.

An example is shown in Figure 5 to illustrate the method of building equality equations of ILP constraints in our work for bounding worst-case energy consumption of the program. Figure 5(a) demonstrates the CFG of this example, and Figure 5(b)

depicts the constraints generated. Figure 6 is an example to show how to build inequality equations of ILP constraints in our research. Figure 6(a) shows the example codes and its corresponding basic blocks, and Figure 6(b) shows the related CFG and constraints generated.

4.4 Radio Energy Consumption Estimation

Radio energy consumption estimation follows our analysis to S-MAC strictly. In the data gathering tree, source nodes, aggregation nodes and the base station have several different S-MAC frames. In order to predict the worst-case lifetime, we need to estimate the upper bound of radio energy consumption for each of these frames. In addition, all notations used for following radio energy consumption estimation are summarized in Table III.

4.4.1 *Source Nodes.* Source nodes are leaf nodes in the tree structured network, and

The Algorithm to Build ILP Objective Function and Constraints for Maximum Execution Cycles

```
1: B: set of basic blocks; L: set of loops; ENTRY: entry block; EXIT: exit block; ILP: ilp file;
2: WHILE (B ≠ ∅) DO {
3:    pick a basic block b ∈B;
4:    calculate the maximum cycles of b;
5:    ILP< information following the format "total cycles of b * b +";
6:    B = B - b;
7: }
8: reinitialize B;
9: WHILE (B ≠ ∅) DO {
10:   pick a basic block b ∈B;
11:   IF (b != ENTRY of the function where b locates) DO {
12:      filter all entry edges of b;
13:      ILP< information following the format "b -- all entry edges of b = 0";
14:   }
15:   IF (b != EXIT of the function where b locates) DO {
16:      filter all exit edges of b;
17:      ILP< information following the format "b - all exit edges of b = 0";
18:   }
19:   IF (b contains function call edges) DO {
20:      filter all function call edges of b;
21:      ILP< information follwoing the format ""b - all function call edges of b = 0";
22:   }
23:   B = B - b;
24: }
25: WHILE (L ≠ ∅) DO {
26:   pick a loop l∈ L;
27:   obtain the upper bound iteration number of l;
28:   obtain the header block of l;
29:   obtain the entry edge from the preheader basic block of l;
30:    ILP< information following the format "max # of iteration x entry edge - header >= 0";
31:    L = L - l;
32: }
33: reinitialize B;
34: WHILE (B ≠ ∅) DO {
35:   pick a basic block b ∈B;
36:   ILP< information following the format "the weight of b >= 0";
37:   B = B - b;
38: }
39: ILP < weight of ENTRY in this program should be 1;
```

Figure 4. The algorithm to build the integer linear programming objective function and constraints.

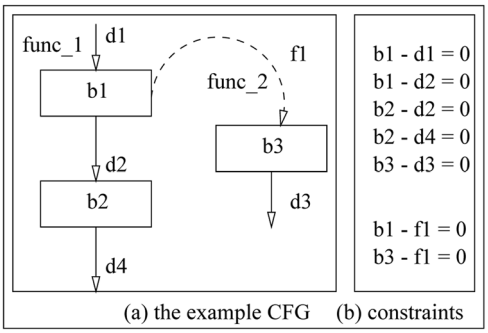


Figure 5. An example for building equality equations of ILP constraints.

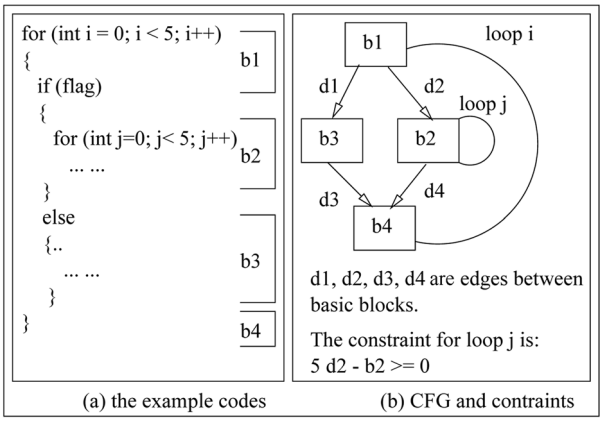


Figure 6. An example for building inequality equations of ILP constraints.

responsible for probing the surroundings and transmit the sensed data to aggregation nodes. Thus, source nodes do not have S-MAC frames to receive DATA packets, as a result we only need to compute the worst-case energy consumption in Tx frames and idle frames.

According to our analysis on S-MAC, the worst-case scenario is to break the listen/sleep scheme to extend duty cycles by the greatest extent. In the worst-case scenario, the RTS packet is only successfully transmitted at the end of the regular DATA segment. Thus, the radio receiving time in the regular DATA segment is $T_{txframe\_dataseg\_rx} = T_{dataseg} - T_{rts\_tx}$. The extended radio transmission time is $T_{txframe\_extend\_tx} = T_{data\_tx}$.

The extended radio receiving time is $T_{txframe\_extend\_rx} = T_{txframe\_space1} + T_{cts\_rx} + T_{txframe\_space2} + T_{txframe\_space3} + T_{ack\_rx}$.

In the SYNC segment, the segment will have one SYNC packet transmission for only one frame in each synchronization period, and the whole segment is used for receiving SYNC packets from neighboring nodes for other frames in this period. The worst-case radio receiving time for the latter circumstance is $T_{synseg\_rx\_nosyntx} = T_{synseg}$, while that is $T_{synseg\_rx\_synctx} = T_{synseg} - T_{sync\_tx}$ for the first one. The radio transmission time in the latter circumstance is T_{sync_tx}.

Table III. Notations used for radio energy consumption estimation.

Description	Notation
RTS packet transmission time	T_{rts_tx}
CTS packet transmission time	T_{cts_tx}
CTS packet receiving time	T_{cts_rx}
DATA packet transmission time	T_{data_tx}
DATA packet receiving time	T_{data_rx}
ACK packet transmission time	T_{ack_tx}
ACK packet receiving time	T_{ack_rx}
SYNC packet transmission time	T_{sync_tx}
DATA segment duration	$T_{dataseg}$
SYNC segment duration	T_{synseg}
radio listen interval duration	T_{radio_listen}
radio transmission time in the SYNC segment	T_{sync_tx}
SYNC segment radio receiving time without SYNC packet Tx	$T_{synseg\_rx\_nosyntx}$
SYNC segment radio receiving time with SYNC packet Tx	$T_{synseg\_rx\_synctx}$
total radio receiving time in the Tx frame	$T_{txf\_rame\_radio\_rx}$
total radio transmission time in the Tx frame	$T_{txframe\_radio\_tx}$
radio receiving time in the DATA segment of the Tx frame	$T_{txframe\_dataseg\_rx}$
extended radio transmission time of the Tx frame	$T_{txframe\_extend\_tx}$
extended radio receiving time of the Tx frame	$T_{txframe\_extend\_rx}$
maximum time to wait for the CTS packet in the Tx frame	$T_{txframe\_space1}$
SIFS in IEEE 802.11 specification for the Tx frame	$T_{txframe\_space2}$
maximum time to wait for the ACK packet in the Tx frame	$T_{txframe\_space3}$
worst-case radio duty cycle time in the Tx frame	$T_{txf\_rame\_radio\_duty}$
total radio receiving time in the idle frame	$T_{idleframe\_radio\_rx}$
worst-case radio duty cycle time in the idle frame	$T_{idleframe\_radio\_duty}$
radio receiving time in the DATA segment of the Rx frame	$T_{rxframe\_dataseg\_rx}$
extended radio transmission time of the Rx frame	$T_{rxframe\_extend\_tx}$
extended radio receiving time of the Rx frame	$T_{rxframe\_extend\_rx}$
SIFS in IEEE 802.11 specification for the Rx frame	$T_{rxframe\_space1}$
SIFS in IEEE 802.11 specification for the Rx frame	$T_{rxprame\_space2}$
SIFS in IEEE 802.11 specification for the Rx frame	$T_{rxframe\_space3}$

Therefore, the total worst-case radio receiving time in Tx fragment is: $T_{txframe\_radio\_rx} = T_{synseg\_rx\_syntx} + T_{txframe\_dataseg\_rx} + T_{txframe\_extend\_rx}$ with SYNC packet transmission, or $T_{txframe\_radio\_rx} = T_{synseg\_rx\_nosynctx} + T_{txframe\_dataseg\_rx} + T_{txframe\_extend\_rx}$ without SYNC packet transmission. Also, the total worst-case radio transmission time in this kind of fragment is: $T_{txframe\_radio\_tx} = T_{rts\_tx} + T_{txframe\_extend\_tx}$ without SYNC packet transmission

or $T_{txframe\_radio\_tx} = T_{sync\_tx} + T_{rts\_tx} + T_{txframe\_extend\_tx}$ with SYNC packet transmission. With the estimated worst-case radio transmission and receiving time, we can predict its energy consumption by the energy model.

The total worst-case radio duty cycle time in this frame can be computed as $T_{txframe\_radio\_duty} = T_{radio\_listen} + T_{txframe\_extend\_tx} + T_{txframe\_extend\_rx}$.

The idle frame is much simpler than the Tx frame, and the radio just follows the regular listen/sleep scheme of S-MAC. For the frame with the SYNC packet transmission, its worst-case radio receiving time is $T_{idleframe\_radio\_rx} = T_{radio\_listen} - T_{sync\_tx}$, transmission time is simply T_{sync_tx}. The worst-case radio receiving time is $T_{idleframe\_radio\_rx} = T_{radio\_listen}$ for the idle frame without the SYNC packet transmission. Thus, the worst-case radio duty cycle is purely $T_{idleframe\_radio\_duty} = T_{radio\_listen}$.

4.4.2 *Aggregation Nodes.* Aggregation nodes are intermediate nodes in the tree structured network, and responsible for data aggregation, relaying, and further processing. Thus, aggregation nodes have full set of S-MAC frames, including the Tx, Rx, and idle frames. The worst-case energy consumption estimation approach in Tx and idle frames is the same as source nodes.

Similarly, the worst-case scenario in the Rx frame is to break the listen/sleep scheme to extend the DATA segment to the greatest extent. In the worst-case scenario, the RTS packet is only successfully received at the end of the regular listening interval, all other control and DATA packet transmissions occur during the extended interval. Thus, the radio receiving time in the regular DATA segment is simply $T_{rxframe\_dataseg\_rx} = T_{dataseg}$. The extended radio transmission time is represented as $T_{rxframe\_extend\_tx} = T_{cts\_tx} + T_{ack\_tx}$. The extended radio receiving time is represented as $T_{rxframe\_extend\_rx} = T_{rxframe\_space1} + T_{rxframe\_space2} + T_{data\_rx} + T_{rxframe\_space3}$. Finally, equations for the SYNC segment are identical to Tx frames. Equations to compute the worst-case radio receiving and transmission time in Rx frames are similar to Tx frames. Obviously, the maximum number of Rx frames in one DATA transmission interval is determined by the number of child nodes and the length of their DATA transmission intervals.

4.4.3 *The Base Station.* The base station is the root in the tree structure network, and in charge of gathering all the data fluxing in the WSN. Consequently, only Rx and idle frames occur in the base station. The worst-case energy consumption estimation approach in Rx and idle frames is the same as the above equations.

4.5 Worst-Case Lifetime Estimation

We can compute the energy consumption of different CPU and radio states in one SMAC frame for the base, the source and the aggregation node by substituting the time staying in different states into the energy model Equations (1) and (2), which are detailed in Subsection 4.3 and 4.4. Then, we obtain the worst-case energy consumption in one SMAC frame by adding up the energy consumption of different states. Thus, with a specific battery energy budget, we can estimate the worst-case life-time for each node depending on their types and positions in the network. Basically, it may have non worst-case scenarios occurring in the runtime. However,

if such cases happen, it reduces the energy consumption and then prolongs the WSN lifetime. Thus, our static worst-case lifetime estimation is reasonably safe. Actually, we provide a safe upper bound of energy consumption of each stage, but may lead to the overestimation of worst-case energy consumption and lifetime.

Our algorithm is summarized in Figure 7. At the beginning, we compute the WCEC of the first ten S-MAC frames, since the S-MAC implementation on TinyOS indicates that radio does not go to sleep in the first ten frames for neighbour searching (Lines 3-7). Our algorithm adds up the energy consumption of each S-MAC frame until the battery energy budget exhaustion (Line 8). According to the type of the target node, we have different processing based on our analysis to different worst-case energy consumption scenarios for the base node (Lines 9-10), the source node (Lines 11-12), and the aggregation node (Lines 13-14). The sensor circuit and EEPROM energy consumption are also considered. We assume that the sensor circuit is always powered on, and EEPROM is only in the standby state (Lines 15-16). Also, we need to count the number of S-MAC frames elapsed (Line 17). Finally, our algorithm transforms the number of S-MAC frames under the specific battery energy budget into the exact time unit. Thus, we can obtain the worst-case lifetime of the target node by adding the life-time of all S-MAC frames elapsed (Line 19). As we mentioned, once we know the life-time of each node in the WSN, we can determine the life-time of the WSN by specific strategies. Suppose a WSN has N nodes and M S-MAC frames can be supported by the initial energy budget, the complexity of our algorithm is $O(N \times M)$.

4.6 Extension to Routing Protocols

In recent years, researchers also put efforts on the routing protocols to improve the

The Algorithm to Estimate the Worst-case Life-time for WSN Nodes

```
1:  N: the set of nodes;  E: energy consumed;  T: time elapsed;
2:  P: # of tracking periods;  B: battery energy budget;  D: tracking period duration;
3:  WHILE (N ≠ ∅) DO {
4:    pick a node n ∈ N;
5:    process the computation of WCEC of the first 10 s-mac frames;
6:    E = E + WCEC of first 10 s-mac frames;
7:    T = T + time of first 10 s-mac frames;
8:    WHILE (E < = B) DO {
9:      IF (n is a base node) DO
10:       E = E + computation result of WCEC in one s-mac frame for base;
11:     ELSE IF (n is a source node) DO
12:       E = E + compuation result of WCEC in one s-mac frame for source;
13:     ELSE IF (n is a aggregation node) DO
14:       E = E + computation result of WCEC in one s-mac frame for aggregation;
15:     E = E + EEPROM standby energy consumption in one s-mac frame;
16:     E = E + sensor circuit power on energy consumption in one s-mac frame;
17:     P = P + 1;
18:   }
19:   T = T + P x D;
20:   N = N - n;
21: }
```

Figure 7. The algorithm to estimate the worst-case lifetime of WSN nodes.

energy efficiency of WSNs besides the MAC protocols. To minimize energy consumption, routing techniques proposed for WSNs employ some well-known routing tactics as well as tactics special to WSNs, e.g., data aggregation and in-network processing, clustering, different node role assignment, and data-centric methods were employed [Al-Karaki and Kamal 2004]. Although our preliminary work focuses on a network with the predetermined routing paths and an energy-efficient MAC protocol (i.e. S-MAC), we believe that our static analysis approach can be extended to handle networks with energy-efficient routing protocols through analyzing their worst-case energy consumption and lifetime scenarios.

Let us take the widely used the LEACH (Low Energy Adaptive Clustering Hierarchy) protocol as example to present the way of statically analyzing the worst-case energy consumption and lifetime scenarios. LEACH randomly selects a few sensor nodes as the cluster heads and rotates this role in each round to distribute the energy load among the sensors in the network [Al-Karaki and Kamal 2004]. Also, the cluster head nodes compress data arriving from the normal nodes in the respective cluster, and send an aggregated packet to the base station, which lead to the cluster head nodes consume significantly more energy than the normal nodes. More information about the LEACH protocol can be found in [Heinzelman et al. 2000]. In LEACH, the number of nodes selected as cluster head nodes are based on a predetermined factor P. Suppose we has total N nodes in the network, and then we have the number of C cluster head nodes, where $C = N \times P$. Obviously, the worst-case energy consumption scenario of each node is that this node is always chosen as the cluster head node in each round during its lifetime. When we do the worst-case lifetime estimation, we can always choose specific C nodes as the cluster head nodes round by round till the depletion of their energy budget. Then, we choose other C nodes as the header nodes from the remaining alive nodes till the depletion of their energy budget as well. We repeat this operation till we run out the energy budget of all the N nodes in the network in order to get the worst-case lifetime estimation. Although this worst-case scenario may not happen in the runtime and lead to overestimation, the worst-case lifetime estimation result is surely safe. Moreover, our future study can also propose the worst-case lifetime predictable head nodes chosen algorithm instead of the random chosen one in LEACH to offer the capability of obtaining tighter estimation.

5. EXPERIMENTAL EVALUATION

5.1 Methodology

The framework to evaluate our proposed static approach is shown in Figure 8. In our approach, the TinyOS application file written in NesC language will firstly translated to a C language file by the NesC pre-compiler. Then, all events C codes that need to be analyzed are compiled by the avr-gcc compiler [AVR-GCC 2010] to generate binary files, and with the same gcc optimization level set for compiling the entire TinyOS. The binary files can be transformed into the assembly files by the auxiliary tools of avr-gcc. These assembly files are the inputs to our WCET analyzer. Our WCET analyzer is implemented by extending the CFG generation feature of a specific WSN

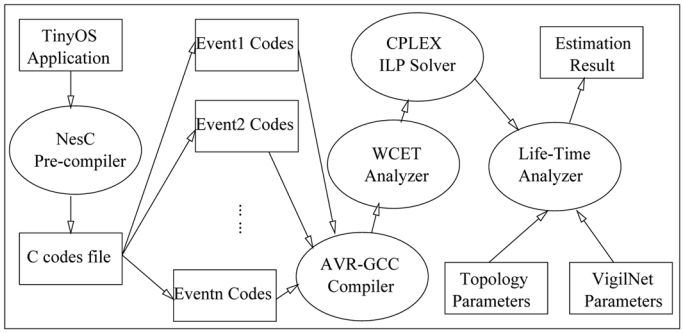


Figure 8. The static worst-case lifetime estimation framework.

simulator named AVRORA [Titzer et al. 2005]. The output of WCET analyzer is the file containing ILP objective function and constraints, which will be solved by a commercial ILP solver CPLEX [CPLEX 2010]. Finally, the lifetime analyzer will generate our estimated results based on WCET analysis results, the network topology and S-MAC parameters.

Besides the static analysis, we also performed the simulation by AVRORA. AVRORA can run the actual AVR microcontroller programs [Titzer and Palsberg 2005], and accurately simulate the devices and the radio communication. This simulator has an energy consumption analysis tool (AEON) [Landsiedel et al. 2005] to simulate the energy consumption of each component on the MICA2 mote, including the radio, the CPU, etc. Also, this simulator utilizes the same energy model used in our static analysis.

Several typical applications are developed on TinyOS with the S-MAC protocol implementation for our experimental evaluation. The characteristics of these TinyOS applications are shown in Table IV. These TinyOS applications work on a specific tree topology with 8 nodes to generate five sample sensor networks, as shown in Table V.

Table IV. The characteristic of TinyOS applications.

Application	Description	Usage
threshold	it periodically probes the environment and transmit the abnormal sensed data which exceeds a pre-defined threshold.	source node
average	it computes the average value of all received data, transmit it, and is a typical data aggregation application.	aggregation node
histogram	it generates histograms based on all received data or sensed data in a specific period of time.	source/aggregation node
maximum	it compares all received data, send out maximum one in a fixed time period, and is a typical data aggregation application.	aggregation node
binary search	it does not transmit the data which is from measuring surroundings directly but from the binary searched result.	source node

Table V. Programs run at each node in our WSN.

sample	1	2	3	4	5
node 0	threshold	histogram	threshold	threshold	binary search
node 1	threshold	histogram	threshold	threshold	binary search
node 2	average	histogram	maximum	histogram	histogram
node 3	base station	base station	base station	base station	base station
node 4	average	histogram	maximum	histogram	histogram
node 5	threshold	histogram	threshold	threshold	binary search
node 6	threshold	histogram	threshold	threshold	binary search
node 7	threshold	histogram	threshold	threshold	binary search

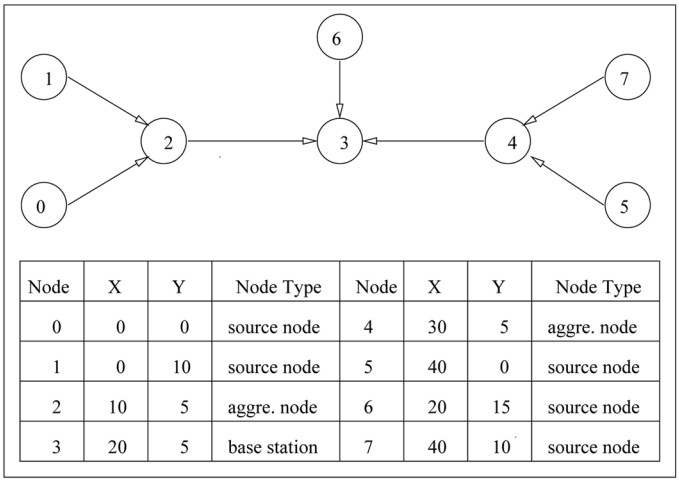
Table VI. DATA packets transmission intervals of all samples.

sample	1	2	3	4	5
node 0	1 frame	10 frames	10 frames	6 frames	1 frame
node 1	1 frame	10 frames	10 frames	6 frames	1 frame
node 2	6 frames	30 frames	30 frames	12 frames	6 frames
node 3	N/A	N/A	N/A	N/A	N/A
node 4	6 frames	30 frames	30 frames	12 frames	6 frames
node 5	1 frame	10 frames	10 frames	6 frames	1 frame
node 6	6 frames	30 frames	30 frames	12 frames	6 frames
node 7	1 frame	10 frames	10 frames	6 frames	1 frame

The transmission in all nodes is unicast with predetermined target addresses (i.e., routes are known in advance), and the transmission range of all nodes is set as 15 meters by TinyOS. We have 5 source nodes, 2 aggregation nodes, and the base station in this experimental network. The topology of our experimental WSN and the coordination of its nodes are given in Figure 9. The experimental network is actually a typical data gathering tree mentioned in section 2. Also, through the topology information, we can observe that three groups of communication interference nodes, including the group 1 (nodes 0, 1, and 2), group 2 (nodes 2, 4, and 6), and group 3 (nodes 4, 5, and 7), since the distance between these nodes are within the transmission range (i.e., 15 meters).

5.2 Experimental Results

We generated five sample networks by utilizing different TinyOS applications on different nodes in the experimental network topology. DATA packets transmission intervals are set to different values in these applications to evaluate our static analysis approach under different communication traffic loads, which are shown in Table VI. The battery energy budgets for all nodes in the WSN are set to 200 and 400 Joules respectively. The major reason to provide the experimental results under different energy budgets is to validate the correctness and consistence of our static



Node	X	Y	Node Type	Node	X	Y	Node Type
0	0	0	source node	4	30	5	aggre. node
1	0	10	source node	5	40	0	source node
2	10	5	aggre. node	6	20	15	source node
3	20	5	base station	7	40	10	source node

Figure 9. The topology of experimental tree structure WSN and the coordination of all nodes.

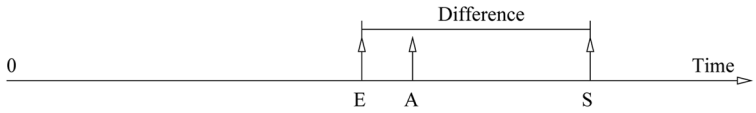


Figure 10. The relationship between static estimated and simulated lifetime (S: simulated lifetime, A: actual worst-case lifetime, E: estimated worst-case lifetime).

analysis method if the experimental results under different energy budgets are stable. Experimental results of both our static estimated and AVRORA simulated lifetime for each node are recorded in Tables VII and VIII. Also, the average life-time of all nodes in the network are recorded in these tables to indicate the lifetime of the WSN based on the average life-time strategy. It should be noted that the longest simulation time in our experiments is about 6.7 hours due to limited computer resources, while the realistic WSN normally has longer life-time. However, this simulation time is long enough to evaluate our work, since S-MAC frames just keep on repeating till the depletion of battery.

We can observe an obvious difference between our static estimated worst-case results and AVRORA simulated ones in Tables VII and VIII. Basically, WSN simulators are designed to simulate the average-case performance of each node in the network based on the average-case communication model, while our estimation provides a worst-case result. Again, the difference between the static estimation and simulation shows the importance to bound worst-case lifetime of the WSN. Although, our estimated worst-case lifetime may not always be achieved in the runtime of the network, it is definitely not safe for engineers to design their WSN based on simulated average case results. The relationship between our static estimated and simulated lifetime is shown in Figure 10.

Another observation is that the difference in samples 1 and 5 are significantly

Table VII. Lifetime of all samples with the 200 Joules energy budget (unit: second).

Energy 200 J	Sample 1			Sample 2			Sample 3			Sample 4			Sample 5		
	Sim	Est	Diff	Sim	Est	Diff	Sim	Est	Diff	Sim	Est	Diff	Sim	Est	Diff
node 0	11880.7	6626.6	44.22%	11462.4	9578.0	16.44%	12035.1	9584.5	20.36%	11979.9	9265.5	22.66%	11775.7	6626.6	43.73%
node 1	11860.4	6626.6	44.13%	11332.6	9578.0	15.48%	12036.0	9584.5	20.37%	12008.1	9265.5	22.84%	11663.6	6626.6	43.19%
node 2	10527.8	6770.1	35.69%	9826.1	8958.0	8.83%	10716.7	8969.6	16.30%	10514.1	8299.6	21.06%	10365.0	6708.6	35.28%
node 3	10588.5	8071.6	23.77%	10277.0	9505.1	7.51%	10038.0	9510.2	5.26%	9286.2	8896.6	4.20%	10423.7	8025.5	23.01%
node 4	9837.9	6770.1	31.18%	10128.0	8958.0	11.55%	10756.6	8969.6	16.61%	10131.3	8299.6	18.08%	10269.9	6708.6	34.68%
node 5	11916.6	6626.6	44.39%	12018.0	9578.0	20.30%	12012.1	9584.5	20.21%	11678.7	9265.5	20.66%	11503.4	6626.6	42.39%
node 6	11756.6	9265.5	21.19%	10710.7	9897.0	7.60%	10985.5	9898.3	9.90%	10898.4	9638.3	11.56%	11778.0	9248.8	21.47%
node 7	11986.7	6626.6	44.72%	12028.1	9578.0	20.37%	12010.9	9584.5	20.20%	11708.2	9265.5	20.86%	11630.3	6626.6	43.02%
Average	11294.4	7173.0	36.49%	10972.9	9453.8	13.84%	11323.9	9460.7	16.45%	11025.6	9024.5	18.15%	11176.2	7149.7	36.03%

Table VIII. Lifetime of all samples with the 400 Joules energy budget (unit: second).

Energy 400 J	Sample 1			Sample 2			Sample 3			Sample 4			Sample 5		
	Sim	Est	Diff	Sim	Est	Diff	Sim	Est	Diff	Sim	Est	Diff	Sim	Est	Diff
node 0	23999.2	13246.8	44.80%	23246.5	19152.2	17.61%	24024.4	19167.6	20.22%	23297.9	18531.0	20.46%	23762.8	13246.8	44.25%
node 1	23846.3	13246.8	44.45%	23225.4	19152.2	17.54%	24020.9	19167.6	20.20%	23181.2	18531.0	20.06%	23854.9	13246.8	44.47%
node 2	20009.9	13536.3	32.35%	20215.3	17917.4	11.37%	20994.7	17944.3	14.53%	19862.9	16594.1	16.46%	20484.4	13412.1	34.53%
node 3	20204.1	16136.8	20.13%	20027.3	19008.9	5.09%	19683.0	19020.3	3.37%	20198.5	17790.5	11.92%	21277.8	16048.4	24.58%
node 4	20663.1	13536.3	34.49%	20017.4	17917.4	10.49%	20591.9	17944.3	12.86%	20153.9	16594.1	17.66%	19680.5	13412.1	31.85%
node 5	23415.3	13246.8	43.43%	24072.8	19152.2	20.44%	22479.5	19167.6	14.73%	23383.8	18531.0	20.75%	23447.0	13246.8	43.50%
node 6	23271.5	18531.0	20.37%	20273.1	19792.7	2.37%	22341.3	19795.3	11.40%	21636.3	19275.2	10.91%	22976.7	18495.1	19.50%
node 7	23378.0	13246.8	43.34%	24070.9	19152.2	20.43%	22312.1	19167.6	14.09%	23401.9	18531.0	20.81%	23419.3	13246.8	43.44%
Average	22348.4	14341.0	35.83%	21893.6	18905.7	13.65%	22056.0	18921.8	14.21%	21889.6	18047.2	17.55%	22362.9	14294.4	36.08%

larger than other samples. Take 200 Joules energy budget as example, the average difference is 36.49% in sample 1 and 36.03% in sample 5, while the maximum average difference in other samples is 18.15% in Table VII. Especially for nodes 0, 1, 6 and 7 in the samples 1 and 5, their difference is above 40% for 200 Joules energy budget. DATA packet transmission intervals applied in samples 1 and 5 are smaller than other samples. For nodes 0, 1, 6 and 7 in these two samples, both the DATA packet transmission intervals are set to only one S-MAC frame, which is the heaviest communication traffic load for S-MAC based WSNs. All these DATA packets transmission are regarded to be successfully finished in our static analysis, which is the worst-case energy consumption scenario in theory. However, a lot of transmissions are blocked during the simulation based on the average-case communication model, because nodes are very difficult to obtain a free wireless medium due to the heavy traffic load. Then, the difference between the static estimated and simulated lifetime is larger.

Also, we observe that the simulated lifetime of samples 1 and 5 is not always

shorter than other sample networks, although they have shorter DATA packet transmission intervals. For instance, the average lifetime of 8 nodes in sample 1 is 11294.4 seconds, while it is 10972.9 seconds in sample 2 with 200 Joules energy budget from Table VII. Ordinarily, the more Tx frames occur, the more energy consumes, which will shorten lifetime of nodes. Actually, although the occurrence of Tx frames increases, the interference among neighboring nodes increases and thus nodes can not transmit in most Tx frames due to interference avoidance. Therefore, their lifetime is prolonged instead of shortened in the average-case communication model.

Last, the major components targeted in this paper are the CPU and radio. Consequently, the sources of overestimation can be categorized into two folds, including the CPU and radio energy consumption estimation, respectively. Table XI shows the worst-case energy consumption distribution between the CPU, radio, and other components (i.e., EEPROM and sensor) under 400 Joules energy budget. We can

Table IX. The comparison of experimental lifetime (unit: second) results between our static method and the all-on method with the 200 Joules budget.

Energy 200 J	Sample 1			Sample 2			Sample 3			Sample 4			Sample 5		
	All-On	Static	Diff	All-On	Static	Diff	All-On	Static	Diff	All-On	Static	Diff	All-On	Static	Diff
node 0	3399.9	6626.6	94.91%	3399.9	9578.0	181.71%	3399.9	9584.5	181.91%	3399.9	9265.5	172.52%	3399.9	6626.6	94.91%
node 1	3399.9	6626.6	94.91%	3399.9	9578.0	181.71%	3399.9	9584.5	181.91%	3399.9	9265.5	172.52%	3399.9	6626.6	94.91%
node 2	3399.9	6770.1	99.13%	3399.9	8958.0	163.48%	3399.9	8969.6	163.82%	3399.9	8299.6	144.11%	3399.9	6708.6	97.32%
node 3	3399.9	8071.6	137.41%	3399.9	9505.1	179.57%	3399.9	9510.2	179.72%	3399.9	8896.6	161.67%	3399.9	8025.5	136.05%
node 4	3399.9	6770.1	99.13%	3399.9	8958.0	163.48%	3399.9	8969.6	163.82%	3399.9	8299.6	144.11%	3399.9	6708.6	97.32%
node 5	3399.9	6626.6	94.91%	3399.9	9578.0	181.71%	3399.9	9584.5	181.91%	3399.9	9265.5	172.52%	3399.9	6626.6	94.91%
node 6	3399.9	9265.5	172.52%	3399.9	9897.0	191.10%	3399.9	9898.3	191.14%	3399.9	9638.3	183.49%	3399.9	9248.8	172.03%
node 7	3399.9	6626.6	94.91%	3399.9	9578.0	181.71%	3399.9	9584.5	181.91%	3399.9	9265.5	172.52%	3399.9	6626.6	94.91%
Average	3399.9	7173.0	110.98%	3399.9	9453.8	178.06%	3399.9	9460.7	178.26%	3399.9	9024.5	165.43%	3399.9	7149.7	110.29%

Table X. The comparison of experimental lifetime (unit: second) results between our static method and the all-on method with the 400 Joules budget.

Energy 400	Sample 1			Sample 2			Sample 3			Sample 4			Sample 5		
	All-On	Static	Diff	All-On	Static	Diff	All-On	Static	Diff	All-On	Static	Diff	All-On	Static	Diff
node 0	6799.7	13246.8	94.81%	6799.7	19152.2	181.66%	6799.7	19167.6	181.89%	6799.7	18531.0	172.53%	6799.7	13246.8	94.81%
node 1	6799.7	13246.8	94.81%	6799.7	19152.2	181.66%	6799.7	19167.6	181.89%	6799.7	18531.0	172.53%	6799.7	13246.8	94.81%
node 2	6799.7	13536.3	99.07%	6799.7	17917.4	163.50%	6799.7	17944.3	163.90%	6799.7	16594.1	144.04%	6799.7	13412.1	97.25%
node 3	6799.7	16136.8	137.32%	6799.7	19008.9	179.55%	6799.7	19020.3	179.72%	6799.7	17790.5	161.64%	6799.7	16048.4	136.02%
node 4	6799.7	13536.3	99.07%	6799.7	17917.4	163.50%	6799.7	17944.3	163.90%	6799.7	16594.1	144.04%	6799.7	13412.1	97.25%
node 5	6799.7	13246.8	94.81%	6799.7	19152.2	181.66%	6799.7	19167.6	181.89%	6799.7	18531.0	172.53%	6799.7	13246.8	94.81%
node 6	6799.7	18531.0	172.53%	6799.7	19792.7	191.08%	6799.7	19795.3	191.12%	6799.7	19275.2	183.47%	6799.7	18495.1	172.00%
node 7	6799.7	13246.8	94.81%	6799.7	19152.2	181.66%	6799.7	19167.6	181.89%	6799.7	18531.0	172.53%	6799.7	13246.8	94.81%
Average	6799.7	14341.0	110.91%	6799.7	18905.7	178.04%	6799.7	18921.8	178.27%	6799.7	18047.2	165.41%	6799.7	14294.4	110.22%

Table XI. The worst-case energy consumption distribution under 400 Joules budget (Unit: Joule).

Energy 400 J	Sample 1			Sample 2			Sample 3			Sample 4			Sample 5		
	CPU	Radio	Other	CPU	Radio	Other	CPU	Radio	Other	CPU	Radio	Other	CPU	Radio	Other
node 0	262.3	109.8	27.9	238.2	121.5	40.3	238.0	121.6	40.4	240.3	120.7	39.0	262.3	109.8	27.9
node 1	262.3	109.8	27.9	238.2	121.5	40.3	238.0	121.6	40.4	240.3	120.7	39.0	262.3	109.8	27.9
node 2	267.2	104.3	28.5	242.4	119.9	37.7	242.3	120.0	37.7	249.8	115.2	35.0	268.3	103.6	28.1
node 3	253.6	112.4	34.0	236.1	123.8	40.1	236.1	123.9	40.0	243.6	118.9	37.5	254.2	112.0	33.8
node 4	267.2	104.3	28.5	242.4	119.9	37.7	242.3	120.0	37.7	249.8	115.2	35.0	268.2	103.6	28.2
node 5	262.3	109.8	27.9	238.2	121.5	40.3	238.0	121.6	40.4	240.3	120.7	39.0	262.3	109.8	27.9
node 6	240.2	120.7	39.1	235.1	123.2	41.7	235.0	123.3	41.7	237.0	122.4	40.6	240.6	120.4	39.0
node 7	262.3	109.8	27.9	238.2	121.5	40.3	238.0	121.6	40.4	240.3	120.7	39.0	262.3	109.8	27.9
Average	259.7	110.1	30.2	238.6	121.6	39.8	238.5	121.7	39.8	242.7	119.3	38.0	260.1	109.9	30.1

observe that the CPU energy consumption is the largest part. The reason is that the energy efficient MAC protocol (i.e., S-MAC) periodically shut down the radio, which greatly saves the energy. Thus, to improve the accuracy of our worst-case energy consumption estimation in the future, we need to put the first priority on improving the accuracy of worst-case CPU energy consumption, which can be achieved through improving the WCET estimation. The current WCET path obtained from the ILP based method may be infeasible in the runtime. Therefore, we may get tighter WCET results through only considering the feasible WCET path in the runtime. More information about WCET estimation through feasible path techniques can be found in [Suhendra et al. 2006]. However, it is still very useful to bound tighter worst-case energy consumption scenarios through considering shortening the worst-case radio Rx duration in the worst-case scenarios, since we still have obvious radio energy consumption from the energy distribution.

Table XII. Comparison among Three Methods.

Methods	Static Analysis	Simulation	Model-Checking
Objective	Estimate the lifetime of WSN to guide sensor network design	Estimate the lifetime of WSN to guide sensor network design	Compare the worst-case life-time performance of routing protocols
Accuracy	Absolute value of the WSN life-time	Absolute value of the WSN life-time	Relative value of the WSN life-time
Result	Worst-case life-time estimation of WSN	Average-case life-time estimation of WSN	Worst-case life-time estimation of WSN
Scope	All components and operations	All components and operations	only radio broadcast and unicast operations
Efficiency	High	Medium	Low

5.3 Methods Comparison

To evaluate the improvement of our static method, we compare it with the most straightforward static analysis one called the *all-on method*. Basically, the all-on method will assume CPU remains active 100% of the lifetime, and the radio is always in the Tx state.

Tables IX and X show the difference between our estimated worst-case lifetime and that of the all-on method. This difference indicates the necessity of utilizing our method, since the all-on one is too pessimistic. Also, the improvement varies from 94.81% to 191.14%, which depends on the exact working load of each node. In other words, the heavier the working load, the closer our estimated result to that of the all-on method. Also, it means that the hardware resources cannot satisfy the worst-case requirement of WSNs if our estimated result is equal to the all-on method.

Up to now, three different types of approach are proposed to predict the lifetime of a WSN prior to deployment. Simulation is the most widely used method with the help of several kinds of WSN simulators. However, they can only estimate the average-case lifetime of a WSN. Model-checking was the first approach to estimating the worst-case lifetime of a WSN proposed by L. Mounier, et al. in [Mounier et al. 2007]. However, their goal is essentially to compare protocols, not to have an absolute value of the network lifetime [Mounier et al. 2007]. They focused on the relative worst-case lifetime derived from different routing protocols by only considering radio energy consumption of unicast and broadcast operations. Thus, results by their method cannot be used to direct the design of a WSN. Our proposed static analysis method, on the other hand, is the first to estimate the absolute value of worst-case network lifetime safely and efficiently. In addition, our method works on TinyOS which is a real OS of WSNs. Our estimation results can be directly used to guide the design of WSN. Detailed comparison of these three methods is illustrated in Table XII.

6. CONCLUSIONS

In this paper, we proposed a static analysis approach to estimating the worst-case lifetime of a WSN. This is a hybrid approach which utilizes the ILP and worst-case energy consumption scenarios analysis on S-MAC. Our evaluation revealed that the proposed approach provides safe and accurate worst-case lifetime estimation, as compared to AVRORA simulated results. Based on our worst-case lifetime estimation, a WSN can be designed safely and predictably.

In the future, we plan to work with several different MAC protocols, topologies and data collections schemes. Also, we plan to implement routing protocols in our estimation.

REFERENCES

AGNIHOTRI, S. AND NUGGEHALLI, P. 2007. Worst-case Interactive Communication and Enhancing Sensor Network Lifetime, In *Proceedings of Information Theory*, 2007. ISIT 2007. IEEE International Symposium, 2111-2115.

AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. Wireless Sensor Networks: a Survey, In *Journal of IEEE Communication Magazine*.

AL-KARAKI, J. AND KAMAL, A. E. 2004. Routing Techniques in Wireless Sensor Networks: A

Survey, In *Transaction of IEEE Wireless Communication.*

Atmega128L AVR Microprocessor Datasheet, ATMEL Inc.

AVR-GCC Homepage, http://winavr.sourceforge.net/index.html

AVRORA Homepage, http://compilers.cs.ucla.edu/avrora/

BARBERIS, A., BARBONI, L., AND VALLE, M. 2007. Evaluating Energy Consumption in Wireless Sensor Networks Applications, In *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, 455–462.

Berkeley Wireless Sensor Network Group Homepage, http://snap.cs.berkeley.edu/

CPLEX Homepage, http://www.ilog.com/products/cplex/

DIETRICH, I. AND DRESSLER, F. 2009. On the Lifetime of Wireless Sensor Networks, In *ACM Transactions on Sensor Networks (TOSN)*, Volume 5, Issue 1.

HEINZELMAN, W. ET AL. 2000. Energy-Efficient Communication Protocol for Wireless Micro-sensor Networks, In *Proceedings of the 33rd Hawaii International Conference on System Sciences.*

iBadge Homepage, http://nesl.ee.ucla.edu/projects/ibadge/

JAYASEELAN, R., MITRA, T., AND LI, X. 2006. Estimating the Worst-Case Energy Consumption of Embedded Software, In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, 81–90.

LANDSIEDEL, O., WEHRLE, K., TITZER, B., AND PALSBERG, J. 2005. Enabling detailed modeling and analysis of sensor networks, Praxis der Informationsverarbeitung und Kommunikation. Volume 28, Issue 2, 101–106.

LI, Y. S. AND MALIK, S. 1995. Performance Analysis of Embedded Software Using Implicit Path Enumeration, In *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, 456–461.

LU, G., KRISHNAMACHARI, B., AND RAGHAVENDRA, C. 2004. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks, In *Proceedings of 18th International Parallel and Distributed Processing Symposium.*

MCHENRY, T. AND HEIDEMANN, J. 2007. MAC Stability in Sensor Networks at High Network Densities, Technical Report ISI-TR-2007-628, USC/Information Sciences Institute.

MOUNIER, L., SAMPER, L., AND ZNAIDI, W. 2007. Worst-Case Lifetime Computation of a Wireless Sensor Network by Model-Checking, In *Proceedings of the 4th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks.*

MPR-MIB User Manual, Revision A, Grossbow Inc, Jun, 2007.

SALHIEH, A., WEINMANN, J., KOCHHAL, M., AND SCHWIEBERT, L. 2001. Power Efficient Topologies for Wireless Sensor Network, In *Proceedings of the International Conference on Parallel Processing.*

SHA, K. ET AL. 2008. Modeling the Lifetime of Wireless Sensor Network, In Proc. of *IEEE/ACS International Conference on Computer Systems and Applications.*

SUHENDRA, V. ET AL., Efficient Detection and Exploitation of Infeasible Paths for Software Timing Analysis, In *Proceedings of the 43rd IEEE Design Automation Conference.*

Teledyne Scientific & Imaging Inc. Homepage, http://www.rsc.rockwell.com/

TinyOS Homepage, http://www.tinyos.net/

TITZER, B. AND PALSBERG, J. 2005. Nonintrusive Precision Instrumentation of Microcontroller Software, In *Proceedings of LCTES'05, Conference on Languages, Compilers and Tools for Embedded Systems*, Chicago, Illinois.

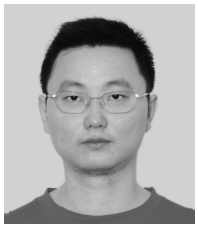
TITZER, B., LEE, D. K., AND PALSBERG, J. 2005. Avrora: Scalable Sensor Network Simulation with Precise Timing, In *Proceedings of IPSN'05*, Fourth International Conference on Information Processing in Sensor Networks, Los Angeles.

WANG, Q., HEMPSTEAD, M., AND YANG, W. 2006. A Realistic Power Consumption Model for Wireless Sensor Network Devices, In *Proceedings of the Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON).* Reston, VA, September.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An Energy-Efficient MAC Protocol for Wireless

Sensor Networks, In *Proceedings of 21st Annual Joint Conference of the IEEE Computer and Communications Societies.*

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2004. Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Network, In *Transaction of IEEE/ACM Transactions on Networking (TON), Volume 12, Issue 3,* 493–506.



Yu Liu is currently a PhD student in the Department of Electrical and Computer Engineering of Southern Illinois University Carbondale. He received his B.S and M.S degrees in Sichuan University, China in 2000 and 2003 respectively, and majored in communication engineering. Also, he worked in Motorola as senior software engineer from 2003 through 2007. His research interest includes real-time system, wireless sensor network, and cyber-physical system.

Wei Zhang is an associate professor in Electrical and Computer Engineering at Southern Illinois University Carbondale. He received the B.S. degree in computer science from the Peking University in China in 1997, the M.S from the Institute of Software, Chinese Academy of Sciences in 2000, and the Ph.D. degree in computer science and engineering from the Pennsylvania State University in 2003. His research interests are in embedded and real-time computing systems, computer architecture and compiler. Dr. Zhang has received the 2009 SIUC Excellence through Commitment Outstanding Scholar Award for the College of Engineering, and 2007 IBM Real-time Innovation Award. His research has been supported by NSF, IBM and Altera. He is a senior member of the IEEE. He has served as a member of the technical program committees for several IEEE/ACM conferences and workshops.

Kemal Akkaya received his BS and MS degrees in Computer Science from Bilkent University, Ankara, Turkey in 1997 and Middle East Technical University (METU), Ankara, Turkey in 1999 respectively. He worked as a software developer in 2000, Ankara, Turkey. He received his PhD in Computer Science at University of Maryland Baltimore County in 2005. Currently, he is an assistant professor in the Department of Computer Science at Southern Illinois University Carbondale. Dr. Akkaya is the Associate Editor of Elsevier Ad Hoc Networks Journal. He has served as the guest editor for Journal of High Speed Networks and in the TPC of many leading wireless networking conferences including IEEE ICC, Globecom, LCN and WCNC. His research interests include energy aware routing, security and quality of service issues in ad hoc wireless, sensor, mesh and underwater networks. Dr. Akkaya is a member of IEEE.