Simulation Blocks for TOSSIM-T2

Prabhakar T V, Venkatesh S, Sujay M S, Joy Kuri, Praveen Kumar
Centre for Electronics Design and Technology,
Indian Institute of Science,
Bangalore, India
(tvprabs, svenkat, mssujay, kuri, mpraveen)@cedt.iisc.ernet.in

Abstract—We develop several hardware and software simulation blocks for the TinyOS-2 (TOSSIM-T2) simulator. The choice of simulated hardware platform is the popular MICA2 mote. While the hardware simulation elements comprise of radio and external flash memory, the software blocks include an environment noise model, packet delivery model and an energy estimator block for the complete system. The hardware radio block uses the software environment noise model to sample the noise floor. The packet delivery model is built by establishing the SNR-PRR curve for the MICA2 system. The energy estimator block models energy consumption by Micro Controller Unit (MCU), Radio, LEDs, and external flash memory. Using the manufacturer's data sheets we provide an estimate of the energy consumed by the hardware during transmission, reception and also track several of the MCUs states with the associated energy consumption. To study the effectiveness of this work, we take a case study of a paper presented in [1]. We obtain three sets of results for energy consumption through mathematical analysis, simulation using the blocks built into PowerTossim-T2 and finally laboratory measurements. Since there is a significant match between these result sets, we propose our blocks for T2 community to effectively test their application energy requirements and node life times.

I. INTRODUCTION

TinyOS-2 (T2) is a second generation operating system which supports application development for resource constrained wireless sensor network hardware systems with greater platform flexibility. By redefining some of the basic TinyOS abstractions and policies such as initialization, task queue and power management, T2 has an improved system reliability and robustness. Service distributions in T2 introduce power management policies on top of the basic OS. It also simplifies the application development by introducing the separation between the low-level systems and applications.

Testing embedded applications without the physical hardware assists in speeding up and debugging user applications. Most of the sensor network applications rely upon energy efficiency from both their hardware as well as software to increase the node and network partitioning times. Therefore, end users would ideally prefer to test and energy estimate their applications close to reality even before field deployments. TOSSIM-T2 (TinyOS-2 SIMulator) is a discrete event simulator that can simulate the entire TinyOS applications. It implements the lower layer hardware by replacing the components with simulation implementations and the level at which these components are replaced is very flexible. TOSSIM-T2 issues events that can be hardware interrupts or high level system events and it can be extended to evaluate the total energy consumed by a node during its life time.

In this work, we present several simulation elements as plug-in blocks to the existing TOSSIM-T2. Since TOSSIM-T2 simulates at the chip level, we consider Mica2 mote and implement several elements of this hardware. We finally build a software block that we call the energy estimator block and this block evaluates the energy consumed by the Atmega128L microcontroller in several of its states. The block also captures the radio CC1000, LED’s and AT45db external flash memory. The rest of the paper is organized as follows. Section 2 describes the motivation in doing this work. Section 3 discusses the component modeling. In section 4, we describe the Energy estimation of each components. In section 5, we take a case study of an application and thus provide the results of energy estimator. Here, we also present the current consumption measurements of the case study using mica2 devices, digital oscilloscopes and current probes and finally compare the simulator results with the same. In section 7 we discuss the future work and conclusion.

II. MOTIVATION AND RELATED WORK

In [13] a fine grained and detailed energy evaluation of the hardware and associated application is available for the MICA2 platform. This tools considers the energy consumption of MCU instructions in addition to MCU expending energy in several of it’s states. We are further motivated by two previous pieces of work that are popular among the community of users for the earlier (1.x) version of TinyOS simulator called TOSSIM [10]. The power extensions [11](also called PowerTOSSIM) used several debug statements which are post processed to compute the energy consumed by a sensor device.

In this work, we start building the blocks towards PowerTOSSIM for T2. The attempt is to provide a proximal energy evaluation well within the TOSSIM-T2 simulator. In this way, users can seamlessly test their code within a single environment. However, we limit our MCU energy consumption to several operating modes it traverses during the lifetime of a sensor node.

III. COMPONENT MODELING

In this section we describe the modeling approach for some new plug-in components that were earlier not supported by the existing simulator. By doing so, we provide support for mica2 simulation for TOSSIM-T2.
Figure 1 shows various components within a mica2 mote. Each component requires a simulation implementation. An energy estimator component is built along with the other components to provide the energy expenditure due to MCU, LED’s, Radio and External Memory within the simulator.

A. Radio Component

The radio component takes away a significant share of the total battery power towards transmission, reception of routing and application data packets. Here, we first model the byte level radio for TOSSIM-T2 and then model the environment noise within the simulator[5] for CC1000 radio.

1) CC1000 Radio Modeling: In reality, CC1000 radio provides a hardware interrupt whenever there is a reception of a byte or on completion of transmitting a byte. All the configurations related to the radio such as transmission power, radio data rate, frequency of operation, receiver sensitivity are pre-configured to several internal hardware registers within the radio.

To simulate the functionality of the hardware registers, we create variables and provide access to them. During initialization phase of the radio, the configuration values are written to the radio register variables under TOSSIM-T2.

The communication between the radio and the microcontroller follows the Serial Peripheral Interface(SPI) protocol. Upon reception of a byte by the radio, it generates an interrupt(SPI Interrupt) signalling the microcontroller to start receiving the bytes. To simulate this in TOSSIM-T2, we use the concept of events which provides the SPI interrupt from the radio to the microcontroller. SPI events run as a loop throughout the simulation. Equation 1 gives the inter event time.

\[
SPI_{\text{event}} = \frac{\text{simulator_ticks}}{\text{radio_ticks}}
\]

where radio_ticks is given by equation 2 and by setting

\[
radio_{\text{ticks}} = \frac{(\text{No_bits})(\text{Mica2}_\text{Clk}_\text{Freq})}{\text{Radio}_{\text{Data Rate}}}
\]

No_bits=8, we simulate the radio at byte level. We choose Radio_Data_Rate as 19200 bps and Mica2_Clk_Freq as 7372800 Hz. Simulator_ticks is the granularity of the timer used in TOSSIM-2 simulator. These events trigger independently and are not synchronous for a pair of nodes.

The default state of the radio is reception state. This is to ensure that it can sense the medium for any incoming packets. Also, if the node wishes to transmit, being in a reception state helps to track any ongoing transmission and thus avoid collision. Whenever a node wishes to transmit, the node gains access to the running event loop of all neighboring nodes by applying a lock. It then issues an SPI event to the other nodes so that they receive the byte successfully. Once the transmission ends, the node releases the lock over the event loop. The introduction of this virtual channel access node solved the problem of packet collisions within the simulator.

2) Noise Modeling: The work in [5], details the methodology to be adapted in building a realistic noise model for CC2420 chipcon [3] radio that operates in the ISM 2.4GHz range. Since our radio CC1000 operates either in 433MHz or 915MHz ranges, we adapt this procedure in building the noise model with the exception of lack of interference from other radios operating in these frequencies. A small TinyOS application was written that measures the RF energy at 1 KHz by reading the ADC0 pin of the microcontroller. The application logs this data to the flash memory and another application reads this data off the mote. We found that the measured noise RSSI samples were in the range of -95 dBm to -105 dBm.

Figure 2(a) shows a 30 second period measured noise trace. The noise trace range from -95 dBm to -105 dBm over time. We assume that each noise sample is independent and build a cumulative distribution function(CDF) of these samples. The same noise distribution can be achieved by generating a uniformly distributed random numbers as inputs to the inverse CDF. The probability distribution function(PDF) of simulated data was nearly identical to the noise samples.

Figure 2(b) shows the PDF of the measured noise RSSI samples. Figure 3(a) and Figure 3(b) show the simulated RSSI values and its PDF respectively. We also made this simulated noise model available to the Media Access Layer(MAC) to
sample the environment noise for the purpose of floor noise estimation. Thus channel access by nodes is also made nearly realistic.

3) Packet Delivery Model: This model is used within the simulator to determine the link connectivity between a pair of nodes. In several experimental studies [7], [8], [9] the existence of three distinct regions in the wireless links namely; Connected, Transitional and Disconnected were identified. In [6] the authors analyze the different regions of low power wireless links by plotting the received Signal-to-Noise Ratio(SNR) values versus the Packet Reception Ratio(PRR). Using the same methodology, we experimentally obtain the desired SNR-PRR curves for -5 and +0 dBm power levels. Figure 4 shows the SNR-PRR curve for CC1000 radio. These results are integrated within TOSSIM-T2 for providing users a realistic link layer model; instead of using the idealized perfect-reception-within-range that essentially is a unit disc connectivity model.

B. External Flash Component

In some of the sensor network applications, we use the external memory for logging the sensor readings. In Mica2, the external memory is provided by Atmel’s AT45DB chip. This device is controlled by the commands passed by the host processor. The flash memory chip enables both block and page mode. We can read/write/erase a page or a block by passing suitable commands. We simulate the hardware registers of the flash similar to the radio. In a way we provide the simulation support to the flash memory but we don’t simulate the actual memory contents.

IV. ENERGY ESTIMATIONS

A. MCU Component

Energy expended by the MCU can be divided into two major categories; the MCU state and MCU execution. The former indicates the state in which the MCU is operating. While we do not evaluate the MCU execution energy, energy consumption in several of the MCU states is computed with the help of the current consumption available in the manufacturer data sheet and the time spent in these states during the simulation time.

There are six MCU states for Atmega128L microcontroller [2]. Table 1 shows the different states of the MCU and also gives the current consumption for each state. The MCU state is determined by polling three bits of MCUCR[2] hardware register. A change in MCU state is reflected in the MCUCR register. During state transitions, we compute the time spent in the previous state. All values are logged to the energy estimator block. Based on the current consumption for each state as mentioned in Table I, the energy estimator will compute the energy spent in that state.

<table>
<thead>
<tr>
<th>CPU Power Mode</th>
<th>Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>3.3</td>
</tr>
<tr>
<td>ADC Noise Reduction</td>
<td>1.0</td>
</tr>
<tr>
<td>Power Down</td>
<td>0.116</td>
</tr>
<tr>
<td>Power Save</td>
<td>0.124</td>
</tr>
<tr>
<td>Power Standby</td>
<td>0.237</td>
</tr>
<tr>
<td>Extended Standby</td>
<td>0.243</td>
</tr>
</tbody>
</table>

B. Radio Component

The current consumption by the radio device depends on several factors. For instance, transmit power, MAC level duty cycle, radio data rate are all contributing factors. Also, a certain energy is spent while powering up and running the crystal oscillator within the radio. Broadly a radio can exist either in Transmit (Tx) or Receive (Rx) state. As seen from Table II, though the reception current is constant, the transmission current varies by the transmit power level. For 433 MHz frequency, the chip supports up to 255 power levels ranging from +10dBm to -20dBm. Each power level has its own current consumption and the total transmission energy is found by evaluating the total time spent in the Tx state and its transmit power level. Similarly, the reception energy can be found by keeping track of the time spent and its current consumption in the Rx state. Table II shows the various states...
TABLE II
RADIO POWER STATE AND THEIR CURRENT CONSUMPTION

<table>
<thead>
<tr>
<th>Radio Mode</th>
<th>Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crystal Oscillator</td>
<td>0.105</td>
</tr>
<tr>
<td>Bias</td>
<td>0.755</td>
</tr>
<tr>
<td>Synthesizer $R_x/T_x$</td>
<td>3.140/4.140</td>
</tr>
<tr>
<td>Power Down</td>
<td>0.001</td>
</tr>
<tr>
<td>Receive</td>
<td>7.4</td>
</tr>
<tr>
<td>Transmit (0 dBm)</td>
<td>10.4</td>
</tr>
<tr>
<td>Transmit (+10 dBm)</td>
<td>26.7</td>
</tr>
</tbody>
</table>

TABLE III
EXTERNAL FLASH MODES AND ITS CURRENT CONSUMPTION

<table>
<thead>
<tr>
<th>Flash Mode</th>
<th>Current (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standby</td>
<td>0.002</td>
</tr>
<tr>
<td>Active Read</td>
<td>4</td>
</tr>
<tr>
<td>Active Program/Erase</td>
<td>10</td>
</tr>
</tbody>
</table>

that the radio can have and also the ideal current consumption for each state that are tabulated using CC1000 datasheet[3].

C. LED Component

LED’s form the visual indicators for the sensor device. If LED toggling code is intelligently embedded within the application code, they can play an important role in indicating bugs and code exceptions that occur during the development phase. Thus, they are mainly used for debugging purposes. The current consumed by the LED is around 2.2 mA. Since some of the MCU port pins are dedicated to the LEDs, we implement a pin state monitor. Such a monitor keeps track of low-high or high-low transitions of the MCU pins. For LED’s, we compute the time spent in switching on the LED component and the transition time from on to off state is logged to the energy estimator to compute the energy.

D. External Flash Memory

For flash memories, the write operation consumes more energy compared to the read operation. Table III shows the different modes of the flash and also the typical current consumed in each mode. Whenever there is read or write operation, the number of bytes($N$) read or written is logged to the energy estimator. The time taken to read or write is specified by the AT45DB datasheet[2]. The Energy estimator uses this to compute the energy spent by the flash for reading and writing the $N$ number of bytes.

V. ENERGY ESTIMATION - A CASE STUDY

[1] presents a routing strategy where a network’s partitioning time gets extended when sources choose forwarding relays specific to a network using a one-by-one, common-last approach. This approach utilizes common relays between two networks at the end; after all other relays specific to a network exhaust their life time. The analytical results show that network partitioning time is greater than the common-first approach; where common relays are chosen first for data forwarding. We implement the scheme with a TinyOS application and perform two evaluations: (a) compare the energy values shown by our energy estimator simulator block (i.e., PowerTossim-T2) with the analytically computed life time calculations and (b) compare energy measurements carried out with a current probe and oscilloscope with the energy estimated by the PowerTossim-T2 results.

A. Implementation

In [1], the energy expenditure is computed only for data packets both during their reception and transmission. In our implementation, we introduce an application level energy storage threshold block. Energy is continuously incremented up to a threshold value in this storage block only when a relay node either receives ($R_x$) or transmits ($T_x$) a data packet. Let this pre-configured node energy threshold be assigned ($E_0$). A relay node is made to shut down all activity soon after it reaches ($E_0$). This is the life time of the relay node.

Figure 5 shows the overall system software layers. The application layer comprises of application and energy storage threshold block. Each time a packet is transmitted/received it computes the energy spent and increments the system energy. The connectivity aware routing layer takes care of establishing the route based on weight and potential computation and uses the packet forwarding engine along these routes. We use BMAC [12] with 100% dutycycle.

B. Experimental setup

We consider both the common-last and common-first scenarios as in [1] and create a network topology to match the scenarios. Figure 6 shows the network topology for scenario 1. Nodes 1 to 5 are the relay nodes and thus have energy threshold block implementation. All other nodes (6 to 9) are ordinary nodes. In this case, when a relay node (say node 1) exhausts its energy, the source nodes switch to the next relay; following the one-by-one approach. The network life time will be equal to the life time of the last relay node in the network which is node 5. In [1] network life time is considered as the time at which the source nodes gets partitioned from the base station.
Fig. 6. An example network of scenario 1 considered in [1]

Fig. 7. An example network of scenario 2 considered in [1]

Figure 7 shows the network topology for scenario 2. Source node 6 forms a network $N_1$ and source nodes 7, 8 and 9 form a network $N_2$. Relay nodes 1, 2 and 3 form $S_1$ and relay nodes 3, 4 and 5 form $S_2$. Relay node 3 forms $S_{12}$. Here, one could again consider the common-last approach and additionally, the scenario is implemented for the common-first by both networks $N_1$ and $N_2$ choosing the common relay node (node 3). In case 1 (common-last), the source nodes in $N_1$ and $N_2$ avoid choosing $S_{12}$ first, whereas in case 2 (common-first), both $N_1$ and $N_2$ uses $S_{12}$ first. Here, network partitioning time is defined as the time at which either $N_1$ or $N_2$ gets partitioned and network life time is considered when both $N_1$ and $N_2$ gets partitioned from the base station.

C. Energy measurements

We have measured the current drawn from the power source to calculate the energy during data and routing beacon transmission. Figure 9 and 10 depict the same respectively. The figures clearly indicate higher peak current drawn during transmissions. The period and amplitude of peak current matches both the data packet size and the 14.8mA of +5dBm transmit power level respectively. The period of routing beacons is comparatively smaller due to a smaller packet size. Figure 11 shows the current drawn during data acknowledgment transmission and data forward to the base station by a relay node. Since we have set the MAC to a 100% duty cycle, most of the time the radio is in reception state.

While the data sheet of CC1000 specifies the current for a given transmit power, we observe from both figure 9 and 10, this specification is essentially the peak current that lasts for a small period of few microseconds. The average current recorded is indeed lower than this peak and is about 10mA. Thus, this value used for energy computation results in lower transmission energy compared to the PowerTossim-T2 results.

Table IV shows the comparison of life time values for all the three methods of evaluation. We have shown the results for a single node due to space constraint. The Mica2 mote’s energy consumption evaluation using PowerTossim-T2 matches the analytical results. Measurement based life time, although close enough, predicts a 3.6% more life time; possibly due to the average current consumption. The MCU total energy provided by the simulator is another useful result that may be hard to obtain from measurements. We also observe that the difference in energy computation for the total radio energy for PowerTossim-T2 and measurements is due to assumption of several peak current values. However, close matches in total received packet energy can be observed.

D. Results and Discussion

We compute the theoretical node life times from equation 3 for scenario 1 provided by [1].

$$T_i = \sum_{j=1}^{i} \frac{E_0(\gamma(E_r + E_t))^{j-1}}{r(\gamma E_r + (\gamma + 1)E_t)^j}$$

here $E_0$ is the threshold energy, $E_r$ and $E_t$ are the transmit and receive bit energy respectively, $T_i$ is the time at which relay $i$ runs out of energy. We take $E_0 = 0.1625J$, $E_t = 1.625$ mJ, $E_r = 1.2625$ mJ, $r = 200$ bps, and $\gamma = 4$.

Figure 12 shows the life time of the relay nodes of scenario 1 shown in figure 6. Since node 5 is chosen at the end, it’s life time is higher compared to the other relay nodes. We see that the life time closely matches to that of analytical value.

<table>
<thead>
<tr>
<th>TABLE IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPARISON OF ENERGY RESULTS FOR NODE 1 IN JOULE</td>
</tr>
<tr>
<td>PowerTossim-T2</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>MCU Total</td>
</tr>
<tr>
<td>Radio Total</td>
</tr>
<tr>
<td>Tx</td>
</tr>
<tr>
<td>Rx_Packet</td>
</tr>
<tr>
<td>Rx_Idle</td>
</tr>
<tr>
<td>Rx_Total</td>
</tr>
<tr>
<td>Life Time (s)</td>
</tr>
</tbody>
</table>
TABLE V

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Network Life time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common-first</td>
<td>600 sec</td>
</tr>
<tr>
<td>Common-last</td>
<td>660 sec</td>
</tr>
</tbody>
</table>

The network life time is the time at which relay node 5 runs out of energy i.e., 494 seconds.

While considering scenario 2, Table V shows the network life time for both case 1 and case 2. As we see, when $S_{12}$ is chosen last, we have an increased life time in the network.

Figure 8 in general, shows the energy distribution of a node during several states of it’s operation as per table I and II. The radio idle reception energy takes away maximum share due to 100% duty cycle of the MAC layer. Figure 8(a) shows the energy distribution of node 1 in scenario 1.

Figure 8(b) shows the energy distribution of node 3 in scenario 2 for case 1(Common-last). Since node 3 is chosen last for forwarding, it would have received less number...
of packets for forwarding and hence both the radio packet reception energy and radio transmit energy are very low.

In scenario 2, considering case 2(common-first), figure 8(c) shows the energy distribution for node 3's life time. In this case the node has more number of packets to receive and forward the same, since it is chosen first. Hence the energy consumption for transmission and reception are higher compared to figure 8 (b).

VI. CONCLUSION

In this work, we have implemented several hardware and software simulation blocks to provide a complete solution for Mica2 mote users. Users can evaluate the energy consumption for their applications using the energy estimator block in addition to and also predict node life times quite accurately by using the energy estimator block. The PowerTossim-T2 energy estimator gives detailed statistics about the energy expended by MCU in several of its states, Radio energy consumption due to transmission, reception, and idle state. The complete code for the Mica2 platform is available to end T2 code developers. The code is covered under GPL.

ACKNOWLEDGMENT

This work is supported by the Indo-Swiss joint project on Routing andAddressing in Sensor networks - A field theoretic approach (ROADSENS).

REFERENCES