Model-Based Design Exploration of Wireless Sensor Node Lifetimes

Deokwoo Jung, Thiago Teixeira, Andrew Barton-Sweeney
and Andreas Savvides

Embedded Networks and Applications Lab, ENALAB
Yale Univerisity, New Haven, CT 06520, USA
{firstname.lastname}@yale.edu

Abstract. This paper presents two lifetime models that describe two of the most common modes of operation of sensor nodes today, trigger-driven and duty-cycle driven. The models use a set of hardware parameters such as power consumption per task, state transition overheads, and communication cost to compute a node's average lifetime for a given event arrival rate. Through comparison of the two models and a case study from a real camera sensor node design we show how the models can be applied to drive architectural decisions, compute energy budgets and duty-cycles, and to preform side-by-side comparison of different platforms.

1 Introduction

The rapid progress of sensor networks in many applications is constantly fueling the quest for extending the lifetime of battery-operated wireless sensor nodes. In fact, many innovative platforms [9, 3, 8, 11, 13] have recently demonstrated several important new techniques for increasing node lifetime. Despite these efforts however, there are numerous situations where design decisions are rather opportunistic and tend to be influenced on the availability of low-power components and techniques without considering the longer term trends in platform design.

To complement these effort, we draw from our experiences in building and using sensor nodes to develop detailed models that characterize two widely used operation patterns for sensor nodes today: trigger-driven and schedule-driven. The models are constructed using Semi-Markov models by considering the power consumption in different operational modes and the energy overheads incurred during transitions. While similar predictions about lifetime could be obtained using simulations, we argue that detailed models are also needed to provide additional insight into how individual platform and application parameters affect lifetime. For instance, one can use the lifetime models presented here to evaluate potential gains from the design of hardware triggering mechanisms, software driven scheduling and duty-cycle modes and power budgets. The models presented here can also be used to perform side-by-side comparison between existing platforms under different application requirements, event arrival rates and detection probabilities. With this, our models can be used as a deployment analysis tool to determine which design is more appropriate for a certain application.

Our presentation is divided into two main parts. The first part states our assumptions and derives our models. The second part demonstrates the usefulness of our models in a case study drawn from our own experiences during the design of a camera sensor node. The case study shows how the models developed here can be applied to analyze the lifetime properties of a sensor node architecture based on application characteristics, hardware properties and changing trends in microprocessor and radio technologies.

2 Related Work

Node lifetime is a frequently discussed topic in platform design and analysis. In the last couple of years new platforms such as LEAP[9], XYZ[8], iMote2[3] and the Hitachi watch in [15] have demonstrated several new techniques for reducing power leakage during sleep time. The LEAP [9] platform adopted a dual processor/radio architecture to exploit the tradeoffs between power efficient and high-power components. An Energy Management and Accounting Preprocessor (EMAP) module based on a low-power MSP 430 processor has been designed to manage different power domains on the LEAP board, enabling the high-end sensors and processors only when needed. Intel's iMote2 [3] uses dynamic frequency and voltage scaling and a power management IC (PMIC) to control different voltage domains on the node. The XYZ [8] node and the Hitachi watch [15] have used an external real-time clock circuit to wake up the node processor from ultra-low power deep-sleep modes. A number of proposals [10],[13],[4] described energy dissipation at the node level. Nath et al.[10] used Markov chains to analyze energy dissipation behavior per node. Each node is assumed to have six distinct power modes and transitions over different modes with given probabilities. Despite the detailed power mode consideration, this work is mostly simulation-based (in ns-2) and does not consider the energy dissipation models pertaining to the power modes. Snyder et al. [13] demonstrated the validity and effectiveness of their power consumption simulation tool, PowerTOSSIM, by predicting energy consumption per node. Hardware components are characterized at a very detailed level to simulate power consumption of a node as close as possible. Another approach presented in [4] uses hybrid automata models for analyzing power consumption of a node at the operating system level (TinyOS).

Our work differs from the above in that it tries to derive longer-term models by considering a node's hardware characteristics and operation patterns. Instead of considering software optimizations, the emphasis of our analysis is in exposing how a chosen combination of hardware components and operation patterns can influence lifetime.

3 Model Overview and Assumptions

The analysis described below models two main sensing schemes commonly employed in sensor nodes today: trigger-driven and schedule-driven. In trigger-driven operation, the sensors are managed by a low-power pre-processing unit

Symbol	Description	Symbol	Description
E_{Total}	Total amount of energy per node	Z	Transmission time per packet
S_i	Power state of mode i	σ	Job inter arrival time per event
P_M	Power consumption at power mode M, where $M \in S_i, i = 1...5$	λ	Average event inter arrival rate
		$n_{ij}(t)$	# of $i \rightarrow j$ transitions during t
P_S	Power consumption at asleep period of schedule based node	P_W	Power consumption at awake period of schedule based node
p_j	Steady state probability of mode j	N_σ	The number of jobs per event
p_{ij}	Transition probability from mode i to j	C_{ij}	Transition energy cost from mode i to j
C_P	CPU wake-up energy cost	N_P	Number of packets per event
C_R	Radio wake-up energy cost	T_W	CPU awake duration
L	Channel-listening time of radio	T_S	CPU asleep duration
Y	Processing time per event	T_c	Duty period, $T_c = T_1 + T_2$
u	Detection probability	d	Duty cycle, $d = T_1/(T_1 + T_2)$

Table 1. List of variables

that continuously samples the sensors. This preprocessor performs a first-order filtering of the data and wakes up a more powerful main processing unit if certain criteria are met. The LEAP node [9] and image sensors described in [14] follow this model. In schedule-driven operation, the node's sensors are connected directly to the node's main processor. To conserve energy, the processor follows a schedule that alternates between a low-power mode (e.g sleep, deep-sleep or shutdown) and a short, full-power mode in which the processor (or its ADC) samples the sensors for interesting activity. If the desired event types are sensed, it proceeds to make the necessary computations and transmits the outcome with the radio if needed. The sentry nodes used in the Vigilnet project [5] follow this type of model. In this case the sentry are asleep most of the time, and periodically wake up to sample for activity.

3.1 Assumptions

The models described in this paper make the following assumptions:

1. The first-order statistical characteristic (mean value) of all random quantities (events, processing time, etc) is known by observation and experiment from the Ergodic property.
2. Event arrivals follow a Poisson distribution.
3. Processing and radio-transmission times are independent and identically distributed (i.i.d.) with arbitrary distribution.
4. When an event is detected, the node processes it and sends the information to a base station (or another node) with probability α.
5. During the processing period, the CPU visits a limited number of low-power states (e.g. idle state).
6. During the communication period, the radio visits a limited number of listen (idle) states.
7. All power consumptions are constant during an operation and a fixed amount of energy is required to turn on or off the CPU and radio.

	Trigger-Driven Node			Schedule-Based Node		
Mode	Preprocessor	CPU	Radio	Sensor	CPU	Radio
S_0	–	–	–	Off	Off	Off
S_1	On	Off	Off	–	–	–
S_2	On	On	Off	On	On	Off
S_3	On	On	TX	On	On	TX
S_4	On	Idle	Off	On	Idle	Off
S_5	On	On	RX	On	On	RX

Table 2. Power state description

The first three assumptions imply that the power state transitions may be modeled as a semi-Markov chain [12] that can be used to compute a node's average power consumption and lifetime. While assumption 2 may not always hold true in all deployments a Poisson arrival rate is a representative model for many applications. For example, the number of people entering a building is a well known example of Poisson arrival [6]. For the purposes of our analysis we argue that the Poisson assumption is a reasonable choice because our main interest is to exercise the node hardware parameters that influence lifetime. Furthermore, by fixing the distribution of arrival events in our models we provide a common baseline for the comparison of many platforms by exercising their features under the same underlying distribution. In order to include communication overhead in the lifetime analysis, the same communication paradigm is adopted for both the trigger-driven and schedule-driven models as stated in assumption 4. The next two assumptions, 5 and 6, related to the idle state of the CPU and listening state of the radio, are necessary to more accurately describe the power consumption of those components. When an event is sensed by the node, the CPU will usually go to a full-power, active mode to perform some processing or additional sensing, but may alternate it with a temporary lower-power state to conserve energy. This is accounted for in assumption 5. Meanwhile, it is common for MAC protocols to listen to the radio channel before any transmission, to avoid packet collisions [2]. For this we have introduced assumption 6. We also emphasize that our models focus on node-level behaviors by examining the parameters of the node hardware under different event arrival rates. Software and network level optimizations are therefore not considered in this analysis.

3.2 Node Power Modes and Variables

Our analysis considers a simplified version of the power modes available on sensor nodes, eliminating some of the impractical modes. The modes considered are described in Table 2. To develop our models we also introduce a set of variables. These are described Table 1. Our notation also uses a bar to denote expected value (i.e the expected value of the variable A is $\bar{A}$).

4 Lifetime Models

In this section, we will show that each sensor node can be modeled by an embedded semi-Markov Chain. Let $X(t)$ denote the power state at time t. Then

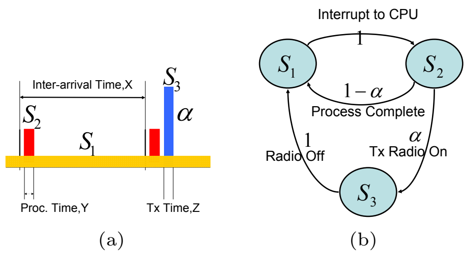


Fig. 1. (a) Power profile of simplified trigger-driven node model, (b) Semi-Markov chain of simplified trigger-driven node model.

change of state $X(t),\ t \geq 0$ does not solely depend on the present state, but also the length of time that has been spent in that state. This characterizes a semi-Markov chain, as states change in accordance with a Markov chain but there is a random length of time between the changes. Let H_i denote the distribution of time that the semi-Markov process spends in state i before making a transition, and let the mean be $\mu_i = \int_0^\infty x dH_i(x)$. With X_n denoting the nth state visit, $X_n, n \geq 0$ becomes a Markov chain with transition probabilities p_{ij}. It is also called the embedded Markov chain of the semi-Markov process [12]. Let T_{ii} denote the time between successive transitions into state i and let $\mu_{ii} = E[T_{ii}]$. If the semi-Markov process is irreducible and if T_{ii} has nonlattice distribution with finite mean, then

$$p_i \equiv \lim_{t\to\infty} P[X(t) = i | X(0) = j] = \lim_{t\to\infty} \frac{T_t}{t}\ , \tag{1}$$

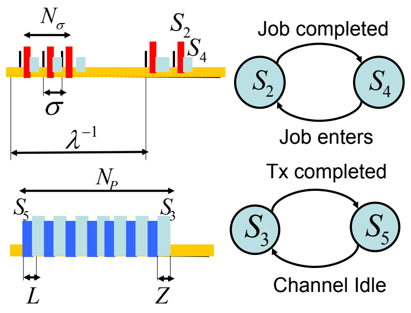
where T_t is the amount of time in i during $[0, t]$, exists and is independent of the initial state, j. In other words, p_i equals the long-run proportion of time in state i (the time spent in i over the combined time spent in all states). Suppose further that the embedded-Markov chain $X_n,\ n \geq 0$ is positive recurrent. Then a stationary probability exists, which is the frequency of visiting each state for infinite time duration . Let its stationary probability be $\pi_j,\ j \geq 0$. Then π_j is the unique solution of

$$\pi_j = \sum_i \pi_i p_{ij}, \sum_j \pi_j = 1 \tag{2}$$

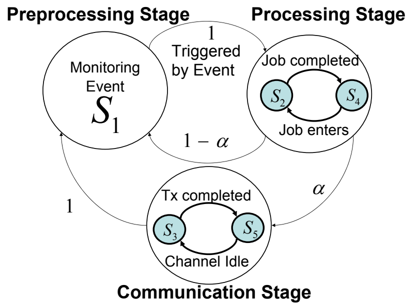
and π_j can be interpreted as the proportion of transitions into state j (over the sum of all state transitions). Then the following theorem holds

$$p_i = \frac{\mu_i}{\mu_{ii}} = \frac{\pi_i \mu_i}{\sum_j \pi_j \mu_j} \tag{3}$$

Using equations (2) and (3), one can compute the long-run proportion of time in state i.



(a)



(b)

Fig. 2. (a) Power profile of complete trigger-driven node model, (b) Semi-Markov chain of complete trigger-triven node model.

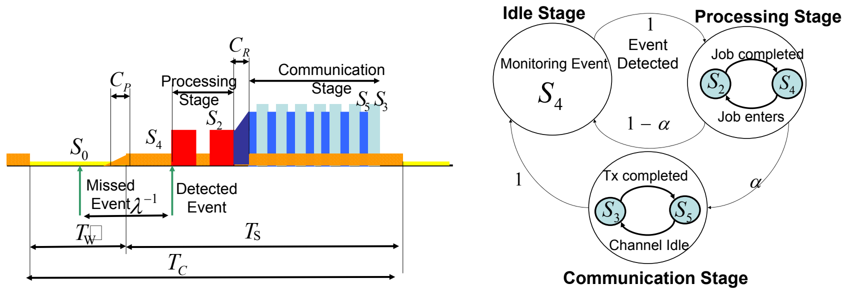
4.1 Trigger-Driven Lifetime Model

Figure 1a shows the simplest power model. In this model, the sensor node has only three states, which can be represented by the semi-Markov chain in Figure 1b. This model does not account for any Idle or Listening modes on the CPU or radio, respectively. However, in reality the CPU and radio often enter Idle mode during the processing and communication stages. For example, when an event is detected, the CPU has a choice of either processing the data, or deciding to drop it (or quickly store it for later). In these situations, the CPU may go back into an idle state to wait for the next job. Meanwhile, radios tend to spend considerable energy listening to the channel before any actual transmission due to impositions of the underlying MAC protocol. In IEEE 802.15.4, for instance, less than 50 percent of energy is spent for actual transmission, and listening activity accounts for more than 40 percent of energy consumption [2]. To take these factors into account, Figure 2a deals with the addition of the idle and listening states of the CPU and radio. The updated semi-Markov chain in Figure 2b shows that each processing and communication stage contains a two-state embedded chain.

Given a long enough time period, T, the total time spent at state i can be approximated as $\lim_{T\to\infty} T_i = Tp_i$. Therefore, the total energy spent at state i is $E_{S_i} = Tp_i \times P_{S_i}$, for $i \in \{1,2,3\}$, and the transition energy cost from state i to j during T can be obtained as $E_{S_{ij}} = C_{ij}\overline{n}_{ij}(T)$. However, only the CPU and radio wake-up costs (C_P and C_R, or, in $E_{S_{ij}}$ notation, $E_{S_{12}}$ and $E_{S_{23}}$) need to be taken into consideration since the sleep cost ($E_{S_{31}}$) is negligible in comparison. Since the total amount of energy spent at each state, E_{S_i}, and the transition energy, $E_{S_{ij}}$, cannot exceed the energy resource, E_{total}, the following inequality holds:

$$\sum_{1\le k\le 3} E_{S_k} + E_{S_{12}} + E_{S_{23}} \le E_{total} \tag{4}$$

By applying (2) and (3) to Figure 1b, we can obtain the asymptotic node lifetime as follows. A more detailed derivation can be found in [7].



(a) (b)

Fig. 3. (a) Schedule-based node power profile, (b) Power state transition during wake-period.

$$\boxed{T_L(\lambda) \le \frac{[1+\lambda K_T]E_{Total}}{P_{S_1}+\lambda K_E}} \tag{5}$$

In (5), K_T and K_E represent the average time and energy spent for a sensed event respectively. Typically, $K_E \gg P_{S_1}$ and $\lambda \ll 1sec^{-1}$. As shown in the denominator of (5), the power component can be roughly broken down into two parts: λK_E, the average power spent for computation and communication per sensed event; and P_{S_1}, the power spent to monitor the events. It can be easily found that a sensor node spends more power monitoring an event than processing it at $\lambda \le \frac{P_{S_1}}{K_E}$. The average steady-state power consumption of the trigger-driven sensor node is simply given as:

$$\overline{P}_{ST,td}(\lambda) = \frac{P_{S_1}+\lambda K_E}{[1+\lambda K_T]} \tag{6}$$

For the simplest power model, Figure 1a, K_T and K_E are given as:

$$K_T = \overline{Y} + \alpha\overline{Z}, \quad K_E = \overline{Y}P_{S_2} + \alpha\overline{Z}P_{S_3} + \frac{C_P+\alpha C_R}{1+\alpha} \tag{7}$$

By taking into consideration the average power consumption and sojourn time in these two-state chains as shown Figure 2b, K_T and K_E are given as:

$$\begin{aligned} K_T &= (\overline{\sigma}+\overline{Y})\overline{N_\sigma} + \alpha(\overline{L}+\overline{Z})\bar{N}_P \\ K_E &= (\overline{\sigma}P_{S_4}+\overline{Y}P_{S_2})\overline{N_\sigma} + \alpha(\overline{L}P_{S_5}+\overline{Z}P_{S_3})\bar{N}_P + \frac{C_P+\alpha C_R}{1+\alpha} \end{aligned} \tag{8}$$

4.2 Schedule-Based Lifetime Model

Let k be the total number of duty cycles during the node's entire lifetime, and each ϵ_i the residual processing time after the ith awake state of the node. Additionally, let T_W be the length of time when the node is awake. Then the average node lifetime is obtained as following:

$$\sum_{0\le i\le k}((T_W+\epsilon_i)\bar{P}_{W,i}+(T_S-\epsilon_i)\bar{P}_{S,i}+C_P)\le E_{Total} \tag{9}$$

where $\bar{P}_{W,i}$ and $\bar{P}_{S,i}$ denote average power consumption of awake and asleep periods during cycle i respectively. Note that each $\bar{P}_{W,i}$ incorporates the power expenditure of four power states: S_2, S_3, S_4 and S_5. As for the power computation, the schedule-based node performs the same function as the trigger-driven node when an event occurs during the awake period (Figure 3). Therefore, given a long enough timespan, the average power consumption of the node during the active period ($\bar{P}_{W,i}$) can be approximated by replacing in (6) the preprocessing power with idle power, and setting $C_P = 0$. The result is shown below:

$$\lim_{i\longrightarrow\infty}\bar{P}_{W,i}=\overline{P}_W(\lambda)=\frac{P_{S_2}+\lambda K'_E}{1+\lambda K'_T} \tag{10}$$

In (10), K'_T and K'_E represent the average time and energy spent for a sensed event respectively during awake period. Typically, $K'_E > P_{S_2}$ and $\lambda \ll 1sec^{-1}$. As before, the nominator of (10) shows the power component during the awake period can be roughly broken down into two factors, namely $\lambda K'_E$, the average power spent for computation and communication per sensed event, and P_{S_2}, the static power spent during the awake period. It can be easily found that a sensor node spends more power for the idle state than processing events at $\lambda \le \frac{P_{S_2}}{K'_E}$.

For the awake period of the schedule-based power model, (Figure 1b) K'_T and K'_E are given as using (8):

$$\begin{aligned}K'_E &= (\overline{\sigma}P_{S_4}+\overline{Y}P_{S_2})\overline{N_\sigma}+\alpha(\overline{L}P_{S_5}+\overline{Z}P_{S_3})\bar{N}_P+\frac{\alpha C_R}{1+\alpha}\\ K'_T &= (\overline{\sigma}+\overline{Y})\overline{N_\sigma}+\alpha(\overline{L}+\overline{Z})\bar{N}_P\end{aligned} \tag{11}$$

Using the fact that $\bar{P}_{S,i}(t)$ is constant ($\bar{P}_{S,i}\equiv P_{S_0}$), the average node lifetime can be obtained by applying (10) to equation (9):

$$\boxed{\overline{T_L}\approx(T_W+T_S)k\le\frac{E_{Total}(T_W+T_S)}{[T_W\overline{P_W}(\lambda)+T_SP_{S_0}+(\overline{P_W}(\lambda)-P_{S_0})\overline{\epsilon}+C_P]} \quad (12)}$$

Since it is typically the case that $T_W \gg \epsilon$, the term $(\bar{P}_W(\lambda)-P_{S_0})\bar{\epsilon}$ can often be ignored as $T_W\bar{P}_W(\lambda)\gg\bar{\epsilon}\bar{P}_W(\lambda)\ge\bar{\epsilon}(\bar{P}_W(\lambda)-P_{S_0})$.

The model derivation is now complete. Before applying these models, we have verified their numerical correctness through simulation. The simulation iterates through each state visited for a certain time period summing up all the power overheads during the lifetime of the node. Due to space limitations, these results are omitted from this paper.

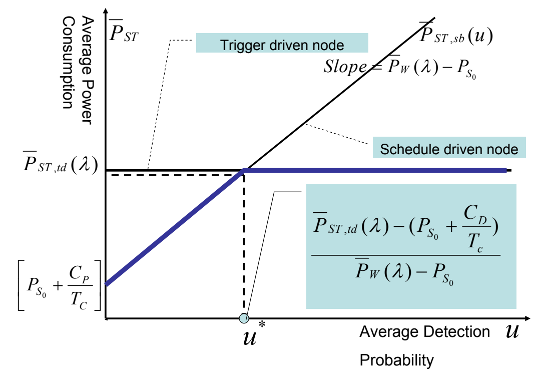


Fig. 4. Trade-Off Diagram: Power versus Average Detection Probability

4.3 Trigger-Driven and Schedule-Driven Comparison

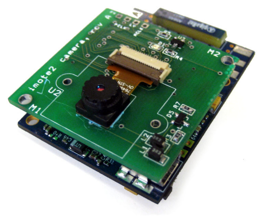
To meaningfully compare the two models, the event detection probability also needs to be considered. For the trigger-driven case, the sensor and preprocessor are always on, so we can assume that event detection happens with probability one. This comes at a price, of course, of added power cost for the preprocessor. The schedule-driven scheme, however, takes no such toll on power, but does so at the expense of event detection probability. All events that do not coincide with the node's duty-cycle remain undetected. To compare, let us define two random variables U and V to describe the number of Poisson sensor events during T_W and T_c respectively. Then the average detection probability, $E\left[\frac{U}{V}\right]$, can be computed as:

$$E\left[\frac{U}{V}\right] = \sum_{0 \leq v \leq \infty} E\left[\frac{U}{v}\middle| V = v\right] P_V(v) = \sum_{0 \leq v \leq \infty} \frac{1}{v}\left[v\frac{T_W}{T_c}\right] P_V(v) = \frac{T_W}{T_c} = d \tag{13}$$

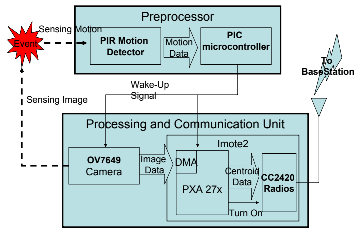
The second equality of Equation (13) comes from the fact that $P(U = u|V = v)$ has a binomial distribution, $B(v, d)$, where $d = \frac{T_W}{T_c}$. As shown in Equation (13), the detection probability is simply the duty cycle of the schedule-driven node. Therefore, we can express the trade-off diagram between the trigger-driven and schedule-driven schemes as a function of the detection probability u (shown in Figure 4), where the node lifetime of the schedule-based node follows the equation:

$$\bar{T}_L(u) \leq \frac{E_{Total}}{(\bar{P}_W(\lambda) - P_{S_0})u + (P_{S_0} + \frac{C_P}{T_c})} \tag{14}$$

From (14), the average steady-state power consumption of the schedule-based node can be found:



(a)



(b)

Fig. 5. (a) $iMote2$ node with a COTS camera board, (b) Trigger-driven sensor node with $iMote2$ fitted with a PIC microcontroller and PIR motion sensor

$$\bar{P}_{ST,sb}(u) = (\bar{P}_W(\lambda) - P_{S_0})u + \left(P_{S_0} + \frac{C_P}{T_c}\right) \tag{15}$$

By superimposing the two average power consumption formulas (Equations (15) and (6)), we can obtain a trade-off diagram as Figure 4. The thick line denotes the lowest-power choice for a given detection probability. The two curves meet at $u^* = \frac{\bar{P}_{ST,td}(\lambda)-(P_{S_0}+\frac{C_P}{T_c})}{\bar{P}_W(\lambda)-P_{S_0}}$. The figure shows that for an application that allows the use of sensors with detection probability smaller than u^*, the schedule-driven scheme is a sound choice. For events with larger arrival rates, u^* gets shifted to the right, further favoring the schedule-driven scheme for frequent, non-critical detections. Otherwise, if the application demands high-accuracy, the trigger-driven scheme is a better alternative. Of course, multiple nodes with complementary schedules may reduce the number of events that are globally missed, but such a network-wide power analysis is out of the scope of this paper.

5 Case Study: Using the models to characterize and make decisions about a camera sensor node

To demonstrate the usefulness of the models derived in the previous sections, we now demonstrate their application in the decision-making process of an experimental camera sensor node designed for the BehaviorScope project at Yale. Our goal is to decide whether it makes sense to develop an improved version of the camera node shown in Figure 5a. This camera node is an Intel $iMote2$ [3] coupled with a custom camera board we have designed with a commercial, off-the-shelf (COTS) image sensor, the Omnivision's OV7649. The node is powered by three AAA batteries (1150mAh capacity). The alternative design we are considering is a new camera board that supports a wakeup preprocessor mechanism comprised of a passive infrared (PIR) sensor for detecting motion and a small 8-bit PIC 10F200 microcontroller to act as a preprocessor. This configuration (described in Figure 5b) would allow the node to follow a trigger-driven mode of operation. Instead of periodically sampling the camera to detect activity, with this improvement the PXA 271 processor onboard the $iMote2$ will wait in a low-power

Parameter	Value	Parameter	Value
λ	0.1/min	$\bar{Y}$	$2sec^*$
$\bar{Z}$	$3.8msec$	$\bar{L}$	0 msec
$\bar{\sigma}$	0 min	$\bar{N}_\sigma$	1
α	1	T_c	10 min
$\bar{N}_P$	1	Total Energy	18.63 kJ

Table 3. Reference scenario, measured (∗)

state until triggered by the PIC-based preprocessor that is always kept on for a small energy overhead (Mode S_1 in Table 5). In this state, the preprocessor will apply a thresholding algorithm to the samples it collects from the PIR sensor. If the observed motion exceeds a predefined value, the preprocessor will power up the $iMote2$ and camera board to acquire and process the images. If the image processing reveals something of interest, the node the transmits the information to a basestation. These transmissions take place with probability α. To provide more concrete numbers in our case study, we set up the camera node to act as a simple single target localization device. An event is defined as the complete trajectory of human centroid in the range of camera sensor. In this setting, the camera sensor node performs the following functions in order.

1. When a person enters the camera's field-of-view, the preprocessor wakes up the iMote2 and camera (only for the trigger-driven node).
2. When awake, the iMote2 continually computes the location of the person at a frequency of 8Hz (8fps) until the person exits the coverage area of the camera.
3. Once the person is out of the sensing range, the node transitions back into the low power mode after sending a stream of locations to the base station.

According to our experiment, event duration is roughly 2 seconds. From our event definition, *processing time* is actually the same as *event duration*, and any incomplete trajectory (set of centroids) of a person is considered a *missed event* (hard-decision). For example, if a person is entering into camera view, and 1 sec later a node wakes up and observes only half of the trajectory, then the event is considered *missed*. For real-time computation, the node may transmit the centroids as soon as they are acquired. Notice, however, that whether the centroids are sent immediatelly or left for transmitting later is not of relevance to our models, as long as the energy consumption of both cases is still the same. From a lifetime perspective, the summation of the energy spent at each stage will be same regardless of the processing order. The time between capturing a frame to extracting a centroid is 123msec/centroid. Therefore, a node generates a total of roughly 16 centroids per event and the total amount of information per event is 96 Bytes. With a packet size of 119 Bytes (including 23 Byte of packet header) transmitted at the rate of 250 kbps, the packet transmission takes $3.8msec$. As a reference scenario, we set the system parameter values as specified in Table 3.

The PXA271 processor provides six power modes: Normal, Idle, Deep Idle, Standby, Sleep and Deep Sleep. Each of the six modes have different levels of

Mode	CPU PXA271	Camera OV7649	Radio CC2420	Total
S_0	Deep Sleep	Standby	Shutdown	
	$1.8mW$	$8mW$	$144nW$	$9.8mW$
C_P	$48.63mJ$	-	$691pJ$	$48.63\ mJ$
	$252msec$	-	$970\mu sec$	$253msec$
S_2	Normal	Active	Idle	
	$193mW$	$44mW$	$712\mu W$	$237.7mW$
S_4	Deep Idle	Active	Idle	
	$88mW$	$44mW$	$712\mu W$	$132.7mW$
C_R	-	-	$6.63\mu J$	$6.63\mu J$
	-	-	$194\mu sec$	$194\mu sec$
S_3	Normal	Active	TX	
	$193mW$	$44mW$	$78mW$	$315mW$
S_5	Normal	Active	RX	
	$193mW$	$44mW$	$78mW$	$315mW$

Table 4. Typical power-consumption specifications of schedule-driven camera sensor node ($iMote2$) at 104 MHz CPU Core frequency, 4MHz PIC and 0dBm TX Power

power consumption and different transition times to the Normal mode. The Normal mode is the state where all internal power domains and clocks are enabled and running. At Idle and Deep Idle modes, the CPU core stops being clocked, but for the latter the PXA is first switched into 13 MHz frequency. Standby mode puts all internal power domains into their lowest power mode except for the real-time clock and the PLL for the core. At Sleep and Deep Sleep modes, the PXA271 core power is turned off. Furthermore, in Deep Sleep mode all clock sources are also disabled. Therefore, Standby mode is the lowest power mode that does not require the node to reboot. To reason with the different design possibilities, we measured the power consumption and transient time of the $iMote2$ at different operational modes that correspond to the schedule-driven and trigger-driven modes we have previously defined in our models. The measurements for these modes are shown in Tables 4 and 5 [1]. More detailed information can be found in [1]. Both tables follow the power mode definitions introduced in Table 2. Since the $iMote2$ does not provide any special interface for measuring the power of the PXA CPU, we measured the total power drawn when the radio is shutdown.

Question 1: *What is the expected lifetime for the existing (schedule-driven, Figure 5a) and proposed (Figure 5b) configuration?* Using our measurements in Table 4, the schedule-driven node will last for only 1.61 days if it is always on, continuously sampling, since $\bar{T}_L(1) = \frac{E_{Total}}{\bar{P}_W(0.1/min)}$ by plugging $u = 1$ and $T_c = \infty$ in Equation (14). The lifetime of the alternative, trigger-driven configuration depends on the event arrival rate and can be computed using the model in

[1] In our tables Normal and Active modes have similar meanings. We opted on using two different terms to be consistent with the naming conventions of the datasheet for each device

	Preprocessor		CPU	Camera	Radio	
Mode	Motion Sensor	PIC10F200	PXA271	OV7649	CC2420	Total
S_1	On	On	Standby	Standby	Shutdown	
	$3.6\mu W$	$340\mu W$	$17mW$	$8mW$	$144nW$	$25.34mW$
C_P	-	-	2.2mJ	-	114nJ	2.2mJ
	-	-	11.432msec	-	970μsec	12.4msec
S_2	On	On	Normal	Active	Idle	
	$3.6\mu W$	$340\mu W$	$193mW$	$44mW$	$712\mu W$	$238.05mW$
S_4	On	On	Deep Idle	Active	Idle	
	$3.6\mu W$	$340\mu W$	$88mW$	$44mW$	$712\mu W$	$133.05mW$
C_R	-	-	-	-	$6.63\mu J$	$6.63\mu J$
	-	-	-	-	194μsec	194μsec
S_3	On	On	Normal	Active	TX	
	$3.6\mu W$	$340\mu W$	$193mW$	$44mW$	$78mW$	$315.34mW$
S_5	On	On	Normal	Active	RX	
	$3.6\mu W$	$340\mu W$	$193mW$	$44mW$	$78mW$	$315.34mW$

Table 5. Typical power-consumption specifications of trigger-driven camera sensor node($iMote2$) at 104 MHz CPU Core frequency, 4MHz PIC and 0dBm Tx Power on a CC2420 radio

Equation (8). The trend for different arrival rates is shown in Figure 6b. At our default configuration (PXA and Camera in Standby Mode), the trigger-driven iMote2 would only last 8.45 days at most (1.03 days at least). Figure 6b shows that less than 4 days of lifetime gain would be achieved by completely turning off the camera sensor board. It reveals the important design guide that in order to obtain a significant lifetime gain (more than 10 times), the trigger-driven node ultimately has to stay at Deep-Sleep mode during preprocessing stage, which is the lowest power state that can be achieved by the node with software control.

Question 2: *Given a specific arrival rate for a certain application, and a lifetime requirement, what is the maximum power a pre-processor(and sensor) can consume?* To obtain the power budget for the pre-processor we need to solve for P_{S_1} of the trigger-driven model in (5). The lifetime trend at different event arrival times as a function of preprocessor power is shown in Figure 6c.

Question 3: *If we don't build the proposed board and use a duty-cycle instead, what is the expected lifetime for a certain detection probability?* We can answer this question by plugging in the detection probability u in the lifetime model for the schedule-driven node described by Equation (14). The expected lifetimes for different detection probabilities are shown in Figure 6a.

Question 4: *Suppose we had an ideal sensor preprocessor (power cost=0) what would be the lifetime of the node at a certain arrival rate?* This trend is shown in Figure 6d. If we use Standby mode as the lowest power mode, in a trigger-driven configuration, the node will last for only 8.62 days! Also, if we entirely disable the preprocessor, the node will operate as in the schedule-driven model with duty-cyle=0, missing all events. Even so, the node lifetime is only 8.62 days, indicating that we should try to operate at power levels lower than the

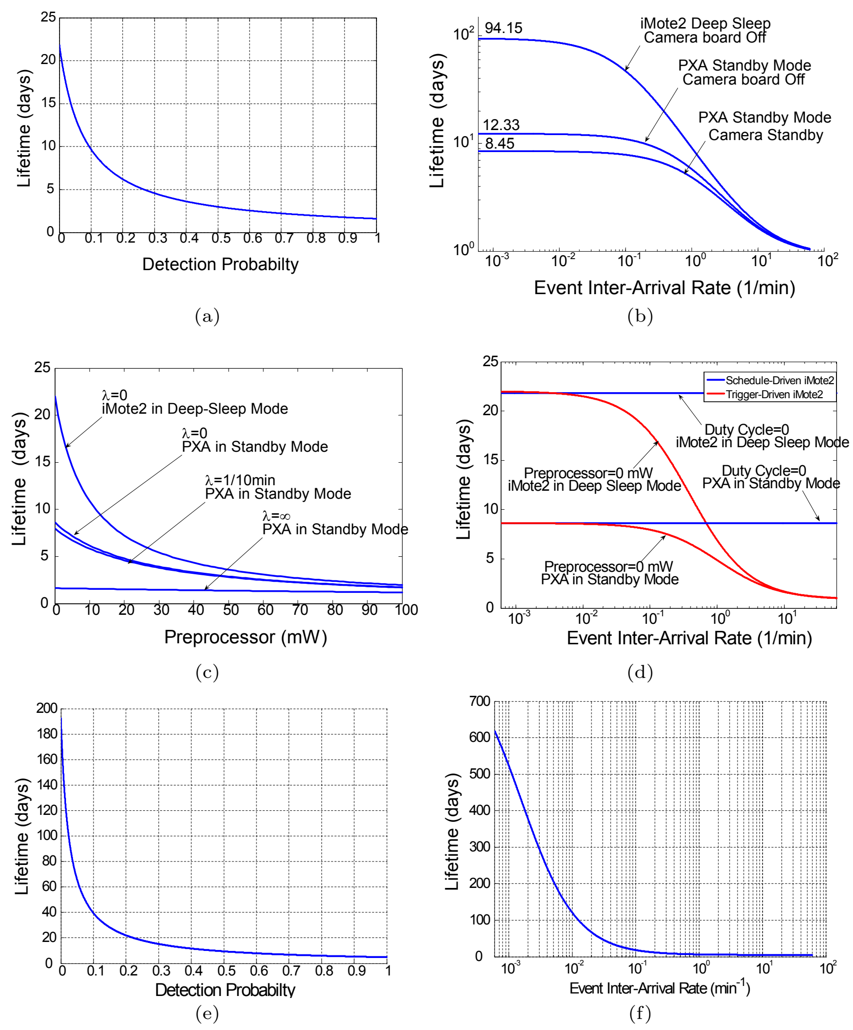


Fig. 6. a) Lifetime trend versus detection probability for question 1, b) Lifetime trend versus arrival rate for question 1, c) Lifetime trend for question 2, d) Lifetime trend for question 3, e) Lifetime trend versus detection probability for the sentry node described in VigilNet[5], f) Predicted hypothetical lifetime trend versus arrival rate for the sentry node described in VigilNet[5]

.

Standby mode. Comparing Figure 6c and Figure 6d, we notice that just lowering power consumption of preprocessor does not impact the lifetime trend of the trigger-driven node since P_{S_1} is heavily dominated by the power consumption of the camera board and PXA at Standby mode. Indeed, our computation shows that for rare events the lifetime increases to 94 days, with the camera board off and the iMote2 in Deep Sleep mode (Figure 6b). Much to our surprise, our models have shown that the addition of a preprocessor and trigger-driven operation will not provide substantial lifetime gains. This is mainly due to the high power consumption in the Standby mode of the PXA and camera. The trends also indicate that it is unlikely to significantly improve lifetime by manipulating the processor power modes alone. A better strategy would be to consider mechanisms that disconnect the entire node from the power supply as suggested in [8]. According to our models, the use of such a mechanism would increase the lifetime of the schedule-driven node to 552 days, a large improvement over the currently predicted 8.45 days for a non-ideal preprocessor (see Figure 6d).

As a sanity check, we also used our model to predict the lifetime of the Micaz nodes used in the Vigilnet project [5]. Using our model, we computed the expected lifetime of a sentry node to be about 442 hours (18.5) days as shown in Figure 6d. According to [5], a sentry node will last 90 days with a role rotation of 4-5 nodes and 25% of sentry duty cycle. Multiplying our prediction by 5 to account for role rotation, our model will anticipate a lifetime of 92.5 days, an estimate that is very close to the lifetime of the real deployment reported by the authors of [5]. Furthermore, Figure 6f shows that the lifetime of the sentry node will significantly increase if we convert it into a trigger-driven node using the same preprocessor as before.[2] Such a high lifetime gain comes from the fact that the power consumption of sentry node at Sleep state is extremely low (42 μW).

6 Conclusion

In this paper, we presented parametric lifetime model for trigger-driven node and schedule-driven node that also takes the associated transition overheads into consideration. The application of the models in making decisions about a camera node platform has helped us to isolate the dominant factors that limit lifetime in our design and provided valuable insight on how to proceed with the architecture. In the near future we are working on extending our models to cover more complex cases involving multiple processors and radios. Additional updates about this work can be found on our website at `http://www.eng.yale.edu/enalab`.

Acknowledgements

This work is funded in part by the NSF under contracts 0615226 and 0448082. The authors are also thankful to their collaborators Mani Srivastava (UCLA), Deepak Ganessan (UMASS Amherst), Mark Corner (UMASS Amherst), Prashant Shenoy (UMASS Amherst) and Lama Nachman (Intel) for the fruitful discussions around this work.

[2] This is a hypothetical lifetime since [5] does not consider a trigger-driven sentry node

References

1. A. Barton-Sweeney, D. Jung, and A. Savvides. imote2 node and ENALAB camera module power measurements. In *ENALAB Technical Report: 090601*, Sep 2006.
2. B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the IEEE 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives. In *Design, Automation, and Test in Europe (DATE), pp.196-201*, March 2005.
3. L. Nachman. Intel Corporation Research Santa Clara. CA. New tinyos platforms panel:iMote2. In *The Second International TinyOS Technology Exchange*, Feb 2005.
4. S. Coleri, M. Ergen, and T. Koo. Lifetime analysis of a sensor network with hybrid automata modeling. In *1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, September 2002.
5. T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. Abdelzaher. Achieving long-termsurveillance in vigilnet. In *Infocom 2006*, April 2006.
6. A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.
7. D. Jung, A. Barton-Sweeney, T. Teixeira, and A. Savvides. Model-based design exploration of wireless sensor node lifetimes. In *ENALAB Technical Report: 090602*, Sep 2006.
8. D. Lymberopoulos and A. Savvides. XYZ: A motion-enabled, power aware sensor node platform for distributed sensor network applications. In *Information Processing in Sensor Networks (IPSN), SPOTS track*, April 2005.
9. D. McIntire, K. Ho, B. Yip, A. Singh, W. Wu, and W. J. Kaiser. The low power energy aware processing (LEAP) embedded networked sensor system. In *Proceedings of Information Processing in Sensor Networks, IPSN/SPOTS*, April 2005.
10. R. A. F. Mini, M. V. Machado, A. A. F. Loureiro, and Badri Nath. Prediction-based energy map for wireless sensor networks. In *Elsevier Ad-hoc Networks Journal (special issue on Ad Hoc Networking for Pervasive Systems)*, March 2005.
11. J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *The Fourth International Conference on IPSN/SPOTS*, April 2005.
12. S. M. Ross. Stochastic processes, second edition, pp213-218. In *Jonn Wiley and Sons,Inc.*, April 1996.
13. V. Shnayder, M. Hempstead, B. Chen, G. Werner-Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *SenSys'04*, November 2004.
14. T. Teixeira, E. Culurciello, D. Lymberopoulos J. Park, A. Barton-Sweeney, and A. Savvides. Address-event imagers for sensor networks: Evaluation and programming. In *Proceedings of Information Processing in Sensor Networks (IPSN)*, April 2006.
15. S. Yamashita, S. Takanori, K. Aiki, K. Ara, Y. Ogata, I. Simokawa, T. Tanaka, K. Shimada, and Ltd.) H. Kuriyama (Hitachi. A 15x15mm, 1ua, reliable sensor-net module: Enabling application-specific nodes. In *IPSN SPOTS 2006*, April 2006.