Sensor Node Lifetime Analysis: Models and Tools

Deokwoo Jung, Thiago Teixeira, Andreas Savvides
Department of Electrical Engineering, Embedded Networks and Applications Lab
Yale University, New Haven, CT, 06520

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 perform side-by-side comparison of different platforms. Based on our models we present a MATLAB Wireless Sensor Node Platform Lifetime Prediction & Simulation Package (MATSNL). This demonstrates the use of the models using sample applications drawn from existing sensor node measurements.

Categories and Subject Descriptors: C.3 [Special-Purpose and Application-Based Systems]: Real-time and embedded systems

Additional Key Words and Phrases: node lifetime, semi-Markov Chain, Schedule-driven, Trigger driven, duty cycle, event arrival rate

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 [D.McIntire et al. 2005; L.Nachman 2005; D.Lymberopoulos and A.Savvides 2005; J.Polastre et al. 2005a; V.Shnayer et al. 2004] 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 by the availability of low-power components and techniques and do not provide a detailed understanding of the longer term trends in platform design.

To complement these efforts, we draw from our experiences in building and deploying sensor nodes to develop detailed models that characterize two widely used operation patterns for sensor nodes today: trigger-driven and schedule-driven operation. The models are constructed using Semi-Markov chains by considering the power consumption in different operational modes and the energy overheads incurred during inter-mode transitions. While similar predictions about lifetime could be obtained using simulations, we argue that a more rigorous model-driven exploration can yield additional insight into how individual hardware and appli-
cation 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, power budgets and duty-cycle modes. The same models can be applied to conduct side-by-side comparisons between existing platforms under different application requirements, event arrival rates and detection probabilities. This can help identify appropriate design matches for each application.

Our work is the follow-up effort of our previous conference paper [D.Jung et al. 2007]. In contrast to our conference paper, this paper provides a more in-depth presentation of the models and describes the implementation of a comprehensive MATLAB toolbox, MATSNL. With this toolbox, sensor node designers and users can make a model of the node to evaluate it with respect to other hardware configurations and application patterns.

Our presentation is divided into two main parts. The first part states our assumptions and derives our models which we validated thorough extensive simulation based on measured data. The second part demonstrates the usefulness of our models in a detailed case study drawn from our own experiences during the design of a camera sensor node. The case study shows how the models and the MATSNL tool can be applied to analyze the lifetime properties of a sensor node architecture and sensor network 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 few years new platforms such as XYZ [D.Lymberopoulos and A.Savvides 2005], LEAP [D.McIntire et al. 2005], iMote2 [L.Nachman 2005] and the Hitachi watch in [S.Yamashita et al. 2006] have demonstrated several new techniques for reducing power leakage during sleep time. The LEAP platform [D.McIntire et al. 2005] adopts a dual processor/radio architecture to exploit the tradeoffs between power efficient and low-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 [L.Nachman 2005] uses dynamic frequency and voltage scaling and a power management IC (PMIC) to control different voltage domains on the node. The XYZ node [D.Lymberopoulos and A.Savvides 2005] and the Hitachi watch [S.Yamashita et al. 2006] use an external real-time clock circuit to wake up the node processor from ultra-low power deep-sleep modes. A number of proposals [R.A.F.Mini et al. 2005], [V.Shnayder et al. 2004], [S.Coleri et al. 2002] describes energy dissipation at the node level. Nath et al. [R.A.F.Mini et al. 2005] 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. Shnayder et al. [V.Shnayder et al. 2004] demonstrated the validity and effectiveness of their power consumption simulation tool, PowerTOSSIM, by predicting energy con-
Sensor Node Lifetime Analysis: Models and Tools

3. MODEL OVERVIEW AND ASSUMPTIONS

The analysis in this paper models two main sensing schemes commonly employed in sensor nodes today: trigger-driven and schedule-driven. These are described in the block-diagrams in Figure 1 (a) and (b). In trigger-driven operation, the sensors are managed by a low-power pre-processing unit that continuously monitors the sensors.

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. This provides an upper bound on the lifetime a certain hardware configuration can achieve.

Fig. 1. (a) Schedule-Driven Architecture, (b) Trigger-Driven Architecture

In PowerTOSSIM, hardware components are characterized at a very detailed level to simulate power consumption of a node as close as possible. Another approach presented in [S.Coleri et al. 2002] uses hybrid automata models for analyzing power consumption of a node at the operating system level (TinyOS). VigilNet [P.Vicaire et al. 2006][T.He et al. 2006] applies schedule based energy management scheme in a real deployment for target tracking. In the VigilNet testbed, a network level and node level energy management approach was used to detect moving objects.

Our work in [D.Jung et al. 2007] is the first to derive, analyze and compare the two major node operation models of trigger-driven and schedule-driven nodes with an in-depth case study. In that paper, we proposed a Semi-Markov chain formulation of the two operational modes and showed the trade-offs associated with each one of them. The models aim at (i) quantifying the relationship between node lifetime and architectural features of the platform by utilizing measured hardware characteristics for preprocessing, processing and communication, (ii) investigating the dependence between node lifetime and characteristics of various workloads. The key observation in that work is that those two operational mechanisms show an explicit break-even point of performance depending on the hardware component characteristics and the event arrival rate.

The analysis in this paper models two main sensing schemes commonly employed in sensor nodes today: trigger-driven and schedule-driven. These are described in the block-diagrams in Figure 1 (a) and (b). In trigger-driven operation, the sensors are managed by a low-power pre-processing unit that continuously monitors the sensors.
This preprocessor performs a first-order filtering of the data and wakes up a more powerful main processing unit only if certain criteria are met. The LEAP node [D.McIntire et al. 2005] and Address-Event image sensors described in [T.Teixeira et al. 2006] 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 sensors are monitored for interesting activity. If the desired event types are sensed, the processor proceeds to make the necessary computations and transmits the outcome with the radio if needed. The sentry nodes used in the Vigilnet project [P.Vicaire et al. 2006] follow this model. The nodes 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 $\alpha$.

(5) During the processing period, the CPU visits a limited number of low-power states (e.g. idle state).
During the communication period, the radio visits a limited number of listen (idle) states.

Power consumption is constant during each operation and a fixed amount of energy is required to turn on or off the CPU or radio.

The first three assumptions imply that the power state transitions may be modeled as a semi-Markov chain [S.M.Ross 1996] 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 [Ihler et al. 2006]. 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 [B.Bougard et al. 2005]. For this reason 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. However, our model can be potentially extended to network level analysis.

3.2 Node Power Modes and Variables

Our analysis considers a simplified set of power modes available on sensor nodes, by eliminating some of the impractical modes (Table I). For example, the power mode where CPU is off but the radio is on is not used in practice since radios (slave) are generally controlled by the CPU (master). Figure 2 illustrates the power profiles among nodes without energy management, schedule based nodes and trigger driven nodes.
nodes. In this paper we consider the most aggressive energy management, which includes sleep control, radio control and data aggregation. To develop our models we also introduce a set of variables. Our notation 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 a semi-Markov chain. Typically a sensor node performs a predefined set of functions for each event, mainly: sensing, computation, and communication. Therefore, the power state transitions follow a certain pattern that is usually constant during the node lifetime unless an adaptive power control mechanism is applied. The amount of time spent at each power level is a random quantity that varies at each instance of the event and in many cases the mean and standard deviation are fixed. The distribution of the random time mainly depends on the event characteristics, i.e. event arrival time and event duration. In order to analyze the asymptotic pattern of energy dissipation at the sensor node, we must consider both the power level and its random duration simultaneously. We can describe the pattern by a semi-Markov model assuming Poisson event arrival. A semi-Markov process is one that changes states in accordance with a Markov chain but takes a random amount of time between changes.

Let $X(t)$ denote the power state at time $t$. Then a change from 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. Let $H_i$ denote the distribution of the time duration at 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 $n$th 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 [S.M.Ross 1996]. 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

Sensor Node Lifetime Analysis: Models and Tools

4.1 Trigger-Driven Lifetime Model

A trigger-driven attempts to reduce power consumption by keeping the main CPU and radio powered down until some interesting event takes place. In charge of sensing this interesting event and waking up the other components is the preprocessor. The preprocessor generally uses a simple hypothesis function to test the existence of potential events in its input. Once the hypothesis is true, the node wakes up the CPU to process information from a second, high-end sensor during the processing.

\[ p_i = \lim_{t \to \infty} P[X(t) = i | X(0) = j] = \lim_{t \to \infty} \frac{T_i}{t}, \]

where \( T_i \) 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 amount of time in state \( i \) (the overall time spent in \( i \) over the combined time spent in all states). Suppose further that the embedded-Markov chain \( X_n, n \in \mathbb{N} \) is positive recurrent. Then a stationary probability exists, which is the frequency of visiting each state for a given infinite time duration. Let its stationary probability be \( \pi_j, j \in \mathbb{N} \). Then \( \pi_j \) is the unique solution of

\[ \pi_j = \sum_i \pi_i p_{ij}, \sum_j \pi_j = 1 \]

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

\[ p_i = \frac{\mu_i}{\sum_j \pi_j \mu_j} \]

Equation (3) states that the long-run amount of time in state \( i \) can be obtained simply by computing ratio of average time duration in state \( i \) per cycle, \( \pi_i \mu_i \), to the average length of one cycle, \( \sum_j \pi_j \mu_j \). Using equations (2) and (3), one can compute the long-run proportion of time in state \( i \).
stage. Another hypothesis function tests the existence of events of interest in the high-end sensor data. If the processed information turns out to be unimportant, it is discarded and the node returns to the preprocessing mode. Otherwise, the data is stored in the memory, awaiting to be transmitted to the base station once a predefined transmission buffer is full.

Figure 3 shows the simplest model of a trigger-driven node. In this model, the sensor node has only three states: Preprocessing, Processing and Communication (Figure 3b). This simplified model does not account for any Idle or Listening modes on the CPU or radio, respectively. However, the reality is that CPUs and radios often alternate between modes during the processing and communication stages. Radios tend to spend considerable energy listening to the channel before 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, while listening activity accounts for more than 40 percent of energy consumption [B.Bougard et al. 2005]. The radio alternates between listening state and Tx state until all data in the transmission buffer is sent. Similarly, CPUs typically switch to idle mode between operations, resulting in a lower overall power profile. To take these factors into account, Figure 4a deals with the addition of the idle and listening states of the CPU and radio. The updated semi-Markov chain in Figure 4b shows that each processing and communication stage contains a two-state embedded chain.

For now, consider the simplest power model (Figure 3(b)). By applying (2) to the semi-Markov chain, the following equations are obtained:

\[
\sum_{1 \leq j \leq 3} \pi_j = 1, \; \pi \left( \begin{array}{ccc}
0 & 1 & 0 \\
1 & -\alpha & 0 \\
1 & 0 & 0
\end{array} \right) = \bar{\pi}
\]

where \( \bar{\pi} = [\pi_1 \; \pi_2 \; \pi_3] \). The solution of (4) gives \( \pi_1 = \pi_2 = \frac{1}{2+\alpha} \) and \( \pi_3 = \frac{\alpha}{2+\alpha} \). Therefore, the steady state probability of each state can be computed applying (3):

\[
p_1 = \frac{\bar{X}}{D_\alpha}, \; p_2 = \frac{\bar{Y}}{D_\alpha}, \; p_3 = \frac{\alpha \bar{Z}}{D_\alpha}
\]

where \( D_\alpha = \bar{X} + \bar{Y} + \alpha \bar{Z} \).

Given a long enough time period, \( T \), the total time spent at state \( i \) can be approximated as \( \lim_{T \to \infty} T_i = T p_i \). Therefore, the total energy spent at state \( i \) is \( E_{Si} = T p_i \times P_{Si} \), for \( i \in \{1,2,3\} \), and the transition energy cost from state \( i \) to \( j \) during \( T \) can be obtained as by multiplying the cost of a single \( i,j \) transition by the average number of \( i,j \) transitions. However, only the CPU and radio wake-up costs (\( C_P \) and \( C_R \), or, in \( E_{Si} \), notation, \( E_{S12} \) and \( E_{S23} \)) need to be taken into consideration since the sleep cost is negligible in comparison. Let’s define the number of cycles during \( T \) as \( k(T) \) (or the number of time the node wakes up during \( T \)). Furthermore, we define the transition energy cost at cycle \( n \) as \( C_n \) and the total transition cost during \( T \) as \( C_T \). Then \( C_T = \sum_{1 \leq n \leq k(T)} C_n \). The whole cycle can be partitioned into two groups, \( A_1 \) and \( A_2 \) where \( A_1 = \{n | X_j = S_1, X_{j+1} = S_2 \} \), \( A_2 = \{n | X_j = S_1, X_{j+1} = S_2, X_{j+2} = S_3 \} \). Then \( C_T = \sum_{n \in A_1} [C_P + C_R] + \sum_{n \in A_2} [C_P + C_R] \). Since \( \mu_{ii} = D_\alpha \), the average number of cycles during \( T \) is \( \frac{T}{D_\alpha} \). Where \( |A_1| \) and \( |A_2| \) are the cardinality of the sets \( A_1 \) and \( A_2 \).
respectively, and can be computed by:

\[
\lim_{T \to \infty} |A_1| = \frac{T(1 - \alpha)}{D_\alpha}, \quad \lim_{T \to \infty} |A_2| = \frac{T\alpha}{D_\alpha}
\]

Therefore, by substituting these into the equation for \(C_T\) we find

\[
C_T = \frac{T(C_P + \alpha C_R)}{D_\alpha}
\]

(6)

Since the total amount of energy spent at each state, \(E_{S_i}\), and the transition energy, \(C_T\), cannot exceed the energy resource, \(E_{total}\), the following inequality holds:

\[
\sum_{1 \leq k \leq 3} E_{S_k} + C_T \leq E_{total}
\]

(7)

By solving inequality (7), the asymptotic node lifetime follows:

\[
T_L(\lambda) \leq \frac{[1 + \lambda(\bar{Y} + \alpha \bar{Z})]E_{total}}{P_{S_1} + \lambda(Y P_{S_2} + \alpha Z P_{S_3} + C_D)}, \quad C_D = C_P + \alpha C_R
\]

(8)

where \(\bar{Y}\) and \(\bar{Z}\) are the average processing and transmission time per packet, respectively; \(\lambda\) is the event arrival rate; and \(P_{S_i}\) is the power consumption at state \(i\).

In the more realistic model in Figure 4 each processing and communication stage contains a two-state embedded chain. Therefore the variables \(\bar{Y}\) and \(\bar{Z}\) must be substituted by expressions that reflect this change. These new expressions are given below as \(\bar{T}_{proc}\) and \(\bar{T}_{comm}\):

\[
\bar{T}_{proc} = \frac{\bar{P}_{S_4} + \bar{Y} P_{S_2}}{\bar{\sigma} + \bar{Y}}, \quad \bar{T}_{proc} = (\bar{\sigma} + \bar{Y}) \bar{N}_\sigma
\]

(9)

\[
\bar{T}_{comm} = \frac{\bar{L} P_{S_5} + \bar{Z} P_{S_3}}{\bar{L} + \bar{Z}}, \quad \bar{T}_{comm} = (\bar{L} + \bar{Z}) \bar{N}_L
\]

(10)

where \(\bar{\sigma}\) is the average duration of the idle states of the CPU and \(\bar{L}\) is the average channel-listening (or wait) time of the radio. Moreover, \(\bar{N}_\sigma\) is the average number of idle states and \(\bar{N}_L\) is the average number of listening (or wait) states in each processing and communication stage, respectively.

The final equation for average node lifetime can be computed by applying (9) and (10) to (8), resulting in:

\[
\bar{T}_L(\lambda) \leq \frac{1 + \lambda[(\bar{\sigma} + \bar{Y}) \bar{N}_\sigma + \alpha(\bar{L} + \bar{Z}) \bar{N}_L]E_{total}}{P_{S_1} + \lambda[(\bar{\sigma} P_{S_4} + \bar{Y} P_{S_2}) \bar{N}_\sigma + \alpha(\bar{L} P_{S_5} + \bar{Z} P_{S_3}) \bar{N}_L + C_D]}
\]

(11)

The node lifetime, (8) and (11) and the corresponding average power consumption of the trigger-driven sensor node can be expressed more compact form as shown in (12).

\[
\bar{T}_L(\lambda) \leq \frac{(1 + \lambda K_T)E_{total}}{P_{S_1} + \lambda K_E}, \quad \bar{P}_{td}(\lambda) = \frac{P_{S_1} + \lambda K_E}{1 + \lambda K_T}
\]

(12)

In (12), $K_T$ and $K_E$ represent the average time and energy spent for a sensed event respectively. Typically, $K_E \gg P_{S1}$ and $\lambda \ll 1 \text{sec}^{-1}$. As shown in the denominator of (12), the power component can be roughly broken down into two parts: $\lambda K_E$, the average power spent for computation and communication per sensed event; and $P_{S1}$, 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 when $\lambda \leq \frac{P_{S1}}{K_E}$. For the simplest power model (Figure 3a), $K_T$ and $K_E$ are given as:

$$K_T = \bar{Y} + \alpha \bar{Z}, \quad K_E = \bar{Y} P_{S2} + \alpha \bar{Z} P_{S3} + C_P + \alpha C_R$$

(13)

By taking into consideration the embedded Markov chains shown in Figure 4b, $K_T$ and $K_E$ are given as:

$$K_E = (\bar{\sigma} P_{S4} + \bar{Y} P_{S2}) N_{\sigma} + \alpha (\bar{L} P_{S5} + \bar{Z} P_{S3}) N_{L} + C_P + \alpha C_R$$

$$K_T = (\bar{\sigma} + \bar{Y}) N_{\sigma} + \alpha (\bar{L} + \bar{Z}) N_{L}$$

(14)

### 4.2 Schedule-Driven Lifetime Model

Similarly to the trigger-driven case, schedule-driven operation can be modeled by a semi-Markov chain, which in the more realistic case is comprised of embedded chains. The schedule-driven node is driven by two timers, $\text{Timer}_{\text{wake}}$ and $\text{Timer}_{\text{sleep}}$, which dictate when the node is to be woken up or put to sleep. These timers have periods $T_w$ and $T_s$, respectively. If an event of interest is detected during the awake period, the node stops $\text{Timer}_{\text{sleep}}$ to process it. If the detected event needs to be transmitted to the base station, then the node turns on its radio and starts transmitting. Otherwise, the node sets $\text{Timer}_{\text{wake}} = T_s$ and sleeps until woken up by $\text{Timer}_{\text{wake}}$. It is important to note that the duty cycle length, $T_c$, is not always the same: it depends on whether an event was detected, and whether transmission is to take place. If no event is detected, $T_c = T_w + T_s$. Otherwise, $T_c$ may be smaller or larger than that quantity, depending on the event detection.
The corresponding power model and semi-Markov chain are shown in Figure 5. Compared to the trigger-driven case, the schedule-driven model has an additional parameter, \( \beta \), which is the probability of an event being detected during the awake period. Furthermore, we make the following assumptions:

1. The transition energy cost from CPU Idle state to Active state is negligible compared to the node wake-up energy;
2. The probability of more than one event arrival is negligible during \( T_w \);
3. Events have a finite constant duration, \( T_e \).

Assumption 1 generally holds for many CPUs and we assume there is no energy cost from idle mode to active mode. Assumption 2 holds for many schedule based sensing algorithms. For example, Vigilnet [P.Vicaire et al. 2006] uses a duty cycle of 25% with \( T_c = 1\text{sec} \) duty period to make the effective awake period \( T_w = 250\text{msec} \). Therefore, it is very unlikely that more than one event occurs during that time. For this reason, in the schedule-driven model the node transitions to sleep after an event is detected. The result is that the cycle in that case may be smaller than \( T_c = T_w + T_s \).

Assumption 3 depicts the situation where an event takes place in the sensing area for a finite amount of time. This assumption holds for many realistic applications where the sleep scheduling algorithm is often based on prior knowledge about the event duration. For example, in a target tracking application, an event may be defined as a person being present in the camera’s field-of-view. In that case, \( T_e = \text{The length of trajectory} \).

Let \( T_{idle} \) denote the time duration in the idle power mode, \( S_4 \), after wake-up. Then \( T_{idle} \) depends on event detection. Let \( t_e \) and \( t_w \) denote an event arrival time and node wake-up time. Then an event can be detected if and only if \( t_e \in [t_w - T_e, t_w + T_w] \). Otherwise, the event is not detected and the node will spend \( T_w \) in idle state (i.e. \( T_{idle} = T_w \)). In case of event detection, \( T_{idle} \) may take one of two values: \( T_{idle} = 0 \) when \( t_e \in [t_w - T_e, t_w] \), or \( T_{idle} = t_e - t_w \) when \( t_e \in [t_w, t_w + T_w] \).

By using assumption 2, \( T_{idle} \) can be computed as follows \(^1\).

\[
T_{idle} = \frac{E[t_e - T_e]}{t_w - T_e \leq t_e \leq t_w + T_w} = \int_{t_w - T_e}^{t_w + T_w} \frac{(t_e - T_e)}{T_w + T_e} dt_e = \frac{T_w^2}{2(T_w + T_e)}
\]

where the time reference is set to \( t_0 = t_w - T_e \).

Since the duration of the idle time in the awake period depends on whether or not an event has been detected, we split the \( S_4 \) power state into two virtual states: \( S_{4i} \), for when no event has been detected (always idle); \( S_{4e} \), for when a detection has occurred. By applying (2) to the semi-Markov chain, the following equations are obtained:

---

\(^1\) we know that in an interval \([t_1, t_2]\) only one arrival of a Poisson process has occurred. Then conditional on this knowledge, the time of this arrival is uniformly distributed in \([t_1, t_2]\)

\[
\sum_{j \in \{0, 4e, 4i, 2, 3\}} \pi_j = 1, \quad \pi = \left(\begin{array}{cccc}
0 & \beta & 1 - \beta & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
1 - \alpha & 0 & 0 & \alpha \\
1 & 0 & 0 & 0
\end{array}\right) = \bar{\pi}
\] (15)

where \(\bar{\pi} = [\pi_0 \, \pi_{4e} \, \pi_{4i} \, \pi_2 \, \pi_3]\). The solution of (4) gives \(\pi_0 = \frac{1}{2 + \beta + \beta \alpha}\), \(\pi_{4i} = \frac{1 - \beta}{2 + \beta + \beta \alpha}\), \(\pi_{2} = \pi_{4e} = \frac{\beta}{2 + \beta + \beta \alpha}\) and \(\pi_3 = \frac{\beta \alpha}{2 + \beta + \beta \alpha}\). Therefore, the steady state probability of each state can be computed applying (3):

\[
p_0 = \frac{T_s}{D_\beta}, \quad p_{4e} = \frac{\beta T_w}{D_\beta} \frac{T_w^2}{2(T_w + T_e)}, \quad p_{4i} = \frac{(1 - \beta)T_w}{D_\beta}, \quad p_2 = \frac{\beta \bar{Y}}{D_\beta}, \quad p_3 = \frac{\beta \alpha \bar{Z}}{D_\beta}
\] (16)

where \(D_\beta = T_s + T_w + \beta \bar{Y} + \beta \alpha \bar{Z} - \beta \frac{T_w(T_w + 2T_e)}{2(T_w + T_e)}\).

Much like the trigger-driven case on the previous section, the equations

\[
\lim_{T \to \infty} T_i = T p_i, \quad \sum_{k \in \{0, 4e, 4i, 2, 3\}} E_{S_k} + C_T \leq E_{total}
\]

are used to yield the asymptotic node lifetime:

\[
\bar{T}_L(\beta) \leq \frac{\left[ (T_s + T_w) + \beta (\bar{Y} + \alpha \bar{Z}) - \frac{T_w(T_w + 2T_e)}{2(T_w + T_e)} \right] E_{total}}{(T_s P_{S0} + T_w P_{S4} + C_P) + \beta \left[ \bar{Y} P_{S2} + \alpha (\bar{Z} P_{S3} + C_R) - \frac{T_w(T_w + 2T_e)}{2(T_w + T_e)} P_{S4} \right]}
\] (17)

The node lifetime (17) and the corresponding average power consumption of the schedule-driven sensor node can be expressed in a more compact form:

\[
\bar{T}_L(\beta) \leq \frac{(1 + \frac{\beta}{T_c} M_T) E_{Total}}{\bar{P}_e + \frac{\beta}{T_c} M_E}, \quad \bar{P}_{sch}(\beta) \geq \frac{\bar{P}_e + \frac{\beta}{T_c} M_E}{1 + \frac{\beta}{T_c} M_T}
\] (18)

where \(M_T\) and \(M_E\) are the average residual time and average residual energy spent given that an event is detected. \(\bar{P}_e\) is the average power consumption of a schedule based node given that no events are detected. Here, again, the power component can be broken down into two parts: \(\bar{P}_e\), the average power consumption given that no events are detected; and \(\bar{M}_E\), the average residual energy normalized by \(T_c\). We note that it is desirable to keep \(\bar{P}_e \gg \frac{\bar{M}_E}{T_c}\). Otherwise, higher wake-up frequency (and, consequently, higher detection probability) will rapidly compromise the node lifetime.

For the simpler power model in Figure 5a, \(\bar{P}_e\), \(M_T\) and \(M_E\) are given as:

\[ ME = \left( \bar{Y} P_{S_2} + \alpha(\bar{Z} P_{S_3} + C_R) - \frac{dT_c(dt_c + 2T_e)}{2(dt_c + T_e)} P_{S_4} \right) \]

\[ MT = \left( \bar{Y} + \alpha \bar{Z} - \frac{dT_c(dt_c + 2T_e)}{2(dt_c + T_e)} \right) \]

\[ \bar{P}_e = P_{S_0} + d(P_{S_4} - P_{S_0}) + \frac{C_P}{T_c} \]  

But this simpler model does not consider the state changes within the processing and communication stages. Taking these into account through the use of embedded chains (much like what was done in the trigger-driven model, in Figure 4b) the average energy consumption and sojourn time \( \bar{P}_e \), \( M_T \) and \( M_E \) can be computed as:

\[ ME = \left[ \tilde{N}_s(\bar{\sigma} P_{S_4} + \bar{Y} P_{S_2}) + \alpha(\bar{N}_L(\bar{L} P_{S_5} + \bar{Z} P_{S_3}) + C_R) - \frac{dT_c(dt_c + 2T_e)}{2(dt_c + T_e)} P_{S_4} \right] \]

\[ MT = \left( \tilde{N}_s(\bar{\sigma} + \bar{Y}) + \alpha \bar{N}_p(\bar{L} + \bar{Z}) - \frac{dT_c(dt_c + 2T_e)}{2(dt_c + T_e)} \right) \]

\[ \bar{P}_e = P_{S_0} + d(P_{S_4} - P_{S_0}) + \frac{C_P}{T_c} \]

5. 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 this comes at the expense of the event detection probability. In many cases [T.He et al. 2006; Q.Cao et al. 2005], the detection probability, \( u \), is a function of duty cycle \( (d) \), duty period \( (T_c) \), and event duration \( (T_e) \). In this section, we derive (18) as a function of event detection probability and compare its average power consumption to that of the trigger-driven model. We can, therefore, express the trade-off diagram between the trigger-driven and schedule-driven schemes as a function of the detection probability \( u \).

Let \( t_0 \) and \( t_1 \) denote the wake-up time of the schedule-driven node for two neighboring duty periods. Then the node will transition from \( S_0 \) to \( S_{4e} \) if and only if events are detected within \([t_0 - T_c, t_0 + T_w]\) (otherwise, it will transition to \( S_{4i} \)). Let \( N \) denote the number of new event arrivals during that time interval. By our Poisson event arrival assumption, \( N \sim \text{Poisson}(\lambda) \) and the probability that \( n \) events occur within an interval of length \( T_w + T_e \) is:

\[ Pr(N = n) = e^{-\lambda(T_w + T_e)} (\lambda(T_w + T_e))^n \]

Then the transition probability from state \( S_0 \) to \( S_{4e} \) is \( \beta = Pr(N \geq 1) \), and that of going from \( S_0 \) to \( S_{4i} \) must be \( Pr(N = 0) \), or \( 1 - \beta \). By Taylor expansion we...
derive following equations:

\[
Pr(N = 0) = e^{-\lambda(T_w + T_e)} = 1 - \lambda(T_w + T_e) + o(T_w + T_e)
\]

\[
Pr(N = 1) = \lambda(T_w + T_e) + o(T_w + T_e)
\]

\[
Pr(N \geq 2) = o(T_w + T_e)
\]

where \(o(T_w + T_e)\) describes the upper bound of the higher-order terms, such that 
\(\lim_{T_w \to 0} \frac{o(T_w + T_e)}{T_w} = 0\). Therefore, \(\beta\) can be approximated as follows by Assumption 2:

\[
b = 1 - e^{-\lambda(T_w + T_e)} \approx \lambda(T_w + T_e) \leq 1
\]  

An event is detected if its arrival time, \(t_e\), is within either \([t_0, t_0 + T_w]\) or \([t_1, t_1 - T_e]\) where \(t_1\) is the wake-up time of next duty cycle of the node. The length of this duty period, \(t_1 - t_0\), will be equal to \(T_e + \delta\) where \(\delta\) is the residual length of the active period and is a function of the event arrival time. That is, if an event is detected near the end of the awake period, \(\delta\) will be positive since the node will have to delay its sleep until the event is fully processed. If the event is detected early within the awake period, \(\delta\) will be negative, as the node will go to sleep before \(T_w\) has elapsed. Finally, if no event is detected, \(\delta\) will be equal to 0. Here we can make an reasonable assumption that arrival time of events is greater than the residual time, \(\frac{1}{\lambda} > \delta\). Therefore, the event detection probability, \(u\), can be computed as follows:

\[
u = Pr(t_0 \leq t_e \leq T_w) + Pr(t_1 - T_e \leq t_e \leq t_1)
\]

\[
= \frac{T_w + T_e}{T_e + \delta} Pr(\delta \neq 0) + \frac{T_w + T_e}{T_e} (1 - Pr(\delta \neq 0))
\]

\[
= \frac{T_w + T_e}{T_e} \left[ 1 - Pr(\delta \neq 0) \delta \right]
\]

We can simplify (22) to \(u \approx \frac{T_w + T_e}{T_e} \) by showing \(1 \gg \frac{1}{T_e + \tau}\). We can compute the upper bound as follows. Let \(N_{[t_0, t_1]}\) denote the number of arriving events during \([t_0, t_1]\). Then \(Pr(\delta \neq 0)\) is bounded as:

\[
Pr(\delta \neq 0) \leq Pr(N_{[t_0, t_1]} \geq 2) = 1 - e^{-\lambda(T_e + \delta)} - e^{-\lambda(T_e + \delta)} \frac{\lambda (T_e + \delta)}{1 - \lambda} = 1 - e^{-\tau} - \tau e^{-\tau}
\]

where \(\tau = \lambda(T_e + \delta)\). Therefore, the following upper bound can be derived:

\[
Pr(\delta \neq 0) \delta \leq \lambda \delta \left[ 1 - \frac{1 - e^{-\tau} - \tau e^{-\tau}}{\tau} \right] \leq 0.3\lambda\delta
\]

Since typically \(\lambda\delta \ll 1\), the detection probability can be approximated as follows:

\[
u \approx \frac{T_w + T_e}{T_e} \leq 1
\]

Note that for an impulse event \((T_e = 0)\) the detection probability is the same as the duty cycle, i.e. \(u = d\).

Then from (18), (21), and (23), the node lifetime of the schedule-driven node can be described as a function of the detection probability \(u\) as follows:

\[
\tilde{T}_L(u) \leq \frac{[1 + u\lambda M_T]E_{Total}}{\bar{P}_e(u)}, \quad \bar{P}_{sch}(u) \geq \frac{\bar{P}_e + u\lambda M_E}{[1 + u\lambda M_T]}
\]  

Based on (24), the trade-off plot of detection probability and average power consumption is shown in the Figure 6. In order to increase detection probability, \( u \), a schedule based node can use two controllable parameters, duty cycle, \( d \) and duty period, \( T_c \). This is because \( M_T \), \( M_E \), and \( \bar{P}_e \) are also a function of those two parameters (see Equations (24)). Therefore, \( \bar{P}_{sch}(u) \) shows a dynamic behavior in event detection probability \( u \) and average power consumption, \( \bar{P}_{avg} \). We note that the trade-off trend of \( \bar{P}_{sch}(u) \) varies with \( M_T \) and \( \lambda \) as follows.

— If \( M_T < 0 \), \( \bar{P}_{sch}(u) \) is a convex function.

— If \( M_T > 0 \), \( \bar{P}_{sch}(u) \) is a concave function.

— If \( \lambda \to 0 \), \( \bar{P}_{sch}(u) \) approaches a linear function.

The above statements give the intuition behind the power consumption of the schedule-driven model as a function of the detection probability. If \( M_T < 0 \) and \( \lambda \) increases, the schedule-driven node quickly trades off detection probability for node lifetime, such that the trigger-driven mode of operation becomes increasingly favorable. We note that increasing duty period, \( T_c \) decreases \( M_T \) in (20), making \( \bar{P}_{sch}(u) \) into a convex function. On the other hand, if \( M_T > 0 \) and \( \lambda \) increases, \( \bar{P}_{sch}(u) \) becomes a concave function. It is important to note that decreasing the duty period \( (T_c) \) causes an increase in the residual time spent when an event is detected \( (M_T) \). This is because the awake period becomes too small to accommodate the event processing time. For frequent events, however, it is necessary to reduce the duty period in order to keep the detection probability at reasonable levels. For rare events, small duty cycles only increase the detection probability by a small amount, hence the duty period should be larger in order to avoid unnecessary energy expenditures. Meanwhile, differently from what happens with the
schedule-driven model, the trigger-driven model has a constant detection probability assumed to be equal to 1 in this paper. Thus the average power consumption for that model, $P_{td}(\lambda)$, is invariant over $u$, hence the constant line in Figure 6. The two curves meet at $u^* = (\lambda(\bar{P}_e K_T - K_E) + \bar{P}_e - P_{S_1}) / (\lambda(M_E - M_T P_{S_1}))$, characterizing an important decision point: for $u < u^*$, the schedule-driven model presents lower power consumption, while for $u > u^*$ the best choice is the trigger-driven mode of operation.

6. VALIDATION

The numerical correctness of our models was validated through simulation. Using empirical iMote2 measurements (Section 7.1), the two sensing schemes (scheduled and trigger-driven) were simulated in MATLAB. The simulator tracks energy dissipation of a node over a large time window by subtracting the accumulated power consumption from a given energy resource. It consists of an event generator (EG), a processing element (PE), and a communication unit (CU). According to assumption 2 (Section 3), the EG triggers the PE following an exponentially distributed random time after each event. The PE alternates a busy state (actively processing information) and an idle state. The CU also alternates a transmission state and a listening (idle) state. The workload simulated on the PE and CU is not taken from a real application, but modeled by a parameterized workload with properties represented by certain input parameters.

The workload of PE in processing stage is characterized by the expected processing time ($\bar{Y}$), average idle time ($\bar{\sigma}$), and job burst probability ($p_\sigma$) as shown in Figure 4(a). The job burst probability, $p_\sigma$, is the transition probability from $S_4$ to $S_2$ in the embedded chain which imply the intensity of job burst. The number of events within a given time duration follows a geometric distribution with parameter $p_\sigma$. Therefore the number of idle states $N_\sigma$ in the embedded processing chain is equal to $1/p_\sigma$. In the simulation, events are generated with a Bernoulli trial and the processing time of the job instance is generated according to an exponential distribution with mean of $\bar{Y}$. In the trigger-driven case all events were considered as being successfully detected. Meanwhile, for the schedule-driven simulation, the detected events were only those that intersected with the awake period.

![Diagram](image)

Fig. 7. (a) CSMA MAC protocol operation model, (b) TDMA MAC protocol operation model.

The workload of the CU in the communication stage is generated according to two typical MAC protocols (CSMA and TDMA with retransmission) shown in Figure 7 (a) and (b). For illustration purposes, we describe the equations, $Z$, $L$ and $N_L$ given a CSMA and a TDMA MAC, both of them with automatic repeat request. For simplicity, we assume the nodes can communicate with the base-station with 1-hop communication. For both MACs, if the current packet transaction is successful the packet is shifted out of the TX queue, and let $N_p$ denote the total number of packets in the TX queue. If the transmission fails, the node attempts to retransmit the failed packet within a given maximum number of times, $Q$. If the BER is known, then each bit in a packet can be modeled as a Bernoulli trial, and the number of erroneous bit in a packet as a Binomial distribution (assuming all bits are i.i.d.). The packet error rate (PER), then, $PER = 1 - (1 - BER)^{packet\ length}$, and the success of a packet transmission can be modeled as a Bernoulli random variable with probability $PER$. In our simulation, given a bit error rate, the packet error rate is simulated by considering each bit in the packet as a Bernoulli trial.

In CSMA, the radio listens to the channel before transmitting: if the channel is busy, the radio waits in idle mode and probe a channel state (busy or idle) after a random amount of time. In our simulation, a node performs exponential binary backoff at most $P$ times. The channel congestion is simulated using the channel congestion ratio, $\rho$: A node is likely to see a busy channel with probability $\rho$ during the communication stage. Let $\bar{T}_{bo}$ denote the average backoff delay, $\bar{T}_{tx}$ denote the average packet transmission duration, and $\bar{N}_{tx}$ denote the average number of retransmission per packet in Figure 7 (a). Then $Z = \bar{T}_{tx}$, $L = \bar{T}_{bo}$, and $L_N = N_p \bar{N}_{tx}$.

In TDMA, each node is assigned a unique time slot during which transmission may occur. Therefore, if transmission fails, packets can be retransmitted immediately without the need to listen to the channel beforehand. Differently from CSMA, in TDMA, if more time is needed due to an overflow of unsuccessful transmissions, the remaining data is delayed to the next cycle – hence $\bar{T}_{Comm}$ must be incremented by however many superframe cycles are necessary for a successful transmission (or before the packet is dropped altogether). Let $\bar{T}_w$ denote the waiting time until next slot assignment, $\bar{T}_{tx}$ denote the average transmission duration of packets per assigned slot, and $\bar{N}_{slot}$ denote the average number of TDMA cycle for successful transmission of $N_p$ packets in Figure 7 (b).

In order to better track the energy dissipation patterns, each simulation was conducted 10 times. The results in this section represent the average over those 10 simulations. The relative error between the model prediction and simulation was also computed (Table IV). The power modes and parameters of the simulations are shown in Table II and Table III respectively. For the trigger-driven case the preprocessor power is set to a constant 0.5W, which explains the additional power consumption when compared to the schedule-driven case in Table II.

The communication parameters, $Z$, $L$, and $N_L$ are computed for CSMA and TDMA given MAC specifications as shown in Table III. For TDMA, we assume that 10 nodes ($n = 10$) share a same channel with a 3 second cycle period ($T_{tdma} = 3$). For CSMA, we assume that each node performs a backoff with 1msec of unit time ($\delta = 1msec$) at most 5 times ($P = 5$). Each node experiences a busy channel for...
one out of five clear channel assessment trials ($\rho = 0.2$). The impact of number of nodes in the network, but not necessarily $n = 10$ is taken into account in $\rho$. For both TDMA and CSMA, a node tries retransmission at most 3 times ($Q = 3$) for a packet drop.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{S_0}$</td>
<td>$-/-1.8$ mW</td>
<td>$P_{S_1}$</td>
<td>$2.5/-$ mW</td>
</tr>
<tr>
<td>$P_{S_2}$</td>
<td>$237.5/237$ mW</td>
<td>$P_{S_4}$</td>
<td>$88.5/88$ mW</td>
</tr>
<tr>
<td>$P_{S_3}$</td>
<td>$307.9/307.4$ mW</td>
<td>$P_{S_5}$</td>
<td>$273.5/273$ mW</td>
</tr>
<tr>
<td>$C_R$</td>
<td>$6.9/6.9$ uJ</td>
<td>$C_P$</td>
<td>$2.206/2.206$ mJ</td>
</tr>
</tbody>
</table>

Table II. Power mode characteristics (trigger-driven / schedule-driven). For each state, the difference in power consumption between the trigger-driven and schedule-driven node parameters is always $0.5W$, since that is the value chosen for power consumption of the pre-processor.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{\sigma}$</td>
<td>4 sec</td>
<td>$\bar{Y}$</td>
<td>2 sec</td>
<td>$T_{tdma}$</td>
<td>3 sec</td>
<td>$\rho$</td>
<td>0.2</td>
<td>$P$</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M$</td>
<td>5 Kbyte</td>
<td>$l$</td>
<td>128 byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\tau$</td>
<td>12 byte</td>
<td>$P_a$</td>
<td>0.5</td>
<td>$n$</td>
<td>10</td>
<td>$\delta$</td>
<td>1 msec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$R$</td>
<td>250Kbps</td>
<td>$\alpha$</td>
<td>1</td>
<td>$L$</td>
<td>5.7 sec</td>
<td>$L$</td>
<td>0.0016 sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q$</td>
<td>3</td>
<td>$BER$</td>
<td>$10^{-3}$</td>
<td>$Z$</td>
<td>0.3724 sec</td>
<td>$Z$</td>
<td>0.175 sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_a$</td>
<td>0 sec</td>
<td>$E_{total}$</td>
<td>1.863 kJ</td>
<td>$N_L$</td>
<td>1</td>
<td>$N_L$</td>
<td>2.12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table III. Simulation parameters

Figure 8 shows the simulation results and model prediction (Equation (12)) for a trigger-driven application with CSMA and TDMA MAC. As shown in the figure, the average power consumption predicted by both models well follow the simulated Under submission to ACM Transactions on Sensor Networks, Vol. V, No. N, November 2007.
ones. The worst case relative error is 4.52% for CSMA and 1.14% for TDMA as shown in Table IV. CSMA tends to have slightly worse prediction than TDMA because $L$ in CSMA has large variance due to the binary exponential backoff.

For the schedule-driven node, the comparison between simulation and model prediction (Equation (24)) can be seen in Figure 9. As expected in Figure 6, the simulated curves for this model may follow a concave, linear, or convex trend depending on $M_T$ and $\lambda$. In the figure, $\overline{P}_{sch}$ follows a convex function due to the long duty period, $T_c = 360\,\text{sec}$ and relatively frequent event arrival, $\lambda^{-1} = 60\,\text{sec}$, causing $M_T$ to be negative. We note that our prediction in this case has large relative error (38.44%). Assumption 2 in Section 4.2 does not hold in this configuration: more than one event often arrive for the same awake period. However, as arrival rate decreases to $\lambda^{-1} = 1\,\text{hr}$, the power consumption $\overline{P}_{sch}$ becomes a linear function, as predicted, and the relative error decreases to 0.51%. Similarly, when $T_c = 1\,\text{sec}$ and $\lambda^{-1} = 5\,\text{sec}$, $\overline{P}_{sch}$ follows concave functions since $M_T$ becomes positive. The prediction in this case shows a fair relative error of 3.7%.
7. CAPTURING THE MODELS INTO A TOOL

To allow sensor node users and designers to use our models to drive design and development, we have developed a MATLAB sensor network lifetime toolkit, MATSNL. This tool implements the trigger-driven and schedule-driven models as well as the discrete-event simulation used to validate them (Section 6). Users can enter new specifications for node architectures and application profiles. MATSNL provides a set of commands for analyzing node lifetimes, comparing platform performance for different application profiles and computing power budgets for new sensor pre-processors, radios and processors.

MATSNL is organized into three main components: configuration, analysis and simulation. **Configuration** is used to specify the power properties of each component in the platform of interest. The user also specifies application profiles characteristics such as the event arrival rate, duty-cycle and detection probability. The **analysis** part of the tool implements the models described in the Sections 4.1 and 4.2. The **simulation** component is independent of the **analysis** component and runs a discrete event simulation on the parameters provided by the **configuration** component. All functions files in MATSNL share the scripts that describe the system parameters.

In its current form, MATSNL can perform four main functions:

1. Analyse lifetime of trigger-driven and schedule-driven nodes (*complf* command).
2. Analyze power breakdown per mode (*comppwr* command)
3. Analyze preprocessor power (*compprep* command)
4. Execute a discrete event simulation for a given configuration (*comp_sim* command).

In this section, we demonstrate the main features of MATSNL with example comparisons between alternative sensor node designs using four popular sensor nodes, Intel’s iMote2, Yale’s XYZ, Crossbow’s Micaz and Moteiv’s Telos. These examples are meant to be illustrative rather than exhaustive. The reader can study them in depth by changing the parameters inside the tool. The MATLAB code for the tool is publicly available and can be found at: [http://www.eng.yale.edu/enalab/aspire.htm](http://www.eng.yale.edu/enalab/aspire.htm).

7.1 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, in this Section we apply them in the decision-making process in the design of an experimental camera sensor node. Our goal is to decide whether it makes sense to develop a new version of our camera node (shown in Figure 10a) for the BehaviorScope project at Yale. The current camera node is an Intel iMote2 [L.Nachman 2005] coupled with a custom camera board we have designed with a commercial, off-the-shelf (COTS) image sensor, Omnivision’s OV7649. The node can be 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 10b) 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 high-end PXA 271 processor onboard the iMote2 will wait in a low-power state until triggered by the low-end PIC-based preprocessor that is always powered on (Mode $S_1$ in Table VII). In this state, the preprocessor applies a thresholding algorithm to the PIR measurements. If the observed motion exceeds a predefined value, the preprocessor will power up the iMote2 and camera board to acquire and process images of the event. If the image processing reveals something of interest, the node then transmits the information to a basestation. These transmissions take place with probability $\alpha$.

To provide more concrete numbers in our case study, we set up the camera node

to act as a simple single target localization device. In this case, an event is defined as the complete trajectory of a human centroid within the field-of-view of the camera. In this setting, the camera sensor node performs the following functions:

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.

Figure 11 shows the measured power profile pattern for each iMote2 according to our functional description. In this task, the PXA processor performs motion differencing between consecutive frames captured by the camera. The location of the target is the centroid of the pixels that moved between two consecutive frames. Each centroid computed is stored in the memory and the node transmits the stored centroid information to the base station with probability 1 (i.e. \( \alpha = 1 \)). Each centroid information consists of 2 bytes of location information, \((X,Y)\) grid and 4 bytes of time information, real time clock. Here we defined the event as follows.

— The event is defined as the complete trajectory of human centroid in the range of camera sensor.

— The event arrival rate is defined as the frequency a person enters the camera field of view. Therefore, it is roughly computed as the number of people observed by camera sensor divided by total observation time.

— The event duration is defined as the time interval a person spends in the camera’s field of view.

— The event is said detected if and only if there is no missed centroid during the event duration. Otherwise, it is said missed.

In our experiments, typically 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 enters the 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. We do however note that whether the centroids are sent

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>0.1/min</td>
<td>( Y )</td>
<td>2 sec*</td>
</tr>
<tr>
<td>( Z )</td>
<td>3.8 msec</td>
<td>( L )</td>
<td>0 msec</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>0 min</td>
<td>( \bar{N}_o )</td>
<td>1</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>1</td>
<td>( T_c )</td>
<td>10 min</td>
</tr>
<tr>
<td>( \bar{N}_L )</td>
<td>1</td>
<td>Total Energy</td>
<td>18.63 kJ</td>
</tr>
</tbody>
</table>

Table V. Reference scenario, measured (*)

\(^2\)This role is simplified and used for illustrative purposes only. More complex roles can be described using our tool presented in Section 7

immediately 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 and 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 a 23 byte 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 V.

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
power consumption and different transition times to the Normal mode. The Nor-
mal mode is the state where all internal power domains and clocks are enabled and
running. In the Idle and Deep Idle modes, the CPU core stops being clocked, but
for the latter the PXA is first switched into 13 MHz frequency. The 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. The 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 VI and VII.

Question 1: What is the expected lifetime for the existing (schedule-driven, Fig-
ure 10a) and proposed (Figure 10b) configuration? Using our measurements in
Table VI, the schedule-driven node will last for only 1.61 days if it is always on,
continuously sampling, since $T_L(1) = \frac{E_{Total}}{P_{W}(0.1/min)}$ by plugging $u = 1$ and $T_c = \infty$ in
Equation (24). The lifetime of the alternative, trigger-driven configuration depends
on the event arrival rate and can be computed using the model in Equation (14).
The trend for different arrival rates is shown in Figure 12b. At our default con-
figuration (PXA and Camera in Standby Mode), the trigger-driven iMote2 would
only last 8.45 days at most (1.03 days at least). Figure 12b 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

\footnote{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}

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

<table>
<thead>
<tr>
<th>Mode</th>
<th>Preprocessor</th>
<th>CPU</th>
<th>Camera</th>
<th>Radio</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>S&lt;sub&gt;0&lt;/sub&gt;</td>
<td>Deep Sleep</td>
<td>PXA271</td>
<td>OV7649</td>
<td>CC2420</td>
<td>9.8mW</td>
</tr>
<tr>
<td>C&lt;sub&gt;P&lt;/sub&gt;</td>
<td>4.863mJ</td>
<td>-</td>
<td>691µJ</td>
<td>4.863mJ</td>
<td></td>
</tr>
<tr>
<td>S&lt;sub&gt;2&lt;/sub&gt;</td>
<td>Normal</td>
<td>193mW</td>
<td>44mW</td>
<td>712µW</td>
<td>237.7mW</td>
</tr>
<tr>
<td>C&lt;sub&gt;R&lt;/sub&gt;</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>S&lt;sub&gt;4&lt;/sub&gt;</td>
<td>Deep Idle</td>
<td>88mW</td>
<td>44mW</td>
<td>712µW</td>
<td>132.7mW</td>
</tr>
<tr>
<td>C&lt;sub&gt;R&lt;/sub&gt;</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>194µsec</td>
<td></td>
</tr>
</tbody>
</table>

Table VII. 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 trigger-driven model in (12). The lifetime trend at different event arrival times as a function of preprocessor power is shown in Figure 12c.

**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 (24). The expected lifetimes for different detection probabilities are shown in Figure 12a.

**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 12d. 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-cycle=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 Standby mode. Comparing Figure 12c and Figure 12d, we notice that just lowering the power consumption of the preprocessor does not impact the lifetime trend of the trigger-driven node since $P_S$ 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 12b). 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 [D.Lymberopoulos and A.Savvides 2005]. According to our models, the use of such a mechanism would increase the lifetime of the schedule-driven node to 552 days (since $T_L(1) = \frac{P_{\text{Total}}}{P_S}$ by plugging $u = 0$ and $T_c = \infty$ in Equation (24)), a large improvement over the currently predicted 8.45 days for a non-Ideal preprocessor (see Figure 12d).

Fig. 12. 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,
7.2 Lifetime comparison and power analysis among platforms

In this section, we compare lifetime trends and corresponding power budget analysis among platforms; Micaz, iMote2, and XYZ. The first example examines the choice of processor for a wireless camera sensor node using numbers from the literature for the Micaz and XYZ nodes, and the iMote2 measurements we have taken during this work. The application parameters and camera sensor board configuration used here are the same as the ones we have described in our case study in Section 7.1. The three nodes, iMote2, XYZ and Micaz have the same radio. We also assume that they have the same sensor, and use their corresponding processor power consumptions and processing times to show how MATSNL could be used to study the effect of different processors. The iMote2 has a PXA 271 processor, which we run at 104MHz and XYZ has an OKI ARM processor running at 58MHz and Micaz has an 8-bit AVR processor running at 8MHz. To emphasize the processor effectiveness, we assume that for the same image processing and decision task the iMote2 processor takes 2 seconds, the XYZ processor takes 4 seconds and the Micaz processor takes 48 seconds.

A plot from the resulting comparisons are shown in Figure 13 (a) and (b). The Figures show that a trigger-driven operation model will be more suitable for this application, with XYZ lasting for 25 days, iMote2 for 100 days and Micaz for 200 days at most. Figure 13 (a) shows that the schedule driven operation is rather impractical since lifetime has to degrade drastically to achieve an acceptable detection probability. Moreover, we draw the reader’s attention to the fact that the effect of processing time on the lifetime is not obvious in the reported results for schedule-driven operation since lifetime heavily depends on the absolute power consumption of each processor. This is because for schedule-driven operation the sleep scheduling is blindly done regardless of the event arrival characteristic. Notice that even though XYZ has the lowest reported sleep power, 0.006mW compared to two other platforms, iMote2 (1.800mW) and Micaz (0.309mW), the benefit does not show on this figure since the effect on lifetime becomes noticeable only below 0.01 duty cycle (detection probability). Figure 13(b) shows that for MicaZ and iMote2 have
significant lifetime gain at lower event arrival rate compared to the schedule-driven operation.

[D.McIntire et al. 2005] shows that choice of processor heavily influences the task energy efficiency using benchmark of FFT, CRC-32, and FIR. This trend becomes obvious in 13(b). For computationally a demanding task, the advantage of using low power processor (as in MicaZ) is rapidly compromised by disadvantage of its low computational power (48 seconds per task) as aggregated workload (event arrival rate) is increasing rapidly. In contrast, the lifetime of iMote2 which has the most powerful CPU, PXA271, has decreased slowly compared to the one of MicaZ to show better lifetime performance beyond the point of arrival rate, 1 event per hour. This trade-off point between two platforms with high-end processor and low-end processor implies the potential advantage of using multi processor core node architectures (with high-end and low-end processor) so that platform exploits the processing power according to the computation efficiency. [D.McIntire et al. 2005] proposed LEAP platform adopting the architectures.

![Figure 14. Power breakdown comparison of multiple trigger-driven platforms](image-url)

A power budget/breakdown comparison between the three different configurations can be performed by running a single MATSNL command, `comppwr`. Figures 14 and 15 show the energy breakdown for the two models. Figure 14 explains why the MicaZ is not practical compared to the iMote2 at large workload (high arrival rate). As shown in the Figure, the processing power per event in MicaZ rapidly outpaces the one in iMote2 so that the total average power consumption of MicaZ exceeds that of iMote2 at points beyond the inter-arrival rate of 1 event per hour. Meanwhile for XYZ the preprocessing power takes most of portion of average power consumption to degrade lifetime performance as shown Figure 13(b). Such a high preprocessing power comes from large leakage current (3.5mA) of the CPU (Peripherals 1.69mA) and board (Radio board:0.883mA, Rest of the board: 0.927

Fig. 15. Power breakdown comparison of multiple schedule-driven platforms

Fig. 16. Lifetime trend of trigger-driven iMote2 over preprocessor power budget

mA)[D.Lymberopoulos and A.Savvides 2005]. Figure 15 shows that idle power dominates the power budget. In contrast, the other power budgets (such as sleeping and processing power) have negligible effects on the lifetime. The Figure implies an important rule of thumb in the energy conservation strategy of a schedule-driven node for those platforms at moderate event arrival rate: to minimize the idle power as much as possible.

The lifetime and trade-off trend for different preprocessor power costs can be computed by running a single MATSNL command, \textit{compprep}. Figure 16 shows the lifetime trend for different preprocessor power costs at various inter-arrival rates. As shown in the figure, the trigger-driven iMote2 has poor lifetime (less than one month) once the preprocessor power exceeds 5mW. This implies a very
important platform design rule that preprocessor power should not exceed 10mW. In contrast, for preprocessor power below 5mW lifetime performance drastically varies depending on the event arrival rate.

7.3 Application to evaluate performance of a deployed testbed

In this section we demonstrate a potential application of our tool for evaluating an existing wireless sensor network testbed. We select the VigilNet testbed setup [P.Vicaire et al. 2006] as an example. VigilNet uses a three level integrated power management (PM) architecture to provide high surveillance performance and energy efficiency simultaneously. The PM architecture provides three hierarchical services: the tripwire service, the sentry service and duty cycle scheduling. The tripwire service controls a network-level power manager dividing the sensor field into multiple sections. It applies different schedules (either of an active or a dormant state) to each section. The sentry service controls section-level PM by selecting a subset of nodes, called a sentry node. The sentry nodes are in charge of monitoring events given the scheduled duty cycle. Once the sentry node detects an event of interest it reports to the neighbor non-sentry nodes so that sensors start to perform complex event-detection and classification computations. In this demonstration we study the effect of platform choice on the lifetime performance of sentry node. Micaz and Telos nodes have been used in addition to XSM [P.Dutta et al. 2005] which was originally used in VigilNet.

The hardware configuration is compared in Table VIII. As shown in the table, the maximum processor capability is the same in terms of in MIPS (16MIPS). We note that the Tx and Rx power in the table is the net power of platform to operate Tx and Rx of radios; in order to run radios MCU must be active. For example, MCU must provide clock to CC2420 since it is configured as slave in SPI serial interface. Assuming all platforms are using the same sensors (therefore they process same type of data) we set the same average event processing time at 6 seconds which includes detection (2 seconds) and classification/velocity estimations (4 seconds). We assume that a sentry node requires at least 125msec to sense an event of interest. Therefore the minimum duty period is constrained to

\[ d \geq \frac{0.125}{T} \]  

The main application of VigilNet is detecting and classifying events (moving objects trespassing the network area, typically civilians, soldiers, and vehicles). The event of interest occurs typically at a rate of 0.1 per hour (2-4 events/day) modeled with a Poisson arrival distribution and event duration ranges from 1 to 10 seconds. Types of events are categorized into 4 groups according to its trajectory pattern passing though area [Q.Cao et al. 2005]. We consider type 1 targets where the start and end points of the target path are outside the monitored area.

Assuming there is only one sentry node in the monitored area the target detection probability as follows:  

\[ P(\beta) = \beta + \frac{\pi R^2 v T}{2vT} \]  

where \( v = \) Target velocity(m/sec), \( R = \) Sensing range(m), \( T = \) duty period(sec), \( \beta = \) duty cycle.

In the following formula all variable notations follow [Q.Cao et al. 2005]

Notice that our derived result of detection probability in previous Section 5 is the same as we get from this Equation (23) if we assume $T_e = \frac{\pi R^2}{v}$.

Taking the network into consideration with a deployment width of $L$, the detection probability is given as follows:

$$P_{detection}(\beta) = 1 - \exp\left(-\frac{2RLd(\beta + \frac{\pi R^2}{v})}{T}\right)$$

(26)

where $L=$ Target travel distance (m), $d=$ node density (the number of nodes/m$^2$).

For a more concrete example, we suppose that a target enters the field with a speed up to 10 m/sec. Furthermore we assume that this target is required to be detected by the first sentry node with a probability higher than 90% assuming the detection range is 10 meters. The following parameters are used for obtaining detection probability for duty cycle: $R = 10m$, $L = 5m$, $d = 0.01$ node/m$^2$, $T = 5sec$, $\beta = 0 \sim 1$, $v = 10m/sec$.

Figure 18(a) shows that the detection probability is a monotonic increasing function of duty cycle ($d$). As shown in the Figure, VigilNet compensates the degradation of detection probability resulted from the low duty cycle by reducing the duty period ($T_c$). The target detection probability is guaranteed to be nearly one regardless of duty cycle when $T_c = 1sec$.

We compares the node lifetimes of schedule-driven platforms at the a duty cycle between 0.125 and 1 and the node lifetimes of trigger-driven platforms at the Table VIII. XSM,MicaZ, and Telos platform hardware comparison

<table>
<thead>
<tr>
<th>HW Components</th>
<th>XSM @3V</th>
<th>MicaZ @3V</th>
<th>Telos @1.8V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Processor</td>
<td>ATmega128L 8 bit-16Mhz up to 16MIPS</td>
<td>ATmega128L 8 bit-16Mhz up to 16MIPS</td>
<td>TI MSP430 16 bit-8Mhz to 16MIPS</td>
</tr>
<tr>
<td>MCU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wakeup time</td>
<td>0.2mse</td>
<td>180usec</td>
<td>6usec</td>
</tr>
<tr>
<td>The lowest MCU power</td>
<td>30uW</td>
<td>75 uW</td>
<td>15uW</td>
</tr>
<tr>
<td>The Active MCU power</td>
<td>24mW</td>
<td>33 mW</td>
<td>3mW</td>
</tr>
<tr>
<td>Radio Unit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC1000</td>
<td>76.8 kbps</td>
<td>CC2420 250 kbps</td>
<td>CC2420 250 kbps</td>
</tr>
<tr>
<td>The Sleep Radio power</td>
<td>30uW</td>
<td>75 uW</td>
<td>15uW</td>
</tr>
<tr>
<td>The Tx(0dbm) Radio power</td>
<td>48mW</td>
<td>52 mW</td>
<td>35mW</td>
</tr>
<tr>
<td>The Rx Radio power</td>
<td>24mW</td>
<td>38 mW</td>
<td>60mW</td>
</tr>
<tr>
<td>Radio Wakeup time</td>
<td>2.5msec</td>
<td>860usec</td>
<td>580usec</td>
</tr>
<tr>
<td>Sensor Unit</td>
<td>PIR, Acoustic Magnetic Sensors</td>
<td>PIR, Acoustic Magnetic Sensors</td>
<td>PIR, Acoustic Magnetic Sensors</td>
</tr>
<tr>
<td>The Active Sensor power</td>
<td>22.044mW</td>
<td>22.044mW</td>
<td>13.226mW</td>
</tr>
<tr>
<td>The Sleep Sensor power</td>
<td>9uW</td>
<td>9uW</td>
<td>5uW</td>
</tr>
</tbody>
</table>

inter-arrival time range between 1 min and 1 day. The resulting plots of above two commands are shown Figure 17 (a) and (b). As shown in the figures, Telos shows better lifetime performance for both models. The schedule-driven Telos shows 3 times longer lifetime than other platforms at the minimum detection probability (90%). Furthermore Figure 17(b) shows that trigger-driven model brings a significant improvement on lifetime to all platforms: XSM (7 times), Telos (2.5 times), and MicaZ (5 times). For schedule-driven model, the lifetime of a sentry node depends not only on duty cycle but also on duty period because the total transition energy cost and minimum sensing time (125msec) depend on the duty period. Figure 18(b) shows such a dependency of the lifetime on the duty period, $T_c$. As shown in the Figure, the sentry node lifetime is maximized at $T_c = 4$ sec. Notice that increasing $T_c$ by more than 4 seconds diminishes the achievable lifetime by duty cycle (lifetime bound).

8. CONCLUSION

In this paper, we presented parametric lifetime model for trigger-driven and schedule-driven nodes that also takes the associated transition overheads into consideration. The application of the models in making decisions about a camera node platform help us isolate the dominant factors that limit lifetime in our design and provided valuable insight on how to proceed with the architecture. It has been common belief that the lowest sleep power is the key factor to determine lifetime of a schedule-driven node. However, our analysis based on MATSNL tool shows that the benefit is negligible except for a significantly low duty cycle (detection probability). We show that the main source of power consumption is the idle power during awake period. Throughout the paper the trigger-driven node shows better lifetime performance in general. By comparing the lifetime trade-off between multiple trigger-driven platforms we also show that the effect of processing time on lifetime dominates over the effect of preprocessing power as aggregate workload (event arrival rate) increases. Finally we demonstrate the application of our MATSNL tool for evaluating deployed testbed. In our VigilNet example, we shows the lifetime of the sentry node can be significantly improved by operating it in trigger-driven fashion or using the Telos. In the near future we extend 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/aspire.htm.

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.

REFERENCES


D. Lymberopoulos and A. Savvides. 2005. XYZ: A motion-enabled, power aware sensor node platform for distributed sensor network applications. In IPSN, SPOTS track.


