Modeling and Implementation of Energy Neutral Sensing Systems

Marcin K. Szczodrak
Columbia University
New York, NY
msz@cs.columbia.edu

Omprakash Gnawali
University of Houston
Houston, TX
gnawali@cs.uh.edu

Luca P. Carloni
Columbia University
New York, NY
luca@cs.columbia.edu

ABSTRACT
We present the modeling, implementation, and evaluation of a single wireless sensor network that executes energy-harvesting algorithms and sensing applications. The network energy-management is modeled as a feedback control system. An asynchronous execution of the energy-management and the application processes is modeled as a finite state machine. To evaluate our approach, we designed energy neutral sensing systems for two applications and implemented them with Fennec Fox.

Categories and Subject Descriptors
D.4.7 [**Operating Systems**]: Organization and Design—*Distributed Systems*

General Terms
Design, Modeling, Experimentation, Measurement

Keywords
Fennec Fox, Energy Neutral Sensing Network Systems

1. INTRODUCTION
Energy neutral sensing systems (ENSSys) achieve long-time operation by combining energy-harvesting hardware with software that regulates energy saving and spending. However, simply managing the energy resources is not the goal in itself. These systems have primary responsibility to execute the Wireless Sensor Network (WSN) or Cyber-Physical System (CPS) applications. Thus, there is an open research question on how to design systems running both the target applications, such as sensing and actuating, and the energy-management algorithms that enable long-time execution of these applications.

Despite advances in WSN research [2] and energy-harvesting algorithms [23], there are still few examples of successful technology transfers from research prototypes to actual commercial products. One of the major challenges has been the difficulty of realizing industry-level WSNs that can operate reliably for a long time with minimum energy and maintenance cost while supporting sophisticated applications. To address this problem it is necessary to develop methods for designing and implementing WSN systems that can run effectively both application processes and energy-management processes. Furthermore, these methods should enable design and software reuse across various product deployments.

In this paper we present the modeling, implementation and evaluation of a WSN running an energy-management system-process and a sensor data collection application-process. We show how the network energy-neutral operation can be modeled as a feedback control system and implemented on the Fennec Fox framework. Using the finite state machine (FSM) model of computation, we show how to separate the execution of the system energy-management from the application. This precludes a potential network communication conflict between these two processes, and it enables designing energy-management algorithms that are application agnostic. We present two case studies to show that the proposed system design and implementation methodology facilitates the composition of complex energy-harvesting systems with improved run-time performance.

2. RELATED WORK
Energy-Harvesting. Recent energy-harvesting improvements offer a spectrum of solutions to the problem of battery-constrained WSN life-time [23]. For example, WSN motes can be powered with solar energy [1, 3, 10] or by people's movement [7]. Motes can also communicate by reflecting TV signals [16] or by harvesting energy from a tiny radioisotope [22]. We apply some of these techniques to enable the execution of sensing applications, which currently operate on non-rechargeable batteries.

Energy Modeling. Modeling energy harvesting and consumption plays a critical role in designing energy-neutral systems. Real-life energy-harvesting traces [12] and power consumption measurements [18] enable the creation of energy models. These models can be used to predict energy-harvesting rates [9] and to run system simulations [17]. We introduce a feedback control model combining both the energy-harvesting and the energy-management algorithms.

Energy-Aware Execution. Energy-neutral systems require careful operation that is constantly aware of its energy resources. The network protocols use energy aware routing [15, 24] and new MAC protocols offer energy-conservation

ENSSys'13, November 13 2013, Rome, Italy

http://dx.doi.org/10.1145/2534208.2534211

primitives [4, 25]. There are new application development methods for power-efficient sensing [11] and actuating [13]. We show that designing an energy-neutral system should address both the system computation and communication costs.

3. BACKGROUND: FENNEC FOX

We implement the energy-neutral sensing system models based on Fennec Fox [19], an open-source framework for the execution of multiple processes in a WSN. The run-time framework execution of multiple processes is specified at design-time by a domain-specific Swift Fox programming language.

Swift Fox programs are used to model the execution of multiple processes across a network of low-power devices as a FSM. Thus, the whole network has a notion of *state*. Each state executes one or more *processes*. Transitions among the network states schedule the execution of different processes on the network motes. This resembles a typical model of an operating system, which switches execution of multiple processes on a single machine.

In Fennec Fox the same set of processes executes on all the motes. These processes require communication services supporting information exchange among the motes. Processes may have different and sometimes conflicting communication requirements. For example, one process may require a duty-cycled many-to-one data collection, and another process may need a high-throughput one-to-one data streaming [19]. To perform their duties, these two processes use different network and MAC protocols and sometimes different radio configurations. Because these two processes use different communication protocols, they cannot execute on the same network simultaneously. Fennec Fox solves this problem by running these processes asynchronously and reconfiguring the whole network protocol stack.

Each Fennec Fox process is scheduled to execute on a dedicated four-layer protocol stack. The top layer of the protocol stack, called Application, runs the process code. For instance, a process code might periodically sample a sensor, compute the average of the last few samples, and send it across the network. The Network layer runs the network protocol (e.g. CTP [6] for data collection or Trickle [15] for data dissemination) that provides the communication service required by the Application layer code. The MAC layer executes one of the available MAC protocols, such as CSMA, TDMA, and their duty-cycled version. The Radio layer provides the driver code controlling a particular radio chip.

Fennec Fox processes are defined in a Swift Fox program. In the program, each process must specify the application code and list the network, MAC, and radio mechanisms that support the application's execution. The compiled program is installed on the WSN motes. At run-time, as Fennec Fox schedules the execution of a process across the network, it also schedules the execution of the network and MAC protocols together with the radio driver.

4. ENERGY-NEUTRAL SYSTEM MODEL

We define a feedback-control model for an energy-neutral sensing system. Within this model we distinguish two types of processes: the application processes and the energy-management processes executing algorithms for sustainable energy-neutral network operation. The energy-management processes generate additional network traffic by reporting the motes' energy harvesting and spending rates. Because the application processes may require network communication protocols conflicting with those used by the energy-management, we propose to asynchronously execute these two types of processes.

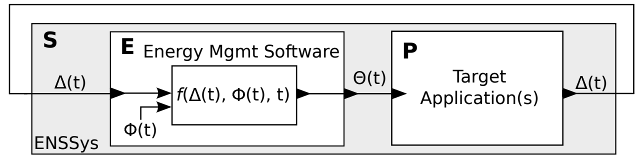

Figure 1: Feedback control model of a system executing application and energy-harvesting processes.

Feedback-Control Model of Energy-Neutral System. First, we define the ENSSys parameters. Let $\Delta(t)$ and $\delta(t)$ be the energy consumed by all the motes and a single mote at time t, respectively. Let $\Phi(t)$ and $\phi(t)$ be the energy harvested by all the motes and a single mote, respectively. Both Δ and Φ represent the system continuous dynamics controlled by the energy-management function f:

$$\Theta(t) = f(\Delta(t), \Phi(t), t) \tag{1}$$

where $\Theta(t)$ represents the energy-related actuation signals sent to the network and $\theta(t)$ denotes a single control signal sent to one mote. This function provides a general model of the energy-neutral system, whose operation is constrained by the amount of energy consumed and the amount of energy harvested by the motes at a given time.

Second, we define the ENSSys processes. Let P be the set of processes that execute the *target applications* performing sensing, actuating, or computing. Let E be the set of processes with algorithms managing energy-harvesting and ensuring the energy-neutral operation. The whole system consists of S processes, where $S = \{P \cup E\}$. During the ENSSys execution, a subset of these S processes is scheduled to run across the network.

With the defined ENSSys parameters and processes, we compose a feedback control model for sustainable energy-neutral operation. Fig. 1 shows the energy-management processes E receiving signals $\Phi(t)$ and $\Delta(t)$. The control function f computes the actuation signals $\Theta(t)$ which impact the execution of the target applications P. The execution of the P applications across the network consumes energy at $\Delta(t)$ rate, which is the input parameter to the energy-management processes.

Sensing and Actuation of Energy Harvesting. In the presented feedback-control model, the energy-management processes E require the following two mechanisms:

1. **Energy Harvesting Sensing (EHS):** reports the energy-harvesting rate $\Phi(t)$.
2. **Energy Harvesting Actuation (EHA):** applies the energy control signal $\Theta(t)$ computed by the function f.

These two mechanisms can be implemented as a single or multiple processes. For example, in one energy-management approach, every mote individually monitors its energy resources and adapts its work accordingly [8]. Here, both EHS and EHA may run as a single process. In a different work, motes exchange their energy status [24], which requires separate processes for executing EHS and EHA, because they have different communication requirements for energy sensing and actuating.

Energy Harvesting Communication Requirements. Depending on how and where the ENSSys parameters are computed and sent, the energy-management has one of the following network communication characteristics:

- **Local:** - On each mote individually, EHA computes the actuation signals based on the EHS reports from its own mote. The actuation signals only affect the mote itself.
- **Neighborhood:** - EHS and EHA processes exchange their messages among the motes in the neighborhood. For instance, a mote may broadcast its energy-harvesting status $\phi(t)$ and adapt its operation according to the information $\Phi(t)$ received from other motes.
- **Global-Distributed:** - Both EHS and EHA algorithms run on all the motes and exchange $\Phi(t)$ and $\Delta(t)$ across the network to collectively compute the energy control decisions $\Theta(t)$.
- **Global-Centralized:** - EHS sends both the $\Phi(t)$ and $\Delta(t)$ across the network to the central node. After centrally computing $\Theta(t)$, EHA sends $\theta(t)$ to all the motes in the network.

These different network characteristics of the energy-management processes might be conflicting with the network traffic generated by the target applications. For example, an energy-management process may require CSMA MAC, while the target application may require TDMA MAC. But these two MAC protocols cannot effectively coexist together in the WSN at the same time.

Scheduling Energy Management Processes. To design and implement a single network with the energy-management E and the target application P processes, we propose to schedule their execution asynchronously. This separate execution is mandatory when the energy-management algorithms and target applications have conflicting communication specifications. When there are no communication conflicts, we still may design ENSSys with the asynchronous execution of the energy-management and the application processes for the following reasons:

1. Preventing degradation of E and P processes execution by isolating their network communication traffic.
2. Improving the energy-harvesting and consumption estimates by computing f without executing the target applications at the same time.
3. Decoupling E from P to enable modular system design and to reuse the E processes with other P.

We apply the FSM model of execution supported by the Fennec Fox framework to asynchronously run the energy-management and target application processes. In the Swift Fox programming language, we map the processes with the conflicting communication requirements into separate FSM states. Then, we create state transition events so that the

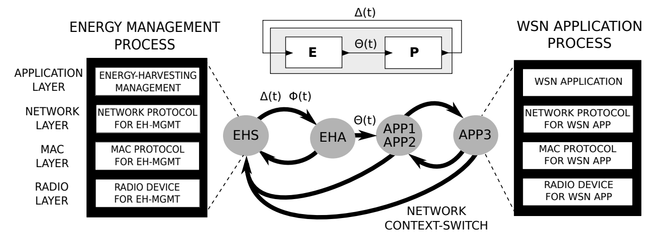

Figure 2: WSN context-switch between the application and the energy-management processes.

system only runs either the energy-management processes (and switches the execution among E processes) or runs the target applications (and switches the execution among the P processes).

Fig. 2 illustrates how we map the feedback control model of the energy-neutral system into a FSM that is programmable in Swift Fox and executable by the Fennec Fox framework. In this example, the FSM has four states: two for EHS and EHA energy-management algorithms and two states for the three target applications denoted as $APP*$. $APP1$ and $APP2$ run concurrently within the same state while $APP3$ runs in a separate state with its own network stack. The transitions from the states running P processes to the states running E processes ensure that the up-to-date control inputs $\Delta(t)$ and $\Phi(t)$ are made available to the system energy controller. The transitions from the states with E processes to the states with P processes set the target applications to run with the most recent energy control signals $\Theta(t)$.

5. CASE STUDIES

We present two cases studies to evaluate our approach. The first case study shows the energy-management processes executing asynchronously with the application processes on the same WSN. The second case study applies the existing Fennec Fox mechanisms to achieve run-time system adaptation based on the energy-harvesting rate.

5.1 Adapting workload to residual energy

We run a simulated replication of the WSN habitat monitoring study [21]. In this application, sensor nodes were deployed outdoor to collect climate information about sunlight, humidity, air pressure, and temperature. In the original deployment, the key engineering challenge was to set the sensor sampling rate so that the network would operate for a sufficient period of time. We address this problem by using solar cells. In particular, we construct a sensing system where the sensor sampling rate $\Theta(t)$ is dynamically adjusted to the amount of the available energy.

Energy-Harvesting Software. The application sensing rate $\Theta(t)$ is computed by the control function f as the difference between the rate $\Phi(t)$ at which energy is harvested and the rate $\Delta(t)$ at which energy is consumed by the sensing application:

$$\Theta(t) = f(\Delta(t), \Phi(t), t) = C(\Phi(t) - \Delta(t)) \quad (2)$$

where C is the signal control scaling parameter. For example, for $C < 1$ the controller sets the application consumption rate below the energy-harvesting rate. Fig. 3 shows the

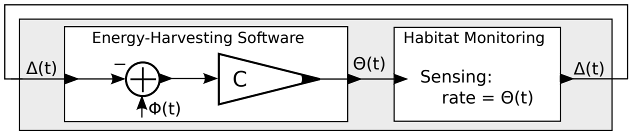

Figure 3: Feedback control model where sensing rate is adjusted to the energy-harvesting rate.

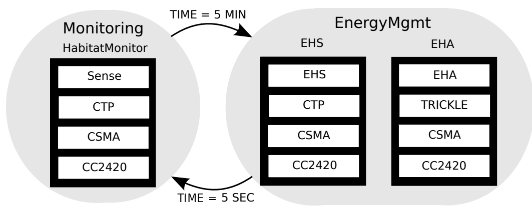

Figure 4: Fennec Fox modeling execution of the application and the energy-management processes.

feedback control model of the WSN for the habitat monitoring. The energy-management function f is computed by the Energy-Harvesting Software, and its output $\Theta(t)$ is sent as a sensing rate parameter to the Habitat Monitoring process.

To implement the modeled system, we specified the communication characteristics of the energy-management software. In particular, we defined the communication mechanisms that deliver the energy harvesting and spending rates as the inputs to the function f, and then transmit the computed sensing rate back to the motes. Specifically, from Section 3 we chose to compute the energy-management centrally. Therefore, the network has the Global-Centralized communication characteristics, with:

- EHS collecting information about the energy-harvesting rate $\Phi(t)$ at the central system.
- EHA computing f and disseminating $\Theta(t)$ to all the nodes in the network.

These communication characteristics demand the following network protocols. The EHS many-to-one data collection requires a protocol such as CTP [6], with the $\Phi(t)$ data from all the motes sinking at the central system node. The EHA one-to-many data dissemination needs a protocol such as Trickle [15], with the $\Theta(t)$ data announced by the central system node to all the motes in the network.

Fennec Fox Model and Swift Fox Program. We implemented the habitat monitoring as an application executing on the Fennec Fox framework [19]. To control the application's sensing rate, every 5 minutes the network switched to execute the energy-management processes. Fig. 4 shows two network states: *Monitoring* - for the execution of the *HabitatMonitor* protocol stack with the *Sense* application, and *EnergyMgmt* - consisting of two protocol stacks running the EHS and EHA software[1].

Fig. 5 shows the Swift Fox program specifying the asynchronous execution of the application and the energy-management processes, together with their supporting communication

```
# Shared variables
uint8_t rate = 0

# Stack Configurations: conf <conf_d> {<app> <net> <mac> <radio>}
conf HabitatMonitor { sense(rate, NODE, 2) ctp(2) csma() cc2420()}
conf EHS {ehs(113) ctp(113) csma(0, 1, 1) cc2420()}
conf EHA {eha(rate) trickle() csma(0, 1, 1) cc2420()}

# States: state <state_id> [priority level] { <conf_id> ... }
state Monitoring L3 {HabitatMonitor}
state EnergyMgmt L1 {EHS EHA}

# Events: event <event_id> {<source> <condition> [scale]}
event CheckEnergy {timer = 5 min}
event TimeOut = {timer = 5 sec}

# Policies: from <state_id> to <state_id> when <event_id>
from Monitoring goto EnergyMgmt when CheckEnergy
from EnergyMgmt goto Monitoring when TimeOut

# Definition of the initial state: start <state_id>
start Monitoring
```

Figure 5: Program of the system from Fig. 4.

protocols. We verified the correct execution of the application and the energy-management software by compiling and executing the Swift Fox program on a testbed built with the Open Testbed Framework [20] and running Zolertia Z1 motes. Because those motes do not provide any energy-harvesting capabilities, we used a simulator to conduct WSN energy-management studies.

Simulation. We used the TOSSIM [14] simulator that allowed us to simulate the same code that runs on the target hardware (e.g. Zolertia Z1), with the exception of the radio driver. We extended TOSSIM to support simulation of the solar energy by accessing real-life traces of the irradiance logs from the Humboldt State University in Arcata, California [12]. We developed a simulation interface to configure the energy-harvesting parameters[2] and to define the simulated energy consumption models according to the power consumption reports [18][3].

We compared the performance of the energy adaptive system against three naive energy-management strategies. The first strategy, called Aggressive, runs with a high-fixed sensing rate of 1Hz. The second strategy, Conservative, has a low-fixed rate of one sample every 8 seconds but ensures 24-hour operation. The third strategy has a scheduler, which during the day time hours (10:00-13:10) samples aggressively and otherwise conservatively. Our proposed Adaptive strategy computes the actuation signal $\Theta(t)$ according to the function f from Eq. 2, with C = 1.01. This strategy dynamically adapts the sensing rate to the energy-harvesting rate and is programmed according to the Fennec Fox model from Fig. 4.

Experimental Results. Fig. 6 shows the experimental results comparing the four energy-management strategies over the same 24-hour irradiance data trace. Table 1 reports the aggregate metrics covering the entire experiment. Two metrics of particular interest are the average rate at which the habitat monitoring application attempts to sample sensors and transmit a message, and the percentage of times when at the given sampling rate there is enough energy to run the application.

[1]Although both the EHS and EHA processes require different protocol stacks, they can run concurrently, because the $\Theta(t)$ dissemination occurs after the $\Phi(t)$ data collection, without network communication conflicts.

[2]In the experiments, we set the solar cell area to 1x2cm, cell efficiency to 20%, and the battery capacity to 400J.

[3]Application sensing and transmitting costs 33mJ, otherwise on average a mote consumes 0.15mJ.

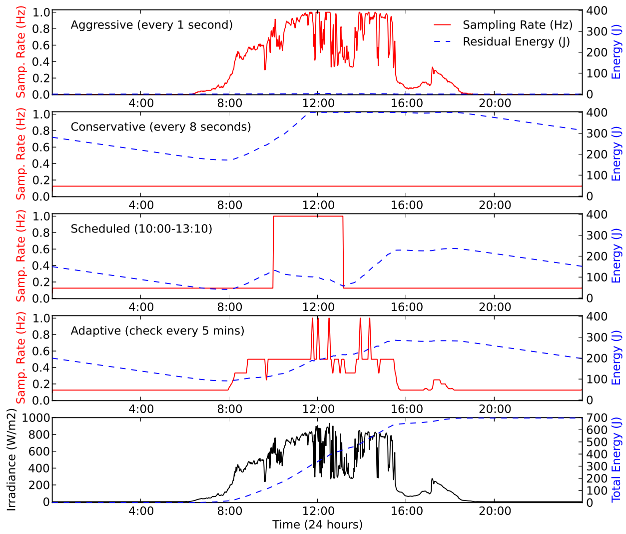

Figure 6: Performance of the solar-cell-based energy harvesting and management strategies.

Strategy	Avg. Samp. Rate (Hz)	Total Packets	Sampling Success (%)
Aggressive	0.243	21006	24.3
Conservative	0.125	10800	100
Scheduled	0.240	20775	100
Adaptive	0.241	20885	100

Table 1: Energy-management experimental results.

From the experimental results we observe that the Aggressive strategy spends energy as soon as the motes harvest it. Only during the middle day hours the battery gets charged up to 3.3 J. When sampling every second, only 24.3% of the time the motes have sufficient energy resources to complete sensing and transmitting. This yields the average successful sampling rate of 0.243 Hz. In particular, as soon as the day ends, the motes stop working and are not operational until the beginning of the next harvesting period. The Conservative strategy achieves 100% successful sensing and transmitting, with a low 0.125Hz sampling rate. During the middle hours of the day, this strategy is missing the available energy-harvesting resources, because the battery reaches its maximum capacity. The Scheduled strategy operates within the limits of the harvested energy, transmitting just 231 less reports than the Aggressive strategy, while successfully operating for all the time. However, to achieve this Scheduled strategy results, we had to run an off-line brute-force parameter optimization to determine when to switch between the two sampling rates. Finally, the Adaptive strategy has a higher average sampling rate than the Scheduled strategy and is transmitting only 0.99% less reports than the Aggressive strategy. Further, the Adaptive strategy sustains a continued sensing and transmitting operation, without requiring off-line schedule optimization and is agnostic to the energy-harvesting rate.

5.2 Adapting execution to residual energy

For the second case study, we considered the energy-harvesting active network tags [8]. The tags run on the energy harvested from the indoor-deployed solar cells and transmit messages over a prototype of an ultra-wideband impulse radio. Our goal is to understand how such system would perform with a low-power radio already available on the market. For example, a similar energy-harvesting method was used in the leaf-to-branch communication approach [25], with the CC2420 radio on the Epic Core motes [5]. In the following experiments, we run two applications on the active network tags. The first application is called *Switch*. Once a tag is pressed, this application sends a broadcast message requesting to turn on the light. The second application, *Occupancy*, broadcasts the motion's sensor measurement, every minute.

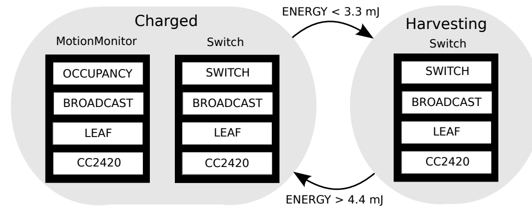

Figure 7: Fennec Fox scheduling execution of processes according to the level of the harvested energy.

Energy Spending and Harvesting. We design two simulation experiments with the network tags running the two applications. In the first design, we let both applications to run concurrently until they consume all the available energy resources. In the second design, we prioritize the *Switch* application over the *Occupancy* one and ensure that during the night hours there is enough energy to press the light switch and turn on the light for at least 10 times.

In the experiment, we use the indoor solar energy-harvesting model [9][4] and derive the indoor irradiance traces from the outdoor ones[5]. Following the energy consumption reports [18], we define the energy-management function f:

$$\theta(t) = f(\delta(t), t) = \begin{cases} 0 & \text{if} \quad \delta(t) \;<\; 3.3\ mJ \\ 1 & \text{if} \quad \delta(t) \;>\; 4.4\ mJ \end{cases} \tag{3}$$

where the output value $\theta(t)$ specifies if there is enough energy resources to run the *Occupancy* application, while securing the energy for 10 *Switch* application executions[6].

Fennec Fox Implementation and Simulation. We use the Fennec Fox energy-based events to trigger a context-switch between the following two states. The *Charged* state runs with the two application processes. The *Harvesting* state only executes the *Switch* application. Fig. 7 shows the FSM model of the system switching between those two states. Here, the context-switch is driven by the energy-management function f from Eq. 3, thus the system state depends on the residual energy level.

To minimize the broadcast message power consumption, both applications use a low-power MAC protocol. This protocol, called Leaf, follows the leaf-to-branch communication

[4]The solar cell area is 1x2cm and the cell efficiency is 1%.
[5]Because indoor solar energy levels are 1 to 3 orders of magnitude lower than the outdoor once [9], we divide the outdoor traces [12] by 100.
[6]1 second of sensing consumes 0.015mJ. A message broadcast requires 0.2mJ. A single *Switch* transmission consumes 0.2mJ. Fennec Fox state reconfiguration overhead is 0.19mJ. 1 minute of running *Occupancy* needs 1.1mJ. A tag stops sensing when the energy is less than 3.3mJ. A tag starts sensing when the energy is more than 4.4mJ.

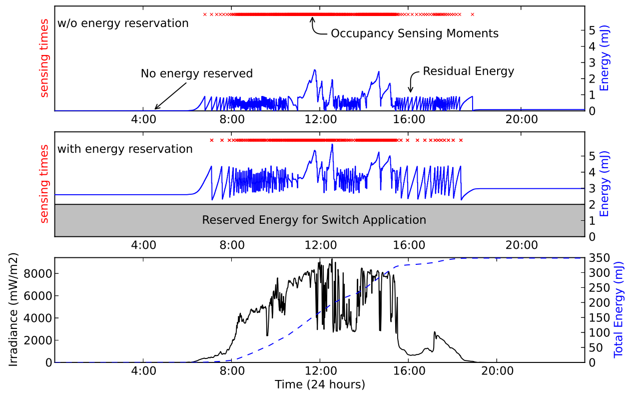

Figure 8: Energy reservation for process execution.

model [25]. Before sending a message, Leaf checks that the radio is turned on and that there are no other ongoing transmissions. Once a message is sent, the radio is turned off until the next transmission.

Fig. 8 shows the simulation results of the two system designs executing the *Switch* and *Occupancy* applications. In the upper graph, the system concurrently executes both applications. In the middle graph, the system reserves the energy resources for executing the *Switch* application, as in the model from Fig. 7. The bottom graph reports the irradiance measurements and the available total energy resources.

The experimental results show the tradeoffs between the two system designs. In the upper graph, the *Occupancy* application sent 387 sensor reports between 6:48am and 6:55pm, when it depleted all the energy resources. In the middle graph, the *Occupancy* application sent, during the 49 minutes shorter period, 356 reports, only 0.92% less than the system without reserving energy for over-the-night *Switch* execution. The system switched between the *Charged* and *Harvesting* states 138 times, with Fennec Fox consuming 27.6mJ.

6. CONCLUSIONS

We analyze the problem of combining into a single wireless sensor network (WSN) both the energy-harvesting software algorithms and the applications collecting sensor measurements. First, we introduce the models of computation for the energy-management software and for scheduling the execution of multiple processes on the same WSN. The network energy-management is modeled as a feedback control system. The distributed multiprocessing is based on the Fennec Fox finite state machine model of computation.

We demonstrate the system design and implementation methodology on two WSN applications. The first application adjusts the sensing rate according to the rate at which the energy is consumed and harvested. The second example presents two applications with different execution priorities. The lower-priority application runs only when there are enough energy resources to ensure the execution of the high-priority application. These examples show energy-neutral sensing systems with energy managed by following a feedback control model, programmed in Swift Fox and executed by the Fennec Fox framework.

Acknowledgements: This project is partially supported by the National Science Foundation under Award #931870 and by an ONR Young Investigator Award. Omprakash Gnawali was partly supported by a generous gift from Cisco.

7. REFERENCES

[1] Y. Afsar et al. Evaluating photovoltaic performance indoors. In *Proc. of the Photovoltaic Specialists Conf.*, pages 1948–1951, June 2012.
[2] D. Culler, D. Estrin, and M. Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37:41–49, Aug. 2004.
[3] P. Dutta et al. Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In *Proc. of the IPSN Conf.*, pages 407–415, Apr. 2006.
[4] P. Dutta et al. A building block approach to sensornet systems. In *Proc. of the ACM SenSys Conf.*, pages 267–280, Nov. 2008.
[5] P. Dutta et al. Wireless ACK collisions not considered harmful. In *ACM HotNets Work.*, Oct. 2008.
[6] O. Gnawali et al. Collection tree protocol. In *Proc. of the ACM SenSys Conf.*, pages 1–14, Nov. 2009.
[7] M. Gorlatova et al. Movers and shakers: Kinetic energy harvesting for the internet of things. *CoRR*, abs/1307.0044, 2013.
[8] M. Gorlatova et al. Prototyping energy harvesting active networked tags (EnHANTs). In *Proc. IEEE INFOCOM'13 mini-conference*, pages 585–589, Apr. 2013.
[9] M. Gorlatova, A. Wallwater, and G. Zussman. Networking low-power energy harvesting devices: Measurements and algorithms. In *Proc. IEEE INFOCOM'11*, pages 1602–1610, Apr. 2011.
[10] A. Hande, T. Polk, W. Walker, and D. Bhatia. Indoor solar energy harvesting for sensor network router nodes. *Microprocessors and Microsystems*, 31(6):420–432, 2007.
[11] V. Jelicic et al. Context-adaptive multimodal wireless sensor network for energy-efficient gas monitoring. *Sensors Journal, IEEE*, 13(1):328–338, Jan. 2013.
[12] N. R. E. Laboratory. Measurement and instrumentation data center. [Online] `http://www.nrel.gov/midc/hsu/`.
[13] S. P. Lau, G. Merrett, and N. White. Energy-efficient street lighting through embedded adaptive intelligence. In *Proc. of ICALT Conf.*, pages 53–58, May 2013.
[14] P. Levis et al. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *Proc. of the ACM SenSys Conf.*, pages 126–137, Nov. 2003.
[15] P. Levis et al. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proc. of the NSDI Symp.*, pages 15–28, Mar. 2004.
[16] V. Liu et al. Ambient backscatter: wireless communication out of thin air. In *Proc. of the ACM SIGCOMM*, pages 39–50, Aug. 2013.
[17] A. Seyedi and B. Sikdar. Modeling and analysis of energy harvesting nodes in wireless sensor networks. In *Proc. of Communication, Control, and Computing Conf.*, pages 67–71, Sept. 2008.
[18] V. Shnayder et al. Simulating the power consumption of large-scale sensor network applications. In *Proc. of the ACM SenSys Conf.*, pages 188–200, Apr. 2004.
[19] M. Szczodrak, O. Gnawali, and L. P. Carloni. Dynamic reconfiguration of wireless sensor networks to support heterogeneous applications. In *Proc. of IEEE DCOSS Conf.*, pages 52–61, May 2013.
[20] M. Szczodrak, Y. Yang, D. Cavalcanti, and L. P. Carloni. An open framework to deploy heterogeneous wireless testbed for cyber-physical systems. In *Proc. of the IEEE SIES Symp.*, pages 215–224, 2013.
[21] R. Szewczyk et al. Habitat monitoring with sensor networks. *Commun. ACM*, 47(6):34–40, June 2004.
[22] S. Tin and A. Lal. Saw-based radioisotope-powered wireless rfid/rf transponder. In *Ultrasonics Symposium*, pages 1498–1501, Oct. 2010.
[23] A. S. Weddell et al. A survey of multi-source energy harvesting systems. In *Proc. of the DATE Conf.*, pages 905–908, Mar. 2013.
[24] Y. Wu and W. Liu. Routing protocol based on genetic algorithm for energy harvesting-wireless sensor networks. *Wireless Sensor Systems, IET*, 3(2):112–118, July 2013.
[25] L. Yerva et al. Grafting energy-harvesting leaves onto the sensornet tree. In *Proc. of the IPSN Conf.*, pages 197–208, Apr. 2012.