Sophisticated Sensing on Transient Power

Josiah Hester
School of Computing
Clemson University
jhester@clemson.edu

ABSTRACT

For decades sensing systems have relied solely on battery power for execution of all activities; this has caused the focus of much research to go towards reducing energy consumption to extend the usable lifetime of a sensor. Recently, a new class of batteryless devices has arisen that promise operation in perpetuity, but often at the cost of reliability, and complexity. Programming, profiling, debugging, and building these applications is a significant challenge; designers must often be capable of implementing custom hardware to manage energy, while writing code in an environment that does not guarantee task completion. In this abstract, we motivate batteryless sensing, examine the state of the art, and propose a novel approach to programming, profiling, debugging, and building, tiny, batteryless sensors.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Microprocessor/microcomputer applications; D.3.2 [**Programming Languages**]: Language Classifications—*Specialized application languages*

General Terms

Measurement, Experimentation, Performance, Reliability

Keywords

Energy Harvesting; Capacitor; Federated Energy; Ekho; UFoP; Embedded System

1. INTRODUCTION

Wireless sensors use of batteries as the primary energy source has hampered their ability to be deployed for long periods of time. Batteries only have a usable lifetime of a few years if used carefully. Additionally batteries are expensive, large, and pose environmental risks when disposed—often requiring human intervention or maintenance. Because of the high costs associated with batteries, a new class of sensors emerged that instead rely on energy harvesting for all computation tasks. By replacing batteries with capacitors, these devices reduce environmental risk, can be made smaller and cheaper, and can be deployed for much longer periods of time. When sensing deployments can last on the order of decades instead of months, very long term studies of infrastructure, wildlife, and human factors become realistic.

Wireless sensing systems generally operate under the assumption that power is a stable, if limited, resource. This assumption has been challenged by batteryless platforms such as computational RFID, that are powered off small capacitors and rely on energy harvested from an RFID Reader. These devices are often only able to store enough energy for a few hundred milliseconds of work, losing memory and time when energy is not sufficient to turn on the device. Devices often operate blind, reconstructing progress from checkpoints left by a possibly long dead version of themselves. Long running tasks must be executed piecemeal across reboots using checkpointing and careful scheduling. Even *with* careful checkpointing and scheduling, sensing outcomes are never guaranteed.

Building applications resilient to inconsistent power is challenging because system designers lack the debugging tools, language support, and hardware platforms. Because of this, batteryless sensor deployments have been limited to simple programs at small scale. To move batteryless sensors beyond computational RFID and into the mainstream of traditional sensing, the developmental burden of batteryless applications must be eased.

The central thesis of this research is that batteryless devices can support long-running, sophisticated sensing applications. To test this thesis we envision building a sensing platform that transparently manages the temporal, computational, and consistency difficulties caused by flaky power, specifically we look at the following:

- Hardware solutions to managing energy storage and timekeeping between power failures (Section 2)
- Language and runtime support for computing on batteryless devices (Section 3)
- Tools which enable repeatable experimentation of batteryless devices (Section 4)

Innovating in these three areas will enable application developers to quickly and confidently prototype, test, and deploy perpetual sensing systems.

2. HARDWARE

Software solutions to managing energy and keeping time on transient power fall far short of enabling sophisticated applications. Microcontrollers draw too much energy to be able to handle energy

SenSys'15, November 1–4, 2015, Seoul, South Korea..
ACM 978-1-4503-3631-4/15/11.
DOI: http://dx.doi.org/10.1145/2809695.2822521 .

management tasks. Additionally, they cold boot too slow to keep accurate time throughout power failures. It is imperative that certain low level tasks are handled by the hardware layer to reduce the energy and time overhead as much as possible. In order to simplify development of batteryless sensing applications, we have relegated energy management, and timekeeping to the hardware layer.

Federating Energy: In most sensing systems, energy originates from a single energy store (a battery or super capacitor). This convention causes task and component coupling for capacitor based devices. This happens for the simple reasons that charging a capacitor large enough to support sensing, computation, and communication, takes longer than charging a smaller capacitor that supports just one of those functions. We have proposed UFoP (the United Federation of Peripherals) [2] as a solution to this problem; UFoP federates a device's total energy storage into multiple capacitors, each dedicated to a specific component or task.

Batteryless Timekeeping: One of the greatest challenges to batteryless computing is keeping track of time across power failures. When energy runs out, the microcontroller, volatile RAM, and all clocks are reset. This means that any previous timestamps have no meaning, since the local clock will start back at zero. Previous approaches to keeping time [3] used the data remanence properties of SRAM to measure the time since a power loss. We can extend this approach using a dedicated hardware component and an ADC for precise measurement—giving us greater configurability and precision.

3. LANGUAGE

Without language and runtime support for batteryless sensing, the hardware techniques previously described will have limited usefulness. Currently applications for batteryless sensing are highly constrained—all tasks must happen in one power cycle. This means that tasks must be simple, execute quickly, and have no dependencies, otherwise the program may not complete before a power loss. Previous work has focused on checkpointing, energy aware sensing, and scheduling; however existing approaches have too high of an overhead, or do not consider how the loss of timekeeping effects the duty cycle. We propose a language and runtime system for batteryless sensing devices that addresses these core problems. This system will be designed to track sensor data through power failures while also enabling task dependencies, and task scheduling.

Track sensor data through power failures: Application developers have no way of knowing how old data is that was gathered before a power failure, meaning that passing that data on could be a waste of energy and time. Using the hardware support for timekeeping described in Section 2, checkpointing, and application hints supplied by developers, our runtime can dynamically decide whether to use, or replace existing senor data gathered in a previous duty cycle. These automatic runtime decisions can enabling intelligent, long term gathering of sensor data despite flaky power.

Enable task dependencies, and task scheduling: Our language will enable developers to describe tasks, and their dependencies, so that sophisticated duty cycles can be created. Our runtime will take these tasks, with their real-time constraints, and attempt to make progress on them when energy is available. Application developers will not need to trim requirements so that everything can be executed in one power cycle. Instead, our language and runtime will enable longer, dependent tasks to run to completion, even if they span numerous power failures.

We envision application developers programming batteryless devices by defining a graph of tasks, connected by edges that define real time constraints, flow controls, and policy information. By separating the scheduling of tasks from the implementation details of the task, developers are free to focus on sensing outcomes and goals.

4. TOOLS

Debugging and verifying functionality in the face of flaky power presents unique challenges. Application developers need debugging tools to make the idea of perpetual batteryless sensing a reality. To address this challenge we created Ekho [1], a tool that records, and emulates energy harvesting environments. Harvested environmental energy is generally variable, scarce, and unpredictable. Using Ekho, developers can record these unpredictable Solar, RF, Kinetic, or Thermal environments, and replay these environments later. This allows developers to compare different hardware configurations, duty-cycles, sensor setups, algorithms, and much more in a realistic energy environment, pre-deployment. Using a mobile version of Ekho we developed, application designers and domain scientists can capture energy environments in difficult to reach places outside the lab, allowing for more realistic testing before a deployment.

Ekho allows us to test repeatably, and conduct rigorous experimentation with batteryless devices that was previously impossible. However, Ekho cannot guarantee that devices will work in deployment, or give line-by-line information on program and execution state. We propose a suite of extensions to Ekho that will give developers more confidence in deploying their batteryless sensors. These tools are centered around the idea of *energy aware test driven development*; specifically we look to enable breakpoint and trace based step debugging, and lay the ground work for unit testing, and code coverage metrics, for batteryless devices.

5. CONCLUSIONS

Batteryless, perpetual sensing presents unique challenges. Dealing with systems that may reboot 10-20 times a second is a difficult task. However, these devices promise to enable new realms of sensing in wildlife tracking, infrastructure, smart cities and buildings, and wearables. Before that can happen, new techniques in handling flaky power must be imagined. The central thesis of this research is that batteryless devices can support long running, sophisticated sensing applications. By constructing a specialized hardware platform, defining a language and runtime, and creating a set of tools for energy aware debugging, we can enable developers to manage the temporal, computational, and consistency difficulties caused by flaky power.

6. REFERENCES

[1] J. Hester, T. Scott, and J. Sorber. Ekho: Realistic and Repeatable Experimentation for Tiny Energy-Harvesting Sensors. In *Proc. 12th ACM Conf. Embedded Network Sensor Systems (SenSys'14)*, pages 1–15, Memphis, TN, USA, Nov. 2014. ACM.

[2] J. Hester, L. Sitanayah, and J. Sorber. Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors. In *Proc. 13th ACM Conf. Embedded Network Sensor Systems (SenSys'15)*, Seoul, Korea, Nov. 2015. ACM.

[3] A. Rahmati, M. Salajegheh, D. E. Holcomb, J. Sorber, W. P. Burleson, and K. Fu. Tardis: Time and remanence decay in sram to implement secure protocols on embedded devices without clocks. In *USENIX Security Symposium*, pages 221–236, 2012.