# Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors

Josiah Hester, Lanny Sitanayah, Jacob Sorber
School of Computing
Clemson University
{jhester, lsitana, jsorber}@clemson.edu

## ABSTRACT

Untethered sensing devices have, for decades, powered all system components (processors, sensors, actuators, etc) from a single shared energy store (battery or capacitor). When designing batteryless sensors that are powered by harvested energy, this traditional approach results in devices that charge slowly and that are more error prone, inflexible, and inefficient than they could be.

This paper presents a novel federated approach to energy storage, called UFoP, that partitions and prioritizes harvested energy automatically into multiple isolated smaller energy stores (capacitors). UFoP simplifies task scheduling, enables efficient use of components with differing voltage requirements, and produces devices that charge more quickly under identical harvesting conditions than a traditional centralized approach. We have implemented a UFoP reference design and conducted extensive experimental evaluation, including a short deployment. Our experimental results using an MSP430-based prototype show that UFoP provides as much as 10% more computational availability, and as much as four times more radio availability than the centralized approach. For all applications and energy environments evaluated, UFoP harvested 0.7-10.2% more energy than the centralized equivalent; meaning UFoP adds zero energy overhead.

## Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Microprocessor/microcomputer applications

## General Terms

Reliability, Measurement, Performance

## Keywords

Energy Harvesting; Federated Energy; Task Coupling; Capacitor; Embedded System

## 1. INTRODUCTION

Energy is the greatest single limiting factor in the design and effective operation of mobile sensors and other untethered computing devices. Reductions in power consumption and advances in energy harvesting are making long-term device deployments increasingly possible; however, harvested energy is often variable, scarce, and difficult to predict. Batteries are large and expensive. Batteries wear out, charge slowly, require special protection and charging circuitry, pose environmental risks, and fundamentally limit the lifetime and deployability of today's mobile computing devices.

These challenges have inspired a range of capacitor-based sensor devices [8, 9, 18, 20–22] that harvest energy, charge quickly, and can store only enough energy for short bursts of operation (a few seconds or even a few hundred milliseconds long). This new generation of tiny batteryless sensing devices will be cheaper, more durable, and more environmentally friendly than their battery powered predecessors. They will also have much tighter energy budgets and much more frequent power failures, two conditions that are poorly-supported by today's mobile hardware platforms, especially when it comes to storing and managing energy.

Traditional mobile devices store energy in a single common energy store (battery or capacitor) that is used to power all system components (*e.g.,* processors, sensors, radios and other peripherals) an approach that is simple to design and works well for devices with large batteries; however, when energy budgets are tight and failures frequent, each component's energy usage can significantly impact the availability of other components. For example, reading from a sensor may impact the device's ability to do computation. Power hungry operations, like transmitting a radio message, may cause the device to lose power entirely, becoming unavailable until the device can be recharged. Reasoning correctly about peripherals, consequent power consumption, and application priorities requires computational resources (for modeling and prediction) that transiently-powered devices cannot afford.

In this paper, we propose a new federated approach to storing harvested energy that relaxes the coupling between a tiny, intermittently powered device's individual components (or subsystems). Our approach, called UFoP (United Federation of Peripherals), uses individual per-peripheral energy stores and low-power control circuitry to isolate and prioritize individual peripherals. Federating energy storage allows power-hungry operations to proceed without sacrificing the device's immediate ability to use other peripherals, gather new data, process incoming data, and respond to incoming stimuli. This approach also simplifies the task of programming energy-aware logic—effectively replacing complex modeling of analog circuit behaviors with simple binary decisions based on whether or not a peripheral is available.

**Listing 1: A conservative example program**

```
when timer fires do { // Once every 1ms
  if can_sense_store_and_send? {
        while (!sample_buffer_full?) {
            collect_sample()
            sleep(1ms)
        }
        compute_and_store_mean()
        transmit_recent_means()
  }
    sleep()
}
```

**Listing 2: A more optimistic program**

```
when timer fires do { // Once every 1ms
  if can_sense? {
    collect_sample()
  }
  if sample_buffer_full? {
    compute_and_store_mean()
  }
  if can_send? && has_stored_means? {
    transmit_recent_means()
    }
    sleep()
}
```

Our initial implementation of the UFoP approach uses ultra-low-power comparators to control and prioritize the charging of individual energy stores, where the microcontroller gets the first priority. The main capacitor charges until it reaches 2.7 V, then the microcontroller turns on. When the input voltage reaches 3.1 V, the peripheral capacitors charge. In our experiments, we found that programs that use UFoP as the energy backbone had as much as 10% more computational availability, and as much as four times more radio availability than the centralized approach. Using UFoP, programs become more resilient, reducing low voltage events dramatically. Additionally, programs that use UFoP harvested more energy for all energy environments evaluated than programs using the traditional centralized energy storage approach.

While distributed energy storage has been employed in large-scale power systems [3], and hybrid storage systems have been used to improve the efficiency of battery charging [12, 15, 16], UFoP is, to the best of our knowledge, the first embedded system to employ federated energy storage, in a general way, to simplify management of individual system components.

## 2. CENTRALIZED ENERGY STORAGE

For half of a century, computing devices have used a centralized approach to power system components (processors, sensors, radios and other peripherals)—an approach that has reduced both device size and cost, and that, until now, has had no significant drawbacks. Whether power is supplied by a dedicated connection to wired infrastructure or by a battery that is regularly recharged, processors, memories, and peripherals all use a shared power supply and the nearly-universal assumption that power is unlimited and stable. System designers have, at times, made efforts to reduce power consumption and extend battery life, but they rarely consider whether the program's next action may impair the device's ability to perform additional functions.

However, when designing applications for batteryless and other transiently-powered mobile devices, supplying power to all components using a single centralized capacitor or battery reduces the system's flexibility and complicates programmer decision-making. The following challenges must be carefully considered by a system designer before developing an application using centralized energy storage:

**Capacitor tuning:** Capacitor size is a critical factor that defines how an energy-harvesting batteryless device will operate. Smaller capacitors charge quickly, but may not be able to store enough energy for more power-hungry tasks. Larger capacitors can store more energy, supporting longer bursts of computation and more power-intensive operations. Larger capacitors also charge more slowly, incur longer power outages, and waste more energy (leakage). A system designer can maximize device availability and up-time by selecting a capacitor that is just large enough to support the operations that the application needs to perform. When an application performs both energy-efficient tasks (*i.e.,* sampling lightweight sensors or performing simple data processing) and energy-intensive tasks (*i.e.,* wireless data transmissions), a centralized energy store must be large enough to support all operations—both heavy- and light-weight operations.

**Task coupling:** A common storage capacitor also produces a tight coupling between program tasks and system peripherals. Sampling a sensor, for example, will consume energy and may either leave insufficient energy for subsequent tasks or may cause the supply voltage to drop low enough that the device loses power altogether, and subsequent tasks must wait until more energy becomes available.

In order to illustrate these challenges, let's consider two simple programs described in Listings 1 and 2. Both programs gather sensor readings until a buffer is full, compute the mean of the readings, and wirelessly transmit a fixed number of recently computed means[1]. The first program (Listing 1) conservatively sleeps until it has harvested enough energy to an entire application cycle (collect, process, and transmit samples). The second (Listing 2) waits only until it has enough energy to complete the next task and then proceeds optimistically, assuming that enough energy will be harvested in the future to complete subsequent tasks.

Power failures are a constant concern for batteryless devices. When power failures do occur, computational state can be saved efficiently to nonvolatile memory (like FRAM) in case a power failure occurs between tasks or before a task completes. These checkpoints can be included explicitly by application developers or inserted automatically, using a system like Mementos [17]. Checkpoints allow some tasks to be resumed after a power failure; however, other tasks, like sampling a sensor or transmitting a radio packet, cannot be easily resumed due to timing and hardware constraints.

Consequently, the batteryless device that implements these example programs will need to determine, at runtime, when enough energy is stored on its capacitor to perform each desired task safely. A capacitor's stored energy can be estimated by measuring its voltage, and tasks can be safely run when that voltage exceeds a predetermined threshold ($V_{safe}$), which can be determined empirically or analytically based on the capacitance used, the time the task requires to complete ($t_{task}$), and the power needed while the task is executing. Figure 1 illustrates how this threshold is determined. If $V_{safe}$ is chosen correctly, starting the task when the capacitor's voltage is higher than $V_{safe}$ ensures that the task will always complete before the capacitor discharges to $V_{fail}$—the point at which essential

[1]A small number of previous means are transmitted to provide some redundancy in case a packet is lost.

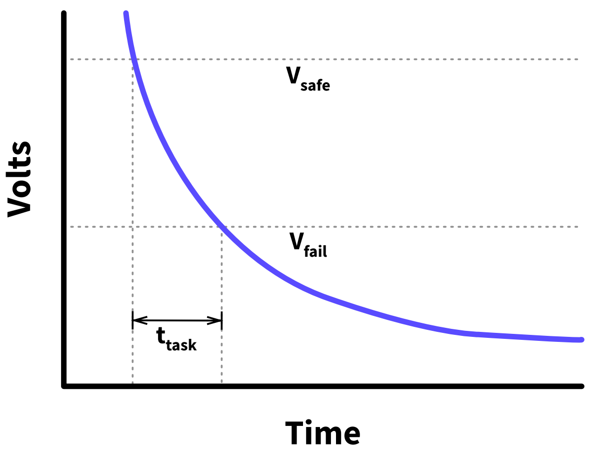

**Figure 1:** During task execution, capacitor discharge is sufficiently predictable to determine the voltage at which it is safe to begin executing it, in order to ensure that it will complete before a power failure.

hardware components—MCU, memory, sensors, radios, or other peripherals—turn off or become unusable.

Some tasks within a single application may take longer or consume more power than others, so a separate threshold, $V_{safe}$, will be needed for each individual task. System designers may optimistically try to execute a task before the safe threshold is reached, in the hope that future harvested energy will be sufficient to finish the task successfully. When energy is abundant, this gamble may allow the device to produce results more quickly. When harvested energy is insufficient, sub-threshold task execution will result in wasted effort and may result in avoidable power failures.

Software federation of the energy supply as described above can lead to close coupling of components. The top plot in Figure 2 shows the capacitor and microcontroller voltages over time for a solar-powered batteryless sensor node that gathers sensor data and transmits it wirelessly to a base station (see Listing 1). In this scenario, a control circuit waits to turn on the microcontroller until the capacitor charges to a voltage (*i.e.,* around 2.7 V) sufficient to initialize the processor and accomplish some computation, the control circuit then turns off the microcontroller when the voltage drops below 2 V[2] (when processing becomes unreliable for most microcontrollers). The chosen capacitor is large enough (195 μF) to power both data collection and radio transmission on a single charge, and the application waits until the voltage charges up high enough to ensure that the transmission completes most of the time.

In this scenario, each radio transmission causes the MCU voltage to drop dangerously close to (and occasionally cross) the 2 V point where the MCU becomes unstable. The larger capacitor also charges slowly, and leaks, meaning that time and energy is lost.

## 3. FEDERATING ENERGY

In light of the shortcomings of centralized energy storage, this paper argues for a different approach which stores harvested energy in multiple independent small capacitors, one for core processing functionality, and another for each peripheral. We call this federated approach UFoP (United Federation of Peripherals).

[2]The control circuit's hysteresis is adjustable. The thresholds used are those we have empirically found to work well, in practice.

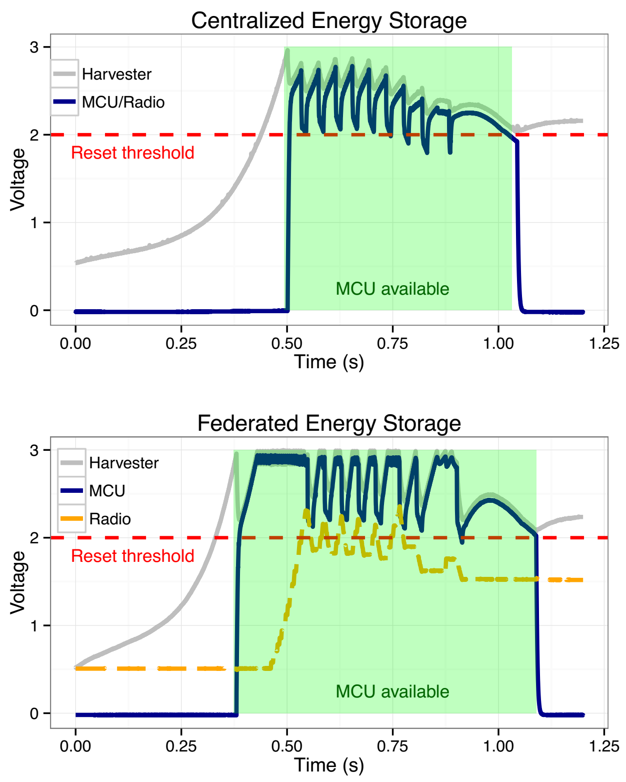

**Figure 2:** This figure shows a "sense-and-send" application using traditional centralized and our proposed federated energy storage approaches. Energy federation allows lightweight functionality (sensing and data processing in this example) to be available sooner.

By allowing the microcontroller, sensors, and radio to function independently, UFoP provides the following key benefits:

**Useful work starts sooner:** While some approaches have tried to mask volatility of the energy supply in software [2], the reality of capacitor based devices is that a smaller capacitor charges to an arbitrary voltage, faster than a larger one. A centralized approach may use software to mask volatility of the energy supply in order to simplify task management, but the energy store will still be unusable until it meets the necessary voltage for components to be turned on. By federating the energy storage, smaller capacitors charge more quickly, allowing lightweight tasks (some sensors and microcontroller operations) to be available while larger capacitors for radios and other power-hungry peripherals charge up. This can be seen in Figure 2 where the MCU becomes available hundreds of milliseconds before the centralized version.

**Fewer power failures:** Isolating each per-component capacitor prevents a power-hungry component from jeopardizing the whole system. For example, when the radio drains its dedicated capacitor's power to transmit messages, the device retains the ability (at least in the short term) to gather and process new data. UFoP prioritizes the charging of individual energy stores. In our current implementation, the microcontroller gets the first priority, while the priorities of sensors, radios, and other peripherals can be configured to suit individual applications. For example, the tight coupling present in the centralized approach shown in Figure 2 causes the supply voltage to dip below the reset threshold, endangering the device duty cycle. With UFoP, this problem occurs much less frequently.

**Simpler application decisions:** Each peripheral is available for use as soon as its capacitor is charged, and can be used independently of the charge state of other peripherals. When a peripheral is used and depletes its own energy, it becomes unavailable until it is recharged. This allows UFoP to "save up" energy for power-hungry tasks, like short bursts of radio communication, while allowing data collection and processing to continue. When using centralized storage, application decisions can be complicated—requiring designers to reason about the aggregate impact of multiple peripherals and tasks on a single capacitor. In contrast, a UFoP device can determine whether it can afford to use two peripherals (*e.g.,* a sensor and a radio), by simply checking their individual voltages.

**Increased flexibility:** UFoP devices provide more flexibility when combining peripherals with different energy requirements. For example, consider a sensor node that combines an MSP430 MCU as its computing core, a low-power sensor, and a more power-hungry radio. A larger capacitor will be needed in order to support the radio, which will charge much more slowly than a smaller capacitor that might be sufficient to support the other components for short bursts of operation. In a centralized energy architecture, radio transmissions would deplete the large common capacitor and may render the entire node unavailable while it recharges. In a UFoP device, the small capacitors dedicated to the core and sensor would charge more quickly, allowing the application to continue gathering and processing data, while waiting for the larger radio capacitor to charge. In both cases, data would be sent at roughly the same rate, but using UFoP the application would have more flexibility in deciding what to send.

**Lower energy consumption:** Not all system components require the same voltages to operate. UFoP allows individual component capacitors to be charged to the voltage required by that component, which often results in lower operating voltages and reduced per-component energy consumption.

**Harvest more energy:** When using UFoP, the energy harvester's voltage rises quickly (as the small microcontroller capacitor charges), but increases in harvester voltage is slow as power is diverted to charge peripherals. When UFoP's thresholds are set appropriately, the device spends more time in its most efficient voltage range and harvests more energy than the centralized equivalent.

## 3.1 UFoP Reference Design

Figure 3 shows a UFoP system that is integrated with two commonly used peripherals in sensing applications (*i.e.,* a sensor and a radio). The system consists of four main components: an energy harvester, charging controller, peripheral controller, and peripherals. The energy harvesting device harvests ambient energy and stores it in the first-stage capacitor. The charge controller is responsible for turning on and off the microcontroller and charging an array of peripheral capacitors. The peripheral controller turns on and off peripherals (sensor and radio), which are only available when their capacitors are charged.

**Energy Harvesting:** The UFoP system is powered by ambient sources. The energy harvester converts free energy from the environment (*e.g.,* solar, thermal, radio frequency (RF), and kinetic energy) into electrical energy (DC), which the harvester supplies to the rest of the system. From the harvester, the current flows to charge the first-stage capacitor that powers the microcontroller.

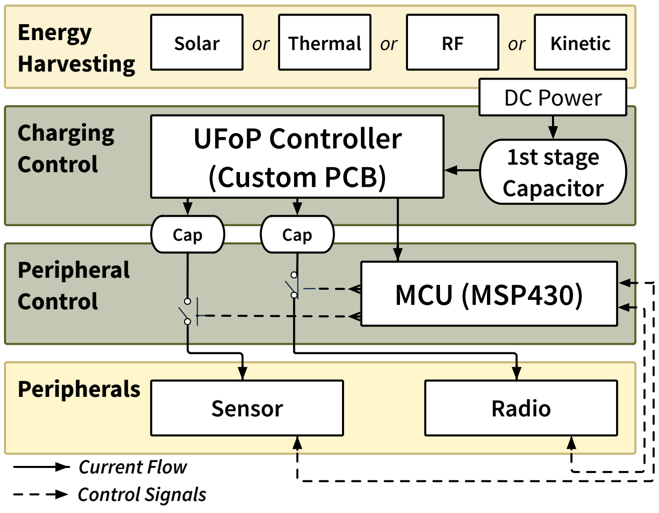

**Figure 3:** Overview of UFoP when integrated with a set of commonly used peripherals; a sensor, and a radio. A UFoP system is made up of four components: an energy harvester, charging controller, peripheral controller, and peripherals. The charge controller manages the charging of an array of capacitors (drawing from the first-stage capacitor) as well as turning on and off the MCU. The peripheral controller (an MSP430FR5739 in the current prototype) gates power to the peripherals, allowing peripherals to be completely off when not in use. Peripherals and the MCU communicate independently of the charging controller.

**Charging Control:** UFoP uses low-power control circuitry as charge controller to control and prioritize the charging of an array of capacitors as well as turning on and off the microcontroller. UFoP is designed with the microcontroller as a non-negotiable first power priority. A sensor without a microcontroller could not process data, and a radio without a microcontroller would have no signal to transmit. From an implementation perspective, this is ideal since the microcontroller can then be used to control the power flow to the peripherals. As sensors and radios can provide functionality regardless of the operation of other peripherals, subsequent priority values can be assigned based on system requirements. In our reference design, the first-stage capacitor charges until it reaches 2.7 V, then the microcontroller turns on. When the input voltage reaches 3.1 V, the current starts to flow into the peripheral capacitors.

**Peripheral Control:** The peripheral controller in the UFoP system can be any ultra-low-power microcontroller, like the FRAM based MSP430 processors. In our reference design, we use the MSP430FR5739 as peripheral controller; which only draws 81.4 $\mu$A/MHz in active mode. When the first-stage capacitor capacitor charges up to 2.7 V, the microcontroller turns on. The microcontroller gates power to the peripherals, allowing peripherals to be completely off when not in use. This control communication is independent from the charging control scheme. The switches in the peripheral controller are designed to open (disconnect) when the microcontroller loses its power and turns off, *i.e.,* when the main capacitor's voltage falls below 1.8 V. This design satisfies the MSP430FR5739 supply voltage requirements, *i.e.,* 1.8 V to 3.6 V. We built a thin software layer to manage the ADC polling, timers, and interrupt wake ups, for the MSP430 line of microcontrollers. This layer can easily be ported to other platforms, as the components and software practices are common among embedded systems.

**Peripherals:** Two of the most commonly used peripherals in sensing applications are sensor and radio. The type of the peripherals and the tasks they perform determine the size of the capacitors used. When an application uses a lightweight sensor and a radio, the size of the capacitor for the sensor should be a lot smaller than that for the radio. If the application requires intensive data transmission, the radio's capacitor size must be large enough to support the tasks. In the UFoP system, the peripherals are only available when their dedicated capacitors are charged.

## 3.2 Application Development Simplification

Federating energy storage changes how sensing applications are built. From the hardware point of view, sizing several capacitors appropriately to peripherals is much easier than sizing one capacitor to multiple peripherals and a microcontroller that could have different supply voltage requirements. For example, the MSP430 microcontroller works within the 1.8 – 3.6 V range, the CC2500 radio (a very common 2.4 GHz radio) works from 2.0 V to 3.9 V, and a humidity sensor used in our greenhouse monitoring application works from 3 V to 5 V. This combination makes it difficult for application developers to size a single capacitor and determine duty cycle. In this example, the radio and microcontroller could potentially never come online while waiting for the voltage to be sufficient for the humidity sensor to function. UFoP allows application developers to simplify this by sizing and dedicating each capacitor at a specific voltage to each peripheral.

From the software development point of view, UFoP allows a duty cycle when energy is scarce, and when it is abundant. That means UFoP lets the duty cycle scale up or down without hurting the average duty cycle performance. For example, in a classic "sense-and-send" application, the device can use a more powerful sensor when there is an abundance of energy, but performs only computations if there is very little energy. With UFoP, programmers have a more deterministic view of energy and task scheduling, allowing them to make better informed decisions during application development. In addition, UFoP simplifies applications by eliminating the need for complicated algorithms to predict when the peripherals become available.

## 4. IMPLEMENTATION

We have implemented a UFoP reference design on a custom printed circuit board. The prototype employs a variety of different hardware components. Two nano-power comparators are used per peripheral, that control the charging, and discharging of the peripheral capacitor. The current prototype supports two peripherals. An ICL7665 or MIC841 voltage monitor is used to monitor the first-stage capacitor, this monitor has a built in reference, and resistor defined hysteresis. The settable hysteresis allows a broad operating range for the microcontroller. Each of the comparators has a resistor divider defined trip point; the trip points and hysteresis are set in such a way that the microcontroller will always be on if the peripherals are charging.

Because UFoP is a hardware addition, some note must be taken of its size and cost. The current UFoP prototype measures 37.0 mm by 15.2 mm, with a low profile. The total cost of the prototype bill of materials, including all components, and PCB from a batch PCB supplier like OSH Park, amounts to less than \$20 per device, further development could easily lower this cost. If the current prototype was produced at scales of a 1000, the price drops to less than \$2 per device. Most of this cost would disappear in a custom silicon solution, which would also reduce the size and component count for a deployed sensor.

In addition to building the UFoP prototype, multiple systems were built or used in the evaluation. We used the energy environment emulator Ekho [10] to record and replay solar energy harvesting environments, as well as RF energy environments. Solar environments were generated by a programmatically controlled headlight focused on a solar panel. RF environments were generated from the harvester of a UMich Moo [24], which harvested RFID energy from the UHF band created by an Impinj Speedway RFID Reader. IV-surfaces were created by moving the reader back and forth across the front of the Umich Moo. The recorded IV-surfaces were later replayed in the lab using Ekho; which provided a realistic energy harvesting environment to test UFoP with. We plan to make all IV-surfaces we recorded freely available online at publication time. We also used an NI USB-6356 [11] and Measurement Computing USB-201 [4] data acquisition device (DAQ) for voltage and current measurements, as well as event counting. For all applications, we used MSP430FR5739 Launchpads as the main processing device.

We translated the programs described in Listings 1 and 2 into embedded C, running on the MSP430FR5739. Each program has two variants, one that is meant to run with UFoP functioning as the energy backbone, and one that runs with the traditional centralized energy approach. The federated and centralized variants are intentionally similar for both programs; the federated versions differ in that they dedicate an ADC per peripheral to monitor the energy storage levels of the radio and accelerometer supply voltages, in addition to the MCU energy storage levels. Both of these programs attempt to federate energy storage, one using UFoP, and one in software. We used these programs to evaluate the effectiveness of the federated energy approach in terms of availability, resiliency, and energy harvesting efficiency. We also developed federated and centralized versions of a greenhouse monitoring application, which is described in Section 6. These are similar in function to the above, but use a different set of peripherals. A basestation program was also created to listen for and log sensor readings during the greenhouse monitoring deployment.

All hardware designs, I–V surfaces, and software will be made freely available online at publication time.

## 5. EVALUATION

In this section, we evaluate the performance of sensing applications that use our UFoP reference design as an energy backbone. Specifically, we examine how federated and centralized variants of sense-and-send behave in solar and RF energy environments. We compare these two approaches and measure them in terms of availability, resiliency, and energy harvesting performance.

In our experiments, we found that programs that use UFoP as the energy backbone had as much as 10% more MCU availability, and as much as four times more radio availability than the centralized competitor. Using UFoP, programs become more resilient, reducing failures by 4.5*x* in some cases. Additionally, programs that use UFoP harvested 0.7-10.2% more energy, for all energy environments evaluated, than programs using a centralized energy storage approach; meaning UFoP adds zero energy overhead.

## 5.1 Methodology

To evaluate UFoP, we consider multiple programs, in a variety of energy environments, with the same hardware and peripherals, but interchanging the centralized and federated approach to energy storage. We use the following experimental setup to evaluate how UFoP contributes to the performance, in terms of availability, resilience, and energy efficiency of a tiny, capacitor-powered sensing system.

| | | Availability | | | | | |
|---|---|---|---|---|---|---|---|
| | | UFoP | | | Centralized | | |
| **Program** | **I–V Surface** | *MCU (%)* | *Radio (%)* | *Accel (%)* | *MCU (%)* | *Radio (%)* | *Accel (%)* |
| Optimistic | Solar | 57.08 | 7.61 | 18.22 | 46.97 | 6.36 | 17.67 |
| | RF High Energy | 81.85 | 4.55 | 10.45 | 76.89 | 1.18 | 25.29 |
| | RF Low Energy | 86.58 | 1.23 | 9.43 | 75.69 | 0.64 | 20.93 |
| Conservative | Solar | 54.75 | 8.14 | 13.67 | 46.64 | 8.10 | 13.40 |
| | RF High Energy | 68.11 | 7.99 | 9.59 | 80.69 | 3.77 | 12.00 |
| | RF Low Energy | 74.23 | 4.67 | 5.70 | 74.97 | 2.66 | 8.09 |

**Table 1:** This table shows the percentage of time over the entire I–V surface that the peripherals and MCU were available, for both UFoP and centralized. For these results, the program described in Listing 2 (optimistic) and the program described in Listing 1 (conservative) were used, with the $V_{safe}$ thresholds (as in, no radio transmission or power failures). For all three surfaces, using UFoP increases the availability of both the MCU (for computation) and the radio (for communication and data sending). UFoP does especially well on RF surfaces, where energy is scarce, since it does not have to wait to charge a much larger capacitor before it begins computation.

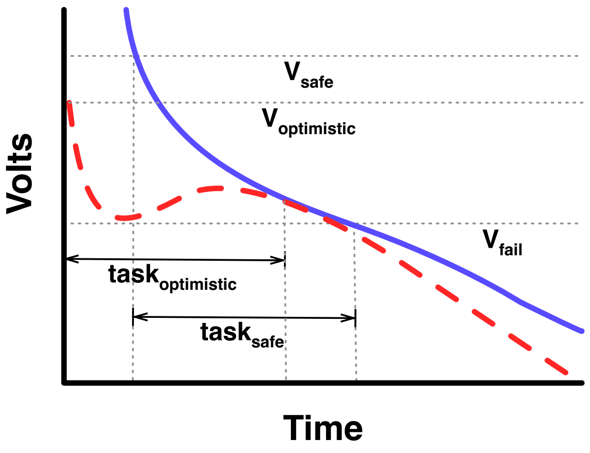

**Figure 4:** For our evaluation, two sets of thresholds were determined for each program variant. The first threshold, $V_{safe}$ (the blue discharge curve), is set such that if no new energy is harvested, tasks started at this threshold are guaranteed to complete. The second threshold, $V_{optimistic}$ (represented by the red discharge curve), optimistically assumes new energy will be harvested to replenish the task capacitor during task execution.

**Programs:** We use the sense-and-send program variants described in Section 2 to provide points of comparison between UFoP and the centralized approach. The program described in Listing 2 we refer to as "optimistic" sense-and-send, the program described in Listing 1 we refer to as "conservative" sense-and-send. The program using UFoP monitors capacitor voltages, and gates energy flow to peripherals as needed. The centralized program attempts to federate energy storage in software, allocating from its single energy store when it becomes available at the required voltage.

**Thresholds:** Each program has certain voltage thresholds ($V_{safe}$) where the radio, sensor, and MCU turns on, as described previously in Figure 1. This threshold is the voltage on the capacitor that signifies there is enough energy to complete a task or set of tasks, such as sending a data packet over the radio. These thresholds are set in software for the radio and sensor, and hardware for the MCU. Thresholds are set differently for centralized, since the single capacitor must store enough energy to accomplish all tasks. We have two sets of voltage thresholds for each program, as shown in Figure 4. The first threshold, termed $V_{safe}$ is the level that guarantees task completion. Setting the threshold above $V_{safe}$ means a program will not get as many tasks done (but will never fail), while setting the threshold below $V_{safe}$ will mean tasks are not guaranteed to complete. The second set of voltage thresholds is termed $V_{optimistic}$; the assumption is that new energy will be harvested to replace energy being used *during the task*, therefore these thresholds are set lower. Using the $V_{optimistic}$ set of thresholds does not guarantee task completion or zero power failures. These thresholds were gathered by manually executing programs with high and low thresholds over a solar I–V surface, effectively binary searching through all possible thresholds. $V_{safe}$ and $V_{optimistic}$ thresholds were only gathered for the "optimistic" sense-and-send program. $V_{optimistic}$ thresholds were gathered for the "conservative" sense-and-send program.

**Test Devices:** Each of the applications were run on Texas Instruments MSP430FR5739 processors. These devices have low sleep currents (*i.e.,* 5.9 µA), multiple ADCs, and FRAM memory for checkpointing and data storage.

**Peripherals:** Each test device manages its own set of peripherals. Peripherals used are an MMA7361 triple-axis accelerometer, and a 315 MHz RF Transmitter (CDT-88). These peripheral types were chosen because they are ubiquitous in sensing applications.

**Voltage Monitor Reference Designs:** To compare the federated and centralized approach, we designed a supply voltage monitor, based on the ICL7665 and MIC841 (also used by our UFoP reference design) to act as the energy storage backbone for all centralized program executions. The ICL7665 or MIC841 charges a large capacitor bank, and turns on and off the MCU with hysteresis. The centralized supply voltage monitor and the UFoP reference design have the same amount of capacitance (energy storage) and many of the same components, however, the UFoP reference design federates its energy storage in hardware, while the centralized reference design federates energy sotrage in software. In the evaluation, we refer to the centralized voltage monitor device as the "centralized reference design."

**I–V surfaces:** To control the energy environment, we use an Ekho [10] device to record and emulate I–V surfaces. Ekho provides a repeatable energy environment which replaces the energy harvester as input to the energy storage approach. Without Ekho, it is very difficult to control for the energy environment in experiments. We recorded three eight-second I–V surfaces for our evaluation. The first surface was recorded from a solar panel harvesting energy from

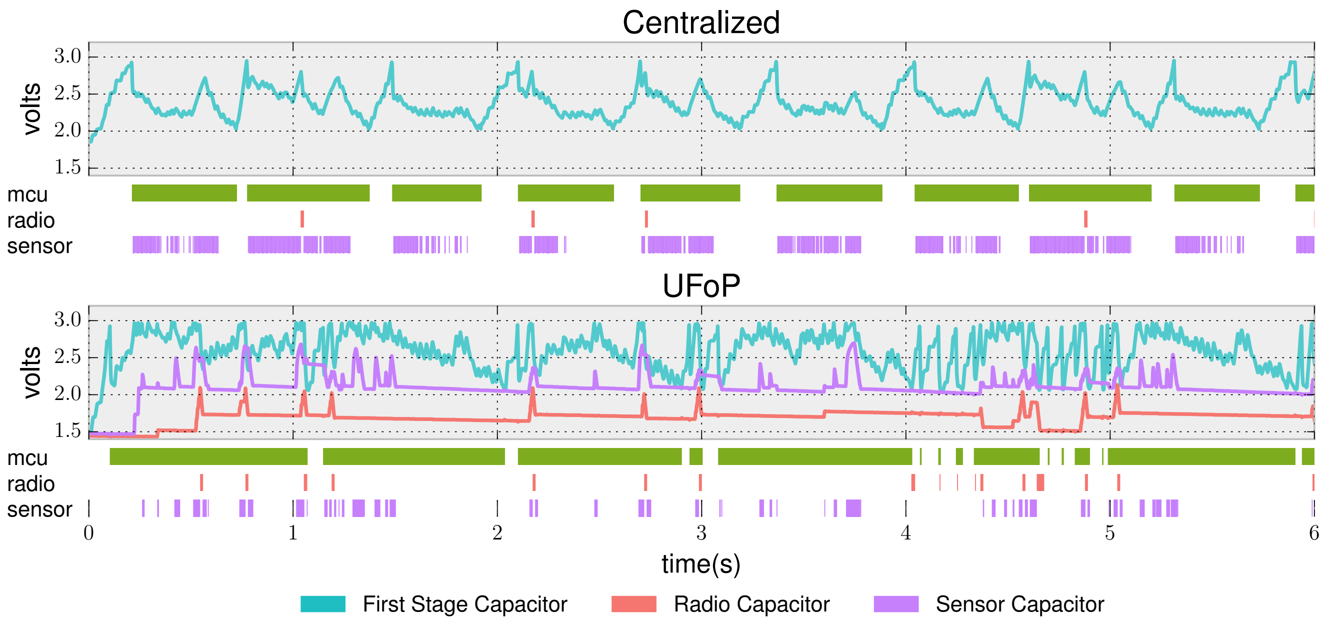

**Figure 5:** This figure compares the availability of the MCU, sensor, and radio when an MSP430FR5739 running the program described in Listing 2 executes across the I–V surface recorded from the RFID reader. The program was run multiple times using either the UFoP reference design or the centralized reference design, and the $V_{safe}$ thresholds. Applications that use UFoP have significantly more MCU on-time, allowing more valuable computation, as well as significantly more sends over the radio. Because the radio can work with a lower supply voltage than the MCU, UFoP allows for more transmissions since the energy store is decoupled from the MCU. Sensor readings, while not as frequent as with centralized, are dispersed more evenly in time.

a car headlamp that turns on and off four times, creating a sinusoidal solar surface. This means the device under test goes through the entire life cycle; charge, deplete, recharge. The second and third I–V surfaces were recorded from the RF energy harvester on the Umich Moo, while it harvested from an Impinj Speedway RFID reader. The reader antenna was waved across the Moo at two distances to produce two surfaces, referred to as "RF High Energy" and "RF Low Energy" in the rest of the evaluation.

Taking these programs, thresholds, test devices, peripherals, supply voltage monitors, and I–V surfaces, we can assemble an experimental setup that will allow us to make fair comparisons between federated and centralized energy storage. By running each program variant (federated or centralized) in each recorded energy environment, we can attempt to answer these questions about UFoP:

- Does federating energy storage provide more sensing and computational availability? (5.2)
- Does federating energy storage make applications more resilient? (5.3)
- What is the effect of federating energy on energy harvesting efficiency as compared to the centralized approach? (5.4)
- What is the overhead of the federated energy approach? (5.5)

## 5.2 Availability

The percentage of availability of the MCU for computation, and peripherals for sensing or sending over an entire duty cycle, is a critical metric of evaluating performance of tiny energy harvesting systems. In this section, we evaluate the availability of both our test programs, when running on our UFoP reference design and the centralized reference design. We execute each of these programs, with both energy storage reference designs, ten times each, on all three of our recorded I–V surfaces. All of the optimistic sense-and-send programs use the $V_{safe}$ thresholds, while the conservative sense-and-send programs use slightly "optimistic" thresholds. Using the Measurement Computing USB-201 [4] data acquisition device (DAQ), we recorded the voltage levels of all capacitors, the supply voltage of the MSP430FR5739 (the MCU), and the on and off times of the radio and accelerometer peripherals.

One set of program executions is shown in Figure 5. This figure shows the optimistic sense-and-send program executing across a surface generated by an RFID reader swiping over a Umich Moo. The three activity bars below each plot show when the MCU, the radio, and the accelerometer were in use. For this RF surface, UFoP provides more MCU on-time, and more radio transmissions. Because UFoP allows the application to use peripherals that do not have to exist at the MCU supply voltage, the UFoP program makes use of the lower voltage threshold of the radio to get extra work done in a low energy environment. Additionally, UFoP charges its capacitors faster, meaning that the MCU turns on sooner than the centralized version, this is shown in the bottom portion of Figure 5.

The results of all availability experiments are shown in Table 1. The percentage of time each component was being used is listed for both the centralized and federated approaches. For all cases, using UFoP shows improvement in availability. The most dramatic increase comes when using UFoP with low energy environments and peripherals that don't match the MCU supply voltage.

## 5.3 Resiliency

In this section, we evaluate the resiliency of our programs when using the UFoP reference design and the centralized reference de-

| | Resiliency | | | | | |
|---|---|---|---|---|---|---|
| | UFoP | | | Centralized | | |
| **Program** | *-Threshold* | *Low Voltage* | *Tx Fails* | *-Threshold* | *Low Voltage* | *Tx Fails* |
| Optimistic | -151 mV | 6.2 | 3.5% | -130 mV | 28.0 | 9.8% |
| Conservative | -93 mV | 5.1 | 5.2% | -115 mV | 34.6 | 16.3% |

**Table 2:** This table shows the effect of over-estimating the harvestable energy. In the table, *-Threshold* is the voltage below the the $V_{safe}$ threshold that was set when running programs for resiliency experiments. This threshold is used to determine when to turn on peripherals for the respective program listed on the left. These results are from execution over the solar I–V surface. The number of times the voltage on the microcontroller went below the minimum supply voltage and the percentage of radio transmission failures (because of MCU reset or peripheral reset) are shown. Poorly choosing voltage thresholds does not have as severe an effect when using UFoP, as with the traditional centralized approach.

sign. Resiliency is the measure of how tolerant an application is to voltage threshold miscalculation, task incompletion, and power failures (of the MCU or peripherals). While most application programmers try to get the equivalent of the $V_{safe}$ threshold described in Section 5.1 to ensure no failures, it is very easy to miscalculate the required energy budget for a specific task, especially when it comes from a single supply. Being overly optimistic about potential energy to be harvested can result in failed radio transmission, corrupted memory, and low voltage events. A low voltage event, where the MCU voltage drops below the minimum supply voltage, does not necessarily mean the microcontroller is reset, but once it happens, the MCU begins to draw more current, memory usually becomes unwritable, and at worst the Supply Voltage Supervisor will trigger a brown out. Therefore it is a state best to avoid if possible.

To evaluate resiliency we executed both of our programs, using either of our energy storage reference designs, ten times each, on our recorded solar I–V surface. We lowered the turn-on voltage of the radio from the $V_{safe}$ threshold by a percentage, to see what effect this optimism over energy harvesting would have. Since we did not have $V_{safe}$ thresholds for the conservative sense-and-send program, we chose thresholds that ensured zero radio transmission failures or resets over the solar surface. Since these thresholds were not matched like the $V_{safe}$ thresholds, the conservative results are illustrative of the effect of over estimating your energy harvesting. Using the DAQ, we recorded the voltage levels of all capacitors, the supply voltage of the MSP430FR5739 (the MCU), and the on and off times of the radio, and accelerometer peripherals. Using this data, we gathered for each execution, the number of times a low voltage event occurred and the percentage of radio transmission that failed.

Table 2 shows the results of this experiment. For the optimistic sense-and-send program, the lowered thresholds result in a dramatic increase in low voltage events, as well as a nearly 10% increase in transmission failures for centralized programs. The optimistic sense-and-send program that ran on the UFoP reference design had a much smaller failure rate. Power failures and low voltage events are inevitable for capacitor based sensing. UFoP reduces the number of low voltage events and failures by prioritizing the MCU and separating the peripherals energy storage. Using UFoP, the consequences of miscalculation of the energy harvesting potential of a future deployment environment becomes much less catastrophic.

## 5.4 Energy Harvesting Efficiency

Energy harvesters such as solar panels, piezoelectric ceramics, and thermal generators, do not supply a stable voltage to a sensor. The voltage on the harvester changes in response to the current draw of the sensor or vice-versa. This relationship between harvesting current and supply voltage can be described by an I–V curve. Energy harvesting over time can be described by a sequence of I–V curves; an I–V surface. Every I–V curve has a maximum power point (MPP) where the most energy can be harvested. Many systems attempt to track this point to increase energy harvesting efficiency. UFoP does not try to track the MPP, but because UFoP charges faster and keeps a more stable supply voltage, for some I–V surfaces, UFoP may harvest more energy by being closer to the MPP. This set of experiments seeks to compare the energy harvesting efficiency of the centralized and federated approach to energy storage, by observing the path that each approach traces across the same I–V surface. By keeping the same sensor combination, energy harvesting environment (I–V surface), and program, we can see the effect of the energy storage technique on efficiency.

The results of this experiment are shown in Table 3. For the energy harvesters used, the programs using the UFoP reference design generally harvested more energy than the centralized equivalent. UFoP will not always cause more energy to be harvested, however, if the voltage at the MPP of the particular I–V surface is close to the hardware set threshold voltage of UFoP, the stability of UFoP should provide more energy.

## 5.5 Overhead

Switching to a federated energy storage approach does come with overhead. This overhead comes from three places: 1) the addition of voltage monitoring hardware, which slightly increases the size, cost, and energy requirements, 2) the energy cost from polling voltage levels with the built-in ADC, and 3) software rewriting. We have not attempted to quantify the cost of software rewriting.

The biggest potential cost is the energy overhead; as UFoP is meant for energy harvesting systems, any energy spent on monitoring must be kept low. The active components that make up our reference design have a typical quiescent current draw of 2.7 µA. As a comparison, the centralized reference design has a quiescent draw of 1.5 µA. Despite the increase in overhead, the energy harvesting gains shown in Table 3 should offset the losses, and often give a net surplus of energy. Additionally, the larger capacitor used for the centralized approach has a larger leakage than the collection of smaller capacitors used in UFoP, causing some of the energy gains seen from using UFoP.

Most traditional centralized applications have some form of supply voltage monitoring through a dedicated ADC, interrupts on an input pin, or special hardware. UFoP has this same overhead, but multiplied by the number of peripheral capacitor voltages it has to monitor. The centralized reference design, over a one second period, expends 13.6 µJ polling, while the UFoP reference design expends 23.3 µJ over the same period. This extra energy cost can be reduced by polling fewer times, or using a low power or faster ADC. To reduce the polling overhead completely, an interrupt driven method

| | **Energy Harvesting Comparison** | | | |
|---|---|---|---|---|
| | **UFoP** | | **Centralized** | |
| **I–V surface** | *mean* (mJ) | *stddev* (mJ) | *mean* (mJ) | *stddev* (mJ) |
| Solar | 11.49 | 0.05 | 11.41 | 0.06 |
| RF High Energy | 11.00 | 0.10 | 9.98 | 0.11 |
| RF Low Energy | 9.15 | 0.12 | 8.66 | 0.10 |

**Table 3:** This table shows the amount of energy harvested by the optimistic sense-and-send program on each of the I–V surfaces, for centralized and UFoP energy storage. The peripheral turn-on voltage was set to the $V_{safe}$ threshold, such that no transmissions would fail, and no resets would occur. When using UFoP, the application harvests more energy for all I–V surfaces tested.

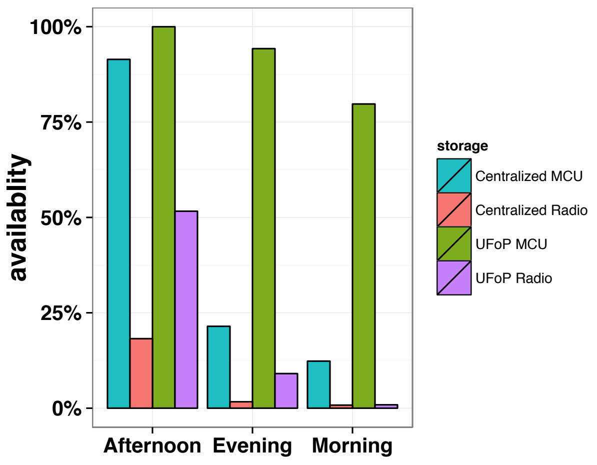

**Figure 6:** This figure shows the availability of the radio, and microcontroller, for both energy storage solutions, for the deployment. Three different time periods are shown; the afternoon, when the sun was brightest, the evening, when the sensors energy harvesting began to decrease dramatically, and the morning of the next day, when the energy harvesting begin to increase. Even though the centralized system and the UFoP system had the same amount of energy storage, the same harvester, and the same duty cycle, the UFoP sensor had more radio, and computational availability.

can be used. With this approach, UFoP can trigger a wakeup pin on the microcontroller when a peripheral capacitor has reached a logic level threshold. This form of voltage monitoring requires no extra energy beyond the sleep current of the microcontroller. If greater accuracy is required, the program can do an ADC check immediately after the interrupt wakes the microcontroller.

## 6. DEPLOYMENT

To evaluate UFoP in a real application scenario, we deployed a UFoP enabled greenhouse monitoring program for eighteen hours over two days, in a local greenhouse in late summer. We also deployed a centralized version on the same bed, as a comparison. Greenhouses waste a significant amount of water by overwatering plants. This happens because in large greenhouses, managers do not know the status of individual plant beds, and overwater to ensure plants do not die. This waste is significant for economic and sustainability reasons, as water is a finite and costly resource especially at large scales. Current commercial plant monitoring systems are little more than weather stations, these have dedicated power supplies, are too large or too expensive to be deployed densely, and can't move with the plants they monitor. Because of this, sensor data is usually very coarse, not localized, and often wrong (in the case of plants that were moved from bay to bay). Dense deployment of tiny, unobtrusive, energy harvesting, sensors has been suggested as one way to monitor large volumes of plants in a greenhouse. Equipping sensors with leaf wetness, temperature, and humidity sensors would provide all the information necessary for managers to make local decisions on water volume and plant health.

We developed an initial implementation of this greenhouse monitoring application with UFoP as an energy backbone. Two sensors derived from those discussed in Section 5.1 were built; one using our UFoP reference design, and one using the centralized reference design. Each sensor had two peripherals: a CC1101 transceiver for communication, and a resistive load that emulated a Decagon Devices LWS leaf wetness sensor, a standard sensor used in plant studies. The MSP430FR5739 was used as computational platform. Each sensor used a small solar panel (of the same model) for energy harvesting. To allow for data comparisons between the sensors, the panels were located as close as possible. Both sensors had the same amount of total energy storage in the form of SMD capacitors, and the same duty cycle. Both sensors periodically wakeup from a low power sleep mode, check the energy level(s) of the supply capacitor(s), and if high enough, sense and send a leaf wetness reading. A basestation was positioned inside the greenhouse to receive, and log, all sensor readings.

### 6.1 Choosing Capacitor Sizes

Choosing capacitors for both sensors required consideration of the duty cycle. Capacitors had to be large enough to support the peripherals, but not too large that they never charged to a high enough voltage. Before deployment, we profiled the distinct stages of the greenhouse monitoring duty cycle in terms of energy consumption, using an oscilloscope and current sensor. Each of these stages we matched to an adequately sized capacitor. The greenhouse monitoring program has three stages: "sense", "send", and "sleep". For each stage, we determined the minimum size of the capacitor, by looking at how much energy was used over time, for what voltage thresholds (we looked up voltage thresholds in device and peripheral datasheets). For example, the CC1101 transceiver when used during the "send" stage required 40ms at an average draw of 3mA to send a message, all with a supply voltage above 1.9V. A capacitor sized at 100 µF and charged to 3.2V stored enough energy at a high enough voltage to power the stage to completion.

The centralized version was equipped with the same amount of total energy storage as the UFoP enabled system, to keep the comparisons fair. Centralized thresholds were calculated in a similar fashion to UFoP. By summing the total energy required for all three stages, at the highest minimum voltage of all the components, the threshold voltage can be determined for the centralized variant.

While these capacitor size and voltage threshold calculations were done manually for this deployment, it is not hard to see how an automated system could size UFoP capacitors using simple peak detection techniques, developer or datasheet specified information about peripheral voltages, and energy harvesting information generated or gathered by an Ekho device.

## 6.2 Deployment Results and Discussion

An ARM microcontroller and light sensor was deployed with the sensors to unobtrusively record availability of the MCU, radio, and sensor over time. The amount of light on the solar panels was also recorded. Data was gathered for each sensor from 4pm, to 10am the following day. The UFoP equipped sensor outperformed the sensor with the centralized reference design in terms of MCU availability and radio availability as Figure 6 shows. The UFoP enabled sensor was able to harvest significantly more energy than the centralized version, especially during times when energy was scarce (evening and morning). In the morning, from 7-10am, the UFoP equipped sensors microcontroller was on for 79% of the time, while the centralized sensor's microcontroller was on for only 12% of the time. The amount of solar energy that was available to harvest, was not enough to charge the much larger capacitor on the centralized version, meaning that data was lost. In the evening, as the sun began to drop, the UFoP equipped system harvested enough energy for the radio to be broadcasting 9.6% of the time, while the centralized version was only able to broadcast 1.7% of the time. UFoP dramatically extended the amount of time the sensor was available compared to using a centralized energy approach.

This first implementation could be improved; in full sun both sensors were able to broadcast readings continuously, meaning that the solar panel used was too large. Greenhouse managers only need leaf wetness reports a few times an hour. By decreasing the size of the solar panel, sensors can be more densely deployed. Computational time was underutilized as well. UFoP allowed the sensor's MCU to be available even when there was very low sun, however, this computational time was not used. Future programs will use this time to calculate average leaf wetness readings, and calculate local statistics on plant status, freeing up computation on the basestation.

## 7. RELATED WORK

Currently in the literature there does not exist a federated approach that stores and manages harvested energy in hardware. However, in this section, we review state-of-the-art designs and implementations in energy harvesting and management for perpetual sensing systems.

Virtual battery [2] attempts to "virtualize" the available energy, allocating energy towards tasks. Virtual battery assumes a consistent voltage, power supply, and a known battery size. It does not consider how energy harvesting can change energy budgets. UFoP is designed to work with a volatile supply voltage, frequent power failures, unknown energy harvesting, and therefore unknown energy budget. The essential difference between these two systems is that virtual battery partitions available energy, while UFoP acts as a disruption tolerant energy manager, storing incoming harvested energy and notifying the application about changes in availability. To make a system like virtual battery work on intermittently powered devices, UFoP must first exist to hide the energy volatility. UFoP does simplify application development for intermittently powered devices in a similar way to virtual battery, in the sense that both systems inform the application what energy is available, however, UFoP manages this energy in real time and handles energy replacement. eShare [26] is another energy sharing system to balance energy supply and demand. The system's energy router consists of an array of ultra-capacitor with different levels of capacitances. Even though eShare can extend the network lifetime, this system can only be useful where power wiring is feasible.

Prometheus [12] uses two storage devices, *i.e.,* super-capacitor and lithium rechargeable battery. A solar panel charges the super-capacitor and when its voltage is higher than a threshold, it charges the battery. When the super-capacitor is exhausted, the battery is used. Heliomote [16] uses a solar panel and two AA type NiMH batteries. Energy harvesting occurs when the solar panel's output voltage is 0.7 V higher than that of the battery. When battery's voltage is higher than the solar panels, even though enough power may be available on the solar panel, a node can still draw current from the battery. Everlast [19] uses a solar panel and a super-capacitor. It charges a huge-sized super-capacitor (100 F) while tracking the Maximum Power Point (MPP) of its solar panel. AmbiMax [15] harvests energy from multiple ambient power sources (solar and wind) while performing Maximum Power Point Tracking (MPPT) on each of them. Each energy harvesting subsystem harvests energy and charges its own reservoir super-capacitors. The system is powered solely by the ambient sources when the reservoir capacitor array has a higher voltage at its terminal than a threshold. It draws power from the battery when the reservoir capacitor array's terminal voltage drops below the threshold.

Other related energy harvesting systems including TwinStar [28–30], EnHANTs [6, 7], SolarWISP [9] and the work of Yerva *et al.* [23]. TwinStar [28–30] is an add-on energy harvesting and management device, which uses ultra-capacitor as the only energy storage unit. It has a dual solar panel solution (a small boot solar panel and a big main panel to charge the ultra-capacitor). On top of the hardware, the controller maintains a high duty cycle when the voltage is high and a low duty cycle when the voltage is low. Energy-Harvesting Active Networked Tags (EnHANTs) [6, 7] can be attached to objects that are traditionally not networked, such as books. The prototypes harvest indoor light energy using custom organic solar cells, communicate and form multihop networks using ultra-low-power Ultra-Wideband Impulse Radio (UWB-IR) transceivers. In [9], the authors add solar panel (SolarWISP) to WISP (RF-powered tag) for hybrid energy harvesting. SolarWISP increases effective communication range threefold and quadruples read rate. In [23], the authors propose to add a new tier in the sensor network architecture by using energy-harvesting leaf nodes, which can communicate with battery-powered branch nodes and wall-powered trunk nodes.

Software strategies for adaptive scheduling based on the dynamic energy supply are studied in [1, 13, 27]. DEOS [27] is a dynamic energy-oriented scheduling method that dynamically adjusts the execution of tasks based on the tasks' energy consumption and the system's real-time available energy. Dewdrop [1] is a CRFID runtime that makes effective use of harvested energy. It adapts a tag's duty cycle to match the harvested power to the sensing and computation cost of tasks. In [13], the authors propose mathematical models to predict the ideal battery size and the rate of availability of harvestable energy with an assumption that the energy consumed by a node is always less than or equal to the harvested energy. Mementos [17] is a software system that transforms general-purpose programs into interruptible computations that are protected from frequent power losses by automatic, energy-aware state checkpointing. At compile time, Mementos inserts function calls that estimate available energy. At run time, Mementos predicts power losses and saves program state to nonvolatile memory.

## 8. DISCUSSION

In our experiments, UFoP was able to improve the availability, resiliency, and energy harvesting efficiency of tiny sensor devices with capacitor-based energy storage. Based on our experimental results, we believe that UFoP is a step forward for perpetual sensing, potentially making long-term deployments for mobile and other untethered computing devices possible. However, ambient energy is still scarce in deployment and energy storage capacity is terminally limited. Therefore, even though UFoP is able to relax the short-term coupling between peripherals, power management is always an important issue to consider. In this section, we discuss software approaches to federating energy, UFoP's design limitations, design alternatives, and applications.

**Software Energy Federation:** An alternative approach to using UFoP is to attempt to federate energy in software, by balancing the peripherals voltage and energy requirements with the volatile energy supply. This approach is used by all centralized programs in Section 6 and Section 5. Federating (or virtualizing, as in Virtual Battery [2]) energy in software has the advantage of being reconfigurable; the duty cycle (and energy partitions) can be changed dynamically. Additionally, no new hardware is required. However, the cost of software complexity in adding a virtual layer must be considered, especially in resource constrained systems that harvests energy. Additionally, a software approach suffers from all the shortcomings (task coupling, slow charging, reduced MCU availability,) associated with a single energy store. Hardware federation, for the same energy supply, will always provide more availability, and resiliency, than a software federated approach.

**Limitations:** The current UFoP reference design has some limitations. Peripherals are susceptible to starvation since the peripheral capacitors start charging when the input voltage in the main capacitor reaches 3.1 V. This can happen when an application requires the microcontroller to run too much computation in a very low energy environment and thus the peripheral capacitors never have the chance to charge. The centralized system has the same starvation problem. Another cause of peripheral starvation is if the microcontroller has a higher active current draw than the one we currently use (81.4 $\mu$A). To overcome this limitation, an application developer can program, through software, the microcontroller to sleep for a brief period immediately after turning on.

The current UFoP reference design trades off speed for low quiescent energy consumption in the choice of its active components. Some of the comparators switch very slowly, meaning that high current peripherals may be able to draw too much from the first stage capacitor if no new energy is being harvested. This can be solved by limiting capacitor charging with a resistor or op-amp, or trading off low quiescent current for a faster switching peripheral gate comparator.

Another limitation of the current prototype is we cannot use the peripherals in the order we need if we change the behavior of an application during runtime. This issue can be handled with dynamic priority, allowing capacitors to be charged and used by different peripherals according to the application. We leave this for future work.

**Applications:** UFoP enables low power sensing applications that use a variety of peripherals, with different energy needs and voltage requirements. UFoP is most useful for perpetual, energy harvesting systems that aggregate multiple types of sensor data. We envisage UFoP being used in applications ranging from greenhouse monitoring, low power wearable devices (humans and animals), and any computational RFID application including but not limited to, inventory management, building monitoring, activity monitoring, and infrastructure monitoring.

## 9. FUTURE WORK

Currently, peripherals charge at the same rate, with no input from the application designer. Situations could be imagined where settable priorities would be wanted. Using a resistor on the charging line after the comparator that gates the peripheral capacitor could give a form of priority; the smaller the resistor the quicker the peripheral capacitor charges. Current languages and operating systems for programming embedded sensors [5, 14, 25] assume a stable power supply and a sufficiently large battery. With UFoP, some system components will not always be available when needed and thus some tasks will need to be postponed. We are currently developing a new programming language to ease building transiently-powered sensing applications using UFoP.

## 10. CONCLUSIONS

This paper presents UFoP, the first system for capacitor-based sensors that employs a federated approach to the storage and management of harvested energy. UFoP stores harvested energy in multiple independent small capacitors, one dedicated to each peripheral. It employs a charge controller that charges the capacitors while maintaining the supply voltage of the microcontroller. With UFoP, power-hungry tasks from a radio or a heavyweight sensor will not cause low voltage events that can potentially reset the microcontroller. In our experiments, we found that programs that use UFoP as the energy backbone had as much as 10% more computational availability, and as much as four times more radio availability than the centralized approach. Using UFoP, programs become dramatically more resilient, reducing low voltage events and radio transmission failure. Additionally, programs that use UFoP harvested more energy for all energy environments evaluated than programs using the traditional centralized energy storage approach; meaning that UFoP functions with zero overhead in many cases.

## 11. ACKNOWLEDGMENTS

The authors would like to thank the members of the PERSIST lab, for helpful discussions around UFoP deployments and experimentation. We also thank our shepherd Kay Römer, and our anonymous reviewers, for their helpful comments.

This research is based upon work supported by the National Science Foundation under grants ACI-1212680 and CNS-1453607. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation

## 12. REFERENCES

[1] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: An Energy-Aware Runtime for Computational RFID. In *Proc. 8th USENIX Conf. Networked Systems Design and Implementation (NSDI'11)*, pages 197–210, Boston, MA, USA, Mar. 2011. ACM.

[2] Q. Cao, D. Fesehaye, N. Pham, Y. Sarwar, and T. Abdelzaher. Virtual Battery: An Energy Reserve Abstraction for Embedded Sensor Networks. In *Proc. 29th IEEE Real-Time Systems Symposium (RTSS'08)*, pages 1–11, Barcelona, Spain, Nov.–Dec. 2008. IEEE.

[3] J. A. Carr, J. C. Balda, and H. A. Mantooth. A Survey of Systems to Integrate Distributed Energy Resources and Energy Storage on the Utility Grid). In *Proc. IEEE Energy 2030 Conf. (ENERGY'08)*, pages 1–7, Atlanta, GA, USA, Nov. 2008. IEEE.

[4] M. Computing. Usb-201 data acquisition usb daq device 12-bit, 100 ks/s. `http://www.mccdaq.com/`

usb-data-acquisition/USB-201.aspx. [Online; accessed 30 March, 2015].

[5] A. Dunkels, B. Grönvall, and T. Voigt. Contiki—A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proc. 1st IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, Nov. 2004. IEEE Computer Society.

[6] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman. Energy Harvesting Active Networked Tags (EnHANTs) for Ubiquitous Object Networking. *IEEE Wireless Communications*, pages 18–25, Dec. 2010.

[7] M. Gorlatova, R. Margolies, J. Sarik, G. Stanje, J. Zhu, B. Vigraham, M. Szczodrak, L. Carloni, P. Kinget, I. Kymissis, and G. Zussman. Energy Harvesting Active Networked Tags (EnHANTs): Prototyping and Experimentation. Technical Report 2012-07-27, Electrical Engineering, Columbia University, New York, NY, USA, Jul. 2012.

[8] M. Gorlatova, R. Margolies, J. Sarik, G. Stanje, J. Zhu, B. Vigraham, M. Szczodrak, L. Carloni, P. Kinget, I. Kymissis, and G. Zussman. Prototyping Energy Harvesting Active Networked Tags (EnHANTs). In *Proc. 32nd IEEE Int'l Conf. Computer Communications (INFOCOM'13)*, pages 585–589, Turin, Italy, Apr. 2013. IEEE.

[9] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan. On the Limits of Effective Hybrid Micro-Energy Harvesting on Mobile CRFID Sensors. In *Proc. 8th Int'l Conf. Mobile Systems, Applications, and Services (MobiSys'10)*, pages 195–208, San Francisco, CA, USA, Jun. 2010. ACM.

[10] 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.

[11] N. Instruments. Ni x series multifunction data acquisition. http://sine.ni.com/ds/app/doc/p/id/ds-163/lang/en. [Online; accessed 11 October, 2013].

[12] X. Jiang, J. Polastre, and D. Culler. Perpetual Environmentally Powered Sensor Networks. In *Proc. 4th Int'l Symp. Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, USA, Apr. 2005. ACM.

[13] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power Management in Energy Harvesting Sensor Networks. *ACM Trans. Embedded Computing Systems (TECS)*, 6(4), Sept. 2007.

[14] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Sensor Networks. In *Ambient Intelligence*. Springer Verlag, 2004.

[15] C. Park and P. H. Chou. AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes. In *Proc. 3rd Ann. IEEE Comm. Society Conf. Sensor, Mesh and Ad Hoc Communications and Networks (SECON'06)*, pages 168–177, Reston, VA, USA, Sept. 2006. IEEE.

[16] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava. Design Considerations for Solar Energy Harvesting Wireless Embedded Systems. In *Proc. 4th Int'l Symp. Information Processing in Sensor Networks (IPSN'05)*, pages 457–462, Los Angeles, CA, USA, Apr. 2005. ACM.

[17] B. Ransford, J. Sorber, and K. Fu. Mementos: System Support for Long-Running Computation on RFID-Scale Devices. In *Proc. 16th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS'11)*, pages 159–170, Newport Beach, CA, USA, Mar. 2011. ACM.

[18] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith. Design of an RFID-Based Battery-Free Programmable Sensing Platform. *IEEE Trans. Instrumentation and Measurement*, 57(11):2608–2615, Nov. 2008.

[19] F. Simjee and P. H. Chou. Everlast: Long-life, Supercapacitor-operated Wireless Sensor Node. In *Proc. Int'l Symp. Low Power Electronics and Design (ISLPED'06)*, pages 197–202, Tegernsee, Germany, Oct. 2006. IEEE.

[20] S. Thomas, J. Teizer, and M. Reynolds. Electromagnetic Energy Harvesting for Sensing, Communication, and Actuation. In *Proc. 27th Int'l Symp. Automation and Robotics in Construction (ISARC'10)*, Bratislava, Slovakia, Jun. 2010. IAARC.

[21] Y. Yang, L. Wang, D. K. Noh, H. K. Le, and T. F. Abdelzaher. SolarStore: Enhancing Data Reliability in Solar-Powered Storage-Centric Sensor Networks. In *Proc. 7th Int'l Conf. Mobile Systems, Applications, and Services (MobiSys'09)*, pages 333–346, Krakow, Poland, Jun. 2009. ACM.

[22] D. Yeager, F. Zhang, A. Zarrasvand, N. T. George, T. Daniel, and B. P. Otis. A 9 $\mu$A, Addressable Gen2 Sensor Tag for Biosignal Acquisition. *IEEE Journal of Solid-State Circuits*, 45(10):2198–2209, Oct. 2010.

[23] L. Yerva, B. Campbell, A. Bansal, T. Schmid, and P. Dutta. Grafting Energy-Harvesting Leaves onto the Sensornet Tree. In *Proc. 11th Int'l Conf. Information Processing in Sensor Networks (IPSN'12)*, pages 197–208, Beijing, China, Apr. 2012. ACM.

[24] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational RFID and sensing platform. Technical Report UM-CS-2011-020, UMass Amherst Department of Computer Science, June 2011.

[25] R. Zhou and G. Xing. Nemo: A High-fidelity Noninvasive Power Meter System for Wireless Sensor Networks. In *Proc. 12th Int'l Conf. Information Processing in Sensor Networks (IPSN'13)*, pages 141–152, Philadelphia, USA, Apr. 2013. ACM.

[26] T. Zhu, Y. Gu, T. He, and Z. L. Zhang. eShare: A Capacitor-Driven Energy Storage and Sharing Network for Long-Term Operation. In *Proc. 8th ACM Conf. Embedded Networked Sensor Systems (SenSys'10)*, pages 239–252, Zurich, Switzerland, Nov. 2010. ACM.

[27] T. Zhu, A. Mohaisen, Y. Ping, and D. Towsley. DEOS: Dynamic Energy-Oriented Scheduling for Sustainable Wireless Sensor Networks. In *Proc. 31st Ann. IEEE Int'l Conf. Computer Communications (INFOCOM'12)*, pages 2363–2371, Orlando, Florida, US, Mar. 2012. IEEE.

[28] T. Zhu, Z. Zhong, Y. Gu, T. He, and Z. L. Zhang. Leakage-Aware Energy Synchronization for Wireless Sensor Networks. In *Proc. 7th Int'l Conf. Mobile Systems, Applications, and Services (MobiSys'09)*, pages 319–332, Krakow, Poland, Jun. 2009. ACM.

[29] T. Zhu, Z. Zhong, T. He, and Z. Zhang. Energy-Synchronized Computing for Sustainable Sensor Networks. *Ad Hoc Networks*, 11:1392–Ű1404, 2013.

[30] T. Zhu, Z. Zhong, T. He, and Z. L. Zhang. Feedback Control-Based Energy Management for Ubiquitous Sensor Networks. *IEICE Trans. Communications*, E93-B(11):2846–2854, 2010.