Getting Things Done on Computational RFIDs with Energy-Aware Checkpointing and Voltage-Aware Scheduling

Benjamin Ransford Shane Clark Mastooreh Salajegheh Kevin Fu
{ransford, ssclark, negin, kevinfu}@cs.umass.edu
Department of Computer Science, University of Massachusetts Amherst

Abstract

Computational RFIDs (CRFIDs) provide flexible, general-purpose computation on a microcontroller via energy that is harvested and stored in capacitors rather than batteries. Our contributions include a definition of CRFIDs, a framework for energy management in CRFIDs, and the preliminary design of Mementos, a medley of compile-time and run-time techniques to achieve effective forward progress of computation on CRFIDs by using energy-aware computational checkpoints and voltage-aware program reordering that maintains program semantics. Our preliminary measurements indicate that Mementos will enable CRFIDs to complete long-running computations despite constant interruptions to power.

1 Introduction

The recent advent of ultra-low-power microcontrollers is leading to an entirely new class of low-power embedded computers. Maintenance-free *computational RFIDs (CRFIDs)* enable general-purpose computation with only harvested radio frequency (RF) energy and can operate in contexts where replacing or recharging a battery is inconvenient or hazardous (e.g., implantable medical devices [8]) or where integrated circuits and small surface-mount capacitors enable economies of scale for manufacturing and miniaturization. The primary challenges to CRFIDs are (1) performing effective computation in spite of continual power interruptions that result in complete loss of computational state, and (2) effectively using energy from a continuously varying voltage supply.

Computation on CRFIDs differs from traditional general-purpose computation in several ways — posing new challenges to energy-aware computation.

Frequent power loss is the common case rather than the exception. Today, software is designed to recover from occasional power failures on PCs and sensor motes. Low-power devices such as contactless smartcards require computations to finish in a single energy lifecycle — that is, one period of energy availability or one charge-discharge cycle of the energy store. CRFIDs instead support delay-tolerant computations that can be suspended and resumed, enabling computation on a larger class of problems. Our experiments on a prototype CRFID typically enjoy less than one second of uninterrupted computation before either entering RAM retention mode or completely losing power and state. The rate of interruption is determined by capacitor size, distance from an RFID reader, and the periodicity of RF energy delivered by the reader. Since longer read ranges allow more flexible, pervasive applications and harvested radio energy drops off with the square of distance [13], there will always be a range outside which deployed CRFIDs lose power despite hardware optimizations. Thus, a key goal is to enable *forward progress* of computation in an energy-efficient manner despite interruptions to power. We define forward progress as measurable progress toward some computational goal.

Voltage-dependent instruction sequencing. The extreme resource constraints of CRFIDs lead to violations of traditional hardware-software abstractions. Not only do different instructions consume different amounts of energy, but different instructions (e.g., erasing flash memory) require different minimum voltage levels. Thus at certain times, a high supply voltage operation may unnecessarily block execution of a low supply voltage operation because of the continuously varying supply voltage.

Continuous problems for discrete power management. Unintentional underclocking leads to idly wasted energy. Devices powered by conventional batteries enjoy relatively constant voltage, but CRFIDs relying on capacitors endure rapidly fluctuating voltage — even across pairs of consecutive instructions. Dynamic voltage and frequency scaling microcontrollers provide a small set of *discrete* power saving modes (supply voltage and clock frequency combinations), but because voltage varies and adjusting power modes during each and every instruction is difficult, a single slow clock frequency is selected regardless of voltage. A CRFID's microcontroller works when its supply voltage is sufficient for the selected clock frequency. If the supply voltage is *higher* than required

for the selected frequency, then after each clock tick, the logic gates settle long before the next clock tick and the remaining time and energy is wasted [6]. Consequently, a particular instruction consumes more energy at higher voltages, posing an opportunity for clever scheduling.

Our contributions include (1) a definition of computational RFID (CRFID), (2) a framework to better understand how to optimize the energy consumption of determinate tasks on CRFIDs, and (3) the preliminary design of Mementos, a medley of static compile-time and dynamic run-time techniques to achieve effective forward progress of computation by using energy-aware computational checkpoints and voltage-aware program reordering that maintains program semantics.

1.1 Framework for Forward Progress

Because of the frequent power loss, a *checkpoint* of computational state is fundamental to the nature of computing in CRFIDs. Upon availability of harvested energy, a CRFID can make *forward progress* by restoring a checkpoint. However, checkpoints themselves consume significant energy because writes to non-volatile memory consume much more energy than computation. Thus, our approach in Mementos uses energy hints to minimize checkpoints yet guarantee forward progress of computation. A second energy-aware strategy in Mementos reorders code while maintaining program semantics so that high-voltage operations appear early in an energy lifecycle. This strategy is unique to the continuously varying voltage supplies common on CRFIDs.

1.2 Background on CRFIDs

A *computational RFID* contains at least four basic subsystems: (1) a microcontroller for general-purpose computation, (2) non-volatile storage such as flash memory, (3) a small, maintenance-free reservoir such as a surface-mount capacitor to store harvested energy, and (4) an ultra-low-power radio link. Sensor motes and computational RFIDs are both energy-constrained, but they serve different purposes and have several fundamental differences.

Energy. CRFIDs often rely on surface-mount capacitors to operate in environments where batteries are difficult or dangerous to replace or recharge. Batteries tend to maintain a relatively constant voltage until depletion. Capacitors exhibit an exponential decay over time, with the steepest loss of voltage at the beginning of discharge. Therefore, a microcontroller relying on a capacitor must tolerate an exponentially decreasing voltage.

Communication & Storage. Motes that have active radio circuitry can transmit data at will, but CRFIDs use backscatter communication — electrical modulation of impedance to change antenna reflectivity — and therefore cannot communicate autonomously. Instead, a CRFID must wait for an RFID reader to initiate both sending and receiving of information. To save energy, motes avoid unnecessary radio communication; CRFIDs avoid unnecessary writes to flash memory for the same reason.

Computation. CRFIDs must tolerate extremely bursty computation because of continual power interruptions. Thus, the microprocessor supports at least three power modes: active, sleep, and reset (loss of state). The typical active lifecycle of a CRFID is less than one second, whereas a sensor mote's active lifecycle is often measured in days or weeks.

The WISP computational RFID. The Wireless Identification and Sensing Platform (WISP) from Intel Research Seattle [3, 11] is an instance of a CRFID. An ultra-low-power TI MSP430F1232 microcontroller provides general-purpose computation (up to 8 MHz) and storage (256 bytes of RAM, roughly 8 KB of flash), and a surface-mount capacitor stores harvested RF energy.

2 The Mementos System

Mementos enables *forward progress* of computation despite continual power interruptions. Mementos combines compile-time and run-time techniques to implement energy-aware execution checkpointing and program reordering on CRFIDs. We are developing Mementos on prototype WISP CRFIDs.

2.1 Execution Checkpointing

Execution checkpointing means saving program state to non-volatile memory in case an external event causes execution to halt. In the CRFID model, fluctuating energy availability and charge leakage contribute to frequent power loss. Unlike previous execution checkpointing systems (e.g., that of Gummadi *et al.* [7] for sensor networks), Mementos assumes that failures are the common case rather than unusual events; it must therefore suspend and resume execution quickly and often.

The checkpointing system's first task is pre-assembly energy profiling of a program. Mementos reads a compiled program and a profile of the energy usage per instruction class for the target architecture (similar to Table 2). It then examines the source and destination operands of each instruction and, for each labeled program block, emits a sum that estimates the total energy required for one execution of that block.

After computing energy estimates for program blocks, Mementos inserts *trigger points* into the instruction stream at salient junctures (e.g., at loop termination tests). A trigger point is a sequence of instructions that decides at run-time whether to checkpoint. Table 1 lists the variables Mementos considers when deciding whether to checkpoint. A trigger point:

1. Measures the storage capacitor's voltage using an on-board analog-to-digital converter (ADC). The volt-

V	Voltage on the storage capacitor
C	Storage capacitor's (fixed) capacitance
E_0	Energy in storage capacitor on wake
$E_{\text{remaining}}$	Energy remaining in capacitor now
E_{program}	Compile-time estimate of total energy required for portion of program not yet completed
$\mathsf{C}_{\text{finish}}$	Instantaneous cost (energy required) of completing program
$\mathsf{C}_{\text{checkpoint}}$	Instantaneous cost (energy required) of checkpointing

Table 1: Variables Mementos uses at trigger points to decide whether to checkpoint.

age V and the capacitance C determine the energy that remains in the capacitor as $E_{\text{remaining}} = (CV^2)/2$.

2. Estimates the energy cost of finishing the program: $\mathsf{C}_{\text{finish}} \approx E_{\text{program}} - (E_0 - E_{\text{remaining}})$, where E_0 is the initial energy on wake.

3. Estimates the energy cost $\mathsf{C}_{\text{checkpoint}}$ of writing state to non-volatile storage (checkpointing).

4. Decides what to do next: if $\mathsf{C}_{\text{finish}} \leq \mathsf{C}_{\text{checkpoint}} \leq E_{\text{remaining}}$, return to the computation. If $\mathsf{C}_{\text{checkpoint}} \leq E_{\text{remaining}} < \mathsf{C}_{\text{finish}}$, checkpoint. Otherwise, save a very small amount of state to non-volatile information memory to indicate that checkpointing should be more aggressive in the device's next lifecycle.

At the beginning of run time, Mementos measures the total available energy (E_0 above). It checks a predetermined location in non-volatile memory for a restorable checkpoint. If it finds no such checkpoint, it starts the program from the beginning. If it does find a checkpoint, it uses the checkpointed information to restore the program's state, including the program counter. It then resumes execution of the program at the point where it had been suspended. Figure 1 illustrates the operation of checkpointing.

We note that energy measurement is not a zero-cost primitive operation; it involves using an on-chip ADC to measure the storage capacitor's voltage. On a WISP (Rev. 1) prototype, we measured the per-bit energy consumption of an ADC read operation to be equivalent to the per-bit energy consumption of a flash write operation.

2.2 Program Reordering

Program reordering means rearranging blocks of program code. Whereas scheduling considers how to order segments that are already separate, program reordering decomposes a single program into segments and schedules those segments. Two observations motivate our suggestion of program reordering in the CRFID model: first, if a program comprises multiple distinct parts, in some cases it is possible to automatically determine the execution order dependencies among those parts. Second, certain operations (e.g., flash memory erasure) require high supply voltage. Taking capacitor leakage into account, Mementos aims to facilitate progress by moving high-voltage operations in front of low-voltage operations in the execution order if program semantics allow.

Slack time or *underclocking* subtly affects scheduling decisions as well. Instructions take more energy when voltage is higher. Therefore, when reordering instructions Mementos must take care to select schedules that minimize the energy wasted on slack time: if two high-voltage operations require different amounts of energy at the same voltage, scheduling the less intensive operation *first* may result in lower total energy consumption.

2.3 Challenges

Resources are limited. Low-power microcontrollers provide limited resources. Mementos necessarily imposes some overhead in non-volatile memory for energy estimates, code, and checkpoints. Using Mementos is appropriate when the cost of its overhead outweighs—over time—the cost of failures.

Identifying "state" is hard. Most programs alter registers and memory during execution. A checkpoint is of no value unless it includes enough state to allow Mementos to resume the computation where it left off, but because resources are limited, checkpoints should include minimal non-state information. One conservative approach requires the programmer to instrument programs with hints that highlight important state. However, because Mementos is meant to operate automatically, and because programmers make mistakes, depending solely on programmer input is undesirable. Mementos currently takes the naïve approach, copying key portions of RAM to non-volatile memory. A future version of Mementos may instrument programs for automatic identification of state.

Inaccuracy makes deciding when to checkpoint difficult. In the simplest cases, Mementos can accurately measure available energy at each trigger point in the instruction stream and can completely checkpoint to non-volatile storage just before losing power. Sadly, applications are not likely to present the simplest cases, and this strategy is risky.

The greatest risk posed by our energy measurement and checkpointing strategy is that power loss may occur before a checkpoint has been completely written. At first, it appears that waiting as long as possible to checkpoint is the best strategy in all cases. But, paradoxically, the *worst* time to write to flash memory is at the end of a lifecycle because of declining voltage. One approach to this prob-

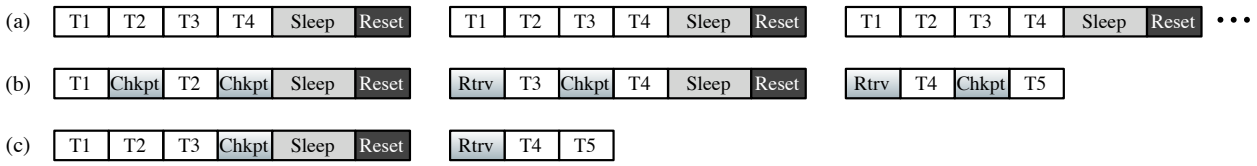

Figure 1: Checkpoints allow computation to span multiple device lifecycles. Sequences (a)–(c) illustrate the effect of different checkpointing strategies for a task T comprising equal-size subtasks T1–T5. In sequence (a), no checkpointing is done and power fails after T4; the sequence is doomed to repeat forever. Sequence (b) shows the effect of excessive checkpointing: the computation makes steady progress but takes three lifecycles to complete. Sequence (c) shows the optimal checkpointing strategy.

lem assumes that energy estimates are inaccurate and inserts trigger points earlier than the estimates recommend.

By introducing a realistic assumption about energy availability, Mementos avoids the aforementioned paradox. In realistic environments, bursty but regular RF energy puts CRFIDs in *active-sleep-active-reset* cycles rather than *active-reset* cycles; periods of activity are interrupted by periods of sleeping in which RAM is retained. Mementos can prepare a checkpoint in RAM inexpensively just before or after the sleep period. During the sleep period, the storage capacitor is replenished. When voltage becomes sufficient, the CPU wakes up and Mementos uses the high voltage to store a checkpoint to non-volatile memory. We call this technique *Flash Forward*.

2.4 Example: modexp

To demonstrate the utility of Mementos, we implemented a standard iterative modular exponentiation algorithm on a WISP (Rev. 1) and instrumented it to raise a GPIO pin on the WISP's CPU upon completion. The toy algorithm repeatedly halves a 32-bit exponent while squaring a 32-bit base and reducing by a 32-bit modulus. After charging the WISP's storage capacitor to 4.5 V using an external power supply, we removed the power supply and observed the WISP's voltage and GPIO pin on an oscilloscope. Without checkpointing, the CPU's voltage fell below 2.7 V—the voltage supervisor's cut-off point—before the WISP signaled completion. We instrumented a second version of the algorithm with a `checkpoint` routine that interrupted execution after 15 iterations and wrote the current values of the base, exponent, and accumulated result to non-volatile memory. We repeated the original experiment and observed that the program signaled completion after one full charge/execute/recharge/execute cycle. Transiently supplying energy with an external power supply simulates the worst-case energy availability circumstances — i.e., total loss of harvestable energy.

2.5 Applications

Mementos on CRFID is not suitable for all applications. Applications that *are* good candidates for deployment on Mementos share several attributes:

- They require more computational resources than could comfortably be provided by conventional RFID in a single lifecycle.
- They must be at least somewhat *delay-tolerant*; it is always possible that any given CRFID device may not be able to harvest enough energy to compute at any given time.
- Their computations are amenable to splitting so that chunks of execution can occur in different device lifecycles.

We provide examples of applications that may benefit from Mementos.

Zero-power security. Halperin *et al.* [8] used a prototype CRFID device to build a defense mechanism for an implantable medical device. They implemented a simple authentication protocol based on RC5; the device raised a CPU pin to signal another device that it was safe to use its own limited battery to communicate with an external party. The advantage of their technique is that the party that wants to authenticate must "pay" the energy cost of the authentication. A future version could leverage Mementos to implement more sophisticated cryptography.

Visitation proofs. Benaloh and de Mare [2] introduced cryptographic accumulators, which allow membership proofs in constant space. A CRFID device using Mementos could securely and reliably accumulate values with the goal of proving that the device visited a required set of locations.

2.6 Measurements

To measure capacitor leakage, we charged the 10 μF storage capacitor on a WISP (Rev. 1) to 4.5 V using an external power supply, then removed the power supply and measured the capacitor's voltage as the WISP's CPU slept in RAM retention mode. The capacitor exhibited a drop from 4.5 V to 2.7 V (the WISP's minimum operating voltage, comfortably above the 2.2 V threshold for flash writes and the 1.8 V threshold for microcontroller operation) in 700 ms, after which the WISP's voltage supervisor cut power to the CPU. Executing an infinite loop in-

crementing a counter instead of sleeping reduced the fall time to 92 ms.

In a resource-limited CRFID environment, it is possible to observe the effects of individual instructions on the storage capacitor's voltage. For example, writing to flash memory is expensive in terms of current, voltage, and time. We measured energy per instruction for each type of memory access by observing the storage capacitor's voltage drop on an oscilloscope. A clear hierarchy of energy consumption, illustrated by Table 2, emerged, with flash memory writes requiring disproportionately large amounts of energy. Note that *reading* from flash memory is comparable to reading from RAM.

Instr.	Dest.	Src.	Energy/Instr. (nJ)	Perc. Error
`NOP`	—	—	2.0	4%
`MOV`	reg	reg	1.1	23%
		flash	5.2	17%
		mem	6.3	33%
`MOV`	mem	reg	8.1	13%
		flash	11.8	4%
		mem	11.7	7%
`MOV`	flash	reg	461.0	4%
		flash	350.3	1%
		mem	1126.2	4%

Table 2: Energy required per instruction varies on the TI MSP430F1232. Each figure is the average of 5 measurements (smallest and largest discarded) on a WISP (Rev. 1).

Additionally, we observed that the energy consumption of a flash memory write on the MSP430 is not data-dependent. For each of four values containing different numbers of `0` bits, we measured the total energy consumption of writing the value to five consecutive words of flash memory (averaged over five runs). We observed that, for example, the energy costs of writing an all-`0` value and an all-`1` value were indistinguishable within the error bounds.

3 Related Work

Work on checkpointing computations has long focused on providing insurance against occasional failures; the fundamental difference in our work is that failure is the common case. Plank *et al.* [9] discuss checkpointing strategies in detail; their portable Libckpt library for UNIX implements both automatic (periodic, checkpoint-on-write) and user-directed checkpointing strategies.

Research on wireless sensor networks has yielded many results related to minimizing energy consumption. Eon [12] is an energy-aware programming language designed to facilitate "perpetual" systems built on energy harvesting devices. Mementos shares many of Eon's goals—long-running computation, easy programmability, accurate on-line energy measurement—but does not require the programmer to reformulate programs into flows, has tighter resource constraints, and lacks access to a sophisticated operating system.

Sensor networking research has also led to developments in cooperative checkpointing. Using neighbor nodes to store state information [15] is not an option for CRFIDs because they lack the ability to initiate conversations. Checkpointing as a "macroprogramming" primitive [7] is an appealing concept, but no macroprogramming platform for CRFIDs currently exists.

Buettner *et al.* [3] describe WISP-based *RFID sensor networks* (RSNs) that present attractive alternatives to conventional mote-based wireless sensor networks. They discuss the challenges RSNs face — emphasizing the intermittent nature of harvested energy and the asymmetric nature of the underlying RFID protocol. They suggest program splitting as an approach to the execution of large programs. CRFIDs endure many of the same challenges as RSNs, but with emphasis on computation. Our CRFID approach in Mementos [10] focuses on checkpointing and scheduling that do not require manual splitting of programs. We believe Mementos is the first system that automates the instrumentation and execution of programs that tolerate intermittent power on CRFIDs.

Chae *et al.* [4] demonstrated RC5 on a WISP CRFID by carefully choosing parameters so that computations would finish in a single lifecycle. Mementos aims to facilitate computation that spans *multiple* lifecycles.

4 Open Problems

The benefits of Mementos may remain compelling as hardware improves. One direction for hardware improvement is in the area of slack time, the energy cost of which depends on frequency and voltage. Energy efficiency research has focused on the NP-hard [1] problem of optimal discrete dynamic voltage selection, in which processor voltage is selected from a small set after a frequency is selected. The continuous case is easier than the discrete case [1], but hardware limitations—namely, the need to carefully tune analog phase-locked loop and voltage regulator designs [5]—make continuous frequency scaling impractical. We hope our work motivates work on low-energy designs that permit higher-resolution frequency scaling.

Our techniques may also be applicable to devices that are powered by conventional batteries, although on a much longer timescale; Mementos would likely need to estimate energy differently. Our work targets present-day computational RFIDs, which are powered by conventional capacitors and run out of energy quickly. Future CRFIDs may include supercapacitors instead of conventional capacitors, as Yeager *et al.* [14] demonstrate. Supercapacitors store more energy per unit volume than conventional capacitors, enabling semi-autonomous com-

putation without a constant supply of harvestable energy, but they charge slowly; it may take several hours to charge a CRFID's supercapacitor sufficiently for several hours of periodic computation. Future work may strike a balance between supercapacitors and quick-charging conventional capacitors.

Future CRFIDs may also incorporate more memory. While we performed experiments on a WISP (Rev. 1) CRFID, its inventors released a new version [11] bearing a different MSP430 microcontroller with four times as much RAM (1 KB) and four times as much flash memory (32 KB). Advances in microcontroller design and fabrication may allow further expansion of memory without increasing power consumption.

5 Conclusion

Computational RFIDs will enable pervasive computation in places where battery-operated devices are difficult to maintain. Our framework motivates the notion of computational checkpoints as a fundamental abstraction. The preliminary design of Mementos generates compile-time energy hints for energy-aware checkpoints at run-time, and reorders voltage-sensitive instructions while maintaining program semantics to more effectively utilize energy from a continuously varying supply voltage.

6 Acknowledgments

We thank the anonymous reviewers and the members of the SPQR group at UMass Amherst Computer Science and Electrical Computer Engineering for their comments and suggestions. We further thank Wayne Burleson for guidance on architectural and circuit-level behaviors of CRFIDs; John Tuttle for assistance with energy measurements; and Alanson Sample and Joshua Smith from Intel Research for their generous support of the WISP platform. This research was supported by NSF grants CNS-0520729 and CNS-0627529. This research is supported in part by UMass through the CVIP Technology Development Fund. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi. Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems. In *Design, Automation and Test in Europe Conference and Exposition (DATE)*, pages 518–525. IEEE Computer Society, 2004.

[2] J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In *EUROCRYPT*, pages 274–285, 1993.

[3] M. Buettner, B. Greenstein, A. Sample, J. R. Smith, and D. Wetherall. Revisiting smart dust with RFID sensor networks. In *Proc. 7th ACM Workshop on Hot Topics in Networks (HotNets-VII)*, October 2008.

[4] H.-J. Chae, D. J. Yeager, J. R. Smith, and K. Fu. Maximalist cryptography and computation on the WISP UHF RFID tag. In *Proceedings of the Conference on RFID Security*, July 2007.

[5] A. P. Chandrakasan, W. J. Bowhill, and F. Fox. *Design of High-Performance Microprocessor Circuits*. Wiley-IEEE Press, 2000.

[6] C. Ellis. *Controlling Energy Demand in Mobile Computing Systems*. Synthesis Lectures on Mobile and Pervasive Computing. Morgan & Claypool, 2007.

[7] R. Gummadi, N. Kothari, T. Millstein, and R. Govindan. Declarative failure recovery for sensor networks. In *Aspect-Oriented Software Development*, 2007.

[8] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy*, May 2008.

[9] J. S. Plank. Libckpt: Transparent checkpointing under Unix. In *USENIX 1995 Technical Conference*.

[10] B. Ransford and K. Fu. Mementos: A secure platform for batteryless persvasive computing, August 2008. USENIX Security Works-in-Progress Presentation.

[11] 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. In *IEEE Transactions on Instrumentation and Measurement*, 2008.

[12] J. Sorber, A. Kostadinov, M. Garber, M. Brennan, M. D. Corner, and E. D. Berger. Eon: A Language and Runtime System for Perpetual Systems. In *ACM SenSys*, November 2007.

[13] R. Want. *RFID Explained: A Primer on Radio Frequency Identification Technologies*. Synthesis Lectures on Mobile and Pervasive Computing. Morgan & Claypool, 2006.

[14] D. Yeager, P. Powledge, R. Prasad, D. Wetherall, and J. Smith. Wirelessly-Charged UHF Tags for Sensor Data Collection. In *RFID, 2008 IEEE International Conference on*, pages 320–327, 2008.

[15] S. Yi, J. Heo, Y. Cho, and J. Hong. Adaptive mobile checkpointing facility for wireless sensor networks. In M. L. Gavrilova, O. Gervasi, V. Kumar, C. J. K. Tan, D. Taniar, A. Laganà, Y. Mun, and H. Choo, editors, *ICCSA (2)*, volume 3981 of *Lecture Notes in Computer Science*, pages 701–709. Springer, 2006.