Read Bulk Data from Computational RFIDs

Yuanqing Zheng, Mo Li
School of Computer Engineering, Nanyang Technological University, Singapore
{yuanqing1, limo}@ntu.edu.sg

***Abstract*—Without the need of local energy supply, computational RFID (CRFID) sensors are emerging as important platforms enabling a variety of sensing and computing applications. Nevertheless, the data throughput of CRFIDs is very low. This paper aims at efficiently transferring bulk data from CRFIDs to commodity RFID readers. We first investigate the problem of low data throughput of CRFIDs. We then propose several simple yet effective techniques to allow CRFIDs to meet stringent timing requirement of commodity RFID readers and achieve efficient data transfer. We implement a prototype system based on the WISP CRFIDs and commercial off-the-self RFID reader. The experiment results show that our approach provides better compatibility with EPCglobal C1G2 compliant RFID devices and works perfectly with the commodity RFID readers.**

I. Introduction

Different from traditional RFIDs, Computational RFIDs (a.k.a CRFID or RFID-based sensors) feature programmable microcontrollers and various sensors, emerging as an important platform for ubiquitous sensing and computing. CRFIDs harvest energy from the reader's RF interrogation, sense the environment, perform local computation, and transmit data to the reader via backscattering reader's RF signals. Researchers have been exploring the potential of CRFIDs in a wide variety of applications, including inventory management [15, 16, 25], environment monitoring [17], activity recognition [10], access control [13, 27, 28] , etc. An essential requirement of those applications is to efficiently transfer bulk data from CRFIDs to an RFID reader [18]. An efficient data transfer scheme allows individual CRFIDs to quickly offload data and resume sensing tasks with transient harvested power.

To leverage commodity RFID infrastructures, we would like a data transfer scheme to be compatible with the *de facto* EPCglobal Class-1 Generation-2 (C1G2) standard, so that we can collect data using commodity RFID readers. The C1G2 standard [1] specifies a collection of message exchanges between RFID reader and RFID devices. A commodity reader shall be able to collect hundreds of bytes from an RFID device per message exchange. In practice, however, the success rate of data transfer approaches zero as the data size increases. Although many potential reasons have been hypothesized (e.g., frequency drift, power drop, etc. [9, 18, 19, 26]), the root cause of transmission failures still remains elusive.

We first investigate the data transfer failures of CRFIDs and pinpoint the root cause. This not only benefits efficient data transfer but also deepens our understanding on the emerging sensing platform. Nevertheless, commodity RFID readers only provide limited lower layer information to investigate the transmission failures. To carry out a thorough investigation, we build a software defined RFID reader and analyze the communication at physical layer. The unveiled root cause falls outside all prior hypotheses (Section III). The experiment results suggest that the fundamental problem of data transfer stems from the mismatch between the stringent timing requirement of commodity standard and the limited packet handling capability of CRFIDs. In particular, commodity RFID readers set the tight response deadlines on the orders of several μs and ignore delayed responses, while resource-constrained CRFIDs cannot respond in time to offload large volume of data due to excessive computing time involved in CRC calculation. A powerful processor incurs higher energy cost, inapplicable for the energy-harvesting CRFIDs.

In this paper, we aim to fully enable the bulk data transfer and improve data transfer efficiency for CRFIDs while being compatible with the C1G2 standard. To this end, we propose HARMONY, an efficient bulk data transfer scheme which combines a set of simple yet effective techniques. In particular, we propose to preprocess sensor data in preparation for data collection to meet the response deadline of commodity readers (Section IV-A). We present a novel data processing technique to reuse intermediate computation results to save energy and reduce computation overhead (Section IV-B). We ensure proper completion and composition of data processing tasks with a post confirmation mechanism which allows CRFIDs to make an aggressive use of harvested energy (Section IV-C). To the best of our knowledge, HARMONY is the first bulk data transfer scheme that enables CRFIDs to transfer large data packets to commodity readers.

We prototype HARMONY based on the Intel WISP CRFIDs [2, 20] and efficiently transfer large data packets to the Alien ALR 9900+ commodity RFID reader [3]. We evaluate data transfer performance of HARMONY under various scenarios with single CRFID, multiple CRFIDs, as well as mixed CRFIDs and commodity RFIDs. Experiment results demonstrate the outstanding data transfer performance of HARMONY as well as its compatibility with the commodity RFID standard (Section V).

The rest of this paper is organized as follows. We first introduce the background and motivation of CRFID bulk data transfer in Section II. We describe the experiments of identifying the root cause of data transfer failures in Section III. We present the detailed design of HARMONY in Section IV. We present experiment results in Section V. Related work is in Section VI followed by conclusion in Section VII.

II. Background and Motivation

A. Computational RFID

UHF RFID systems work at the frequency band between 902MHz and 928MHz [25]. High power RFID readers with

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

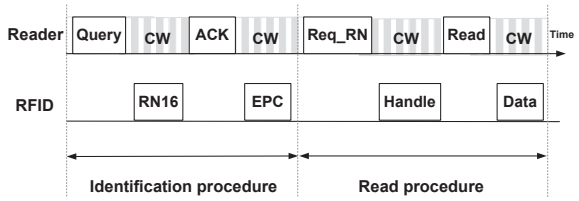


Fig. 1. An illustrative example of C1G2 protocol.

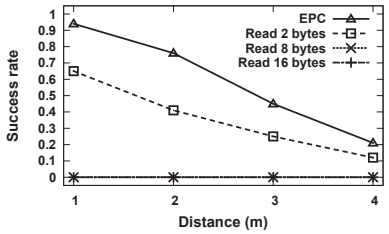


Fig. 2. Transmission success rates of CRFID when a commodity RFID reader requests different amount of data.

30dBm transmission power energize and interrogate RFID devices and collect data from them. A commodity RFID is typically assembled with an dedicated chip implementing state machines, and a printed antenna which harvests energy and backscatters RF signals in communication [25]. The C1G2 standard specifies a set of message exchange primitives.

The commodity RFIDs however do not provide flexible programmability and sensing capability. To fill such a gap, CRFIDs implement the C1G2 protocol on programmable microcontrollers and integrate numerous sensors, e.g., accelerometer, temperature sensor, etc. As CRFIDs work on harvested energy, they adopt ultra-low power microcontrollers such as MSP430 for energy efficiency and low manufacturing cost [2, 20]. The computation tasks involved in communication as well as data processing are performed by the microcontrollers.

Many compelling applications are proposed based on CRFID platforms. For instance, recent work labels everyday objects with CRFIDs and collect accelerometer data to infer daily activities [10]. The compact form factor of CRFIDs allows scientists to collect bio-signals using CRFIDs attached on in-flight insects [29]. Those promising applications benefit from efficient data transfer for CRFIDs.

B. Commodity RFID protocol

Figure 1 illustrates how an RFID reader collects data from an RFID in the C1G2 standard. Before the actual data transmission, a reader can use an optional `Select` command to inform particular RFIDs whether to reply or not in the successive sessions (e.g., using prefix matching). The reader initiates identification procedure by sending the `Query` command. The reader keeps transmitting continuous waves (denoted as CW) to energize RFIDs. The RFID responds a short 16-bit message `RN16`. Then the reader sends an `ACK` containing the `RN16`. The RFID checks whether the `ACK` matches its `RN16` and responds a 96-bit Electronic Product Code (EPC) if it is a match. The {`Query`, `RN16`} and {`ACK`, `EPC`} message exchanges allow the reader to identify an RFID by collecting its globally unique 96-bit EPC [1].

Following the identification procedure, the RFID reader further requests for more data in the read procedure. As shown in Figure 1, the RFID reader first establishes a handshake of {`Req_RN`, `Handle`}. Similar to the {`Query`, `RN16`} handshake in identification procedure, {`Req_RN`, `Handle`} is short, serving the collision arbitration purpose. Then the reader acknowledges the 16-bit `Handle` and requests a large amount of data. According to the C1G2 standard, the reader shall be able to collect up to 510 bytes per {`Read`, `Data`} exchange. Different from {`ACK`, `EPC`} primitive in the identification procedure, the {`Read`, `Data`} primitive in the read procedure is tailored for bulk data transfer with variable lengths [1].

C. Motivating experiment

A natural way of reading bulk data from CRFIDs is to use the {`Read`, `Data`} primitive of C1G2 standard. This approach naturally allows the C1G2 compatibility and may collect hundreds of bytes per message exchange in theory. Nevertheless, the actual success rate of data transfer substantially decreases as large data packets are requested from CRFIDs. We let the Alien ALR 9900+ commodity RFID reader [3] request data from WISP CRFIDs. Figure 2 plots the transmission success rates when the reader requests different amount of data at varied interrogation distance. For comparison, we also measure success rates of EPC transmission. As a successful read procedure always follows a successful EPC transmission, the success rates of reading data from CRFIDs are lower than that of EPC transmission. When more than 8 bytes of data are requested in the read procedure, the success rates suddenly drop to zero even within a small interrogation range. Independent study reports similar error rates when reading bulk data with Impinj Speedway reader [18].

The data transfer failures might be due to clock drift, signal attenuation, and memory overhead, while many other hypotheses include strong interference, high computation overhead, sudden power drop, and incomplete operation execution [9, 18, 19, 26]. Driven by practical demands, researchers endeavor to design various solutions to collect sensor data from CRFIDs. Recent work [18] proposes to let CRFIDs preempt backscatter channel and transfer multiple short identification packets loaded with sensor data in bursts. To ensure fairness in channel access, the C1G2 standard specifies that each RFID should contend for the channel only once in each interrogation round [1]. Thus, CRFIDs experience severe collisions and contentions from coexisting RFID devices.

In this paper, we aim to fully enable the bulk data transfer and improve data transfer efficiency for CRFIDs while being compatible with the C1G2 standard. To this end, we first conduct a series of experiments to identify the root cause of data transfer failures. Based on the experiment findings, we design a set of practical solutions to enable data transfer for CRFIDs and substantially improve the data transmission efficiency.

III. Understanding CRFID data transfer

A. Software-defined testbed

Commercial off-the-self RFID readers do not expose sufficient lower layer information for us to pinpoint the root cause of data transfer failures. We develop a Software-Defined RFID

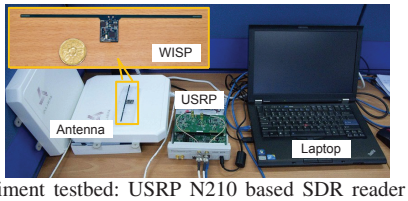


Fig. 3. Experiment testbed: USRP N210 based SDR reader interrogates a WISP CRFID. PHY layer symbols are processed at the laptop.

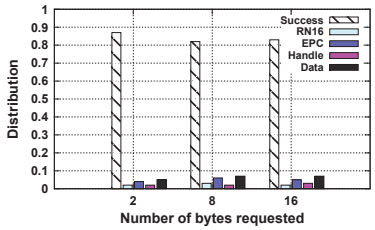


Fig. 4. Distribution of successful transmissions and errors in RN16, EPC, Handle, and Data sessions.

reader (SDR reader) based on the GNURadio [4] platform and the Gen2 project [5] to fetch PHY symbols for analysis. We set the Backscatter Link Frequency (BLF) to 250KHz. The SDR reader uses a USRP RFX900 daughterboard as frontend, which operates at the center frequency of 915MHz. The daughterboard comprises of a full-duplex transceiver with the separate transmission chain and the reception chain [6]. Two Alien ALR-8696-C circular polarized antennas are connected to the daughterboard. The typical power output of RFX900 daughterboard is only 23dBm, far less than 30dBm of commodity reader. The weak power output of SDR reader restricts communication range within 1m in our experiments. Therefore, we start with controlled experiments to let the SDR reader provide sufficient energy for CRFIDs. We also conduct experiments in indoor lab at midnight to reduce potential interferers. We note that the controlled experiment is only for investigation purpose. We later evaluate our data transfer scheme under various practical scenarios in Section V. Figure 3 depicts the experiment testbed.

B. Microscopic investigation

In the experiment, the SDR reader reads different amount of data for 200 times with an interval of 1s. As a successful data transfer involves multiple sessions, we look into each session in Figure 4. We measure the successful data transfer as well as the errors during `RN16`, `EPC`, `Handle`, and `Data` transmission. We let the SDR reader request 2, 8, and 16 bytes of data. We find that in contrast to the high error rates under the interrogation of commodity reader as in Figure 2, CRFIDs can transfer even 16 bytes of data with approximately 83% success rate under the interrogation of SDR reader as in Figure 4. On the other hand, we find that the errors in `RN16`, `EPC`, `Handle`, as well as `Data` are consistently lower than 8%. This experiment result is unexpected since the SDR reader generally cannot match commodity RFID readers in term of power output, signal processing capability, etc.

To understand why the SDR reader outperforms the commodity reader, we look into the PHY layer symbols during data exchanges when the SDR reader requests 8 bytes from the CRFID. Figure 5 plots the PHY layer symbols observed at the SDR reader. At first glance, the raw signals do not

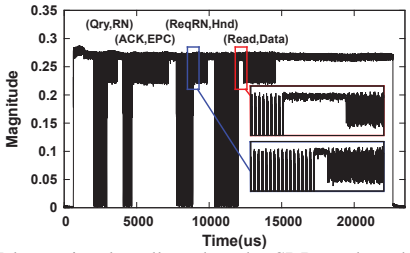


Fig. 5. PHY layer signals collected at the SDR reader when requesting 8 bytes of data. The time interval between Read and Data indicates a large response latency.

TABLE I
READ COMMAND OF RFID READER

Cmd	MemBank	Ptr	WordCount	RN	CRC
8bits	2bits	EBV	8bits	16bits	16bits
0xC2	00:Rsv 01:EPC 10:TID 11:User	Starting address pointer	# of words to read	handle	CRC

exhibit any anomaly. When we zoom in, however, unlike other message exchanges (e.g., {`Query`, `RN16`}, {`ACK`, `EPC`}, and {`Req_RN`, `Handle`}), the big gap between the `Read` command of the SDR reader and the `Data` response of the CRFID clearly stands out. Such a large time interval manifests an excessively long response latency. From the RFID reader's perspective, the delayed responses imply idle slots. In other words, the commodity RFID reader would view the long idle period as no response and ignore the delayed responses from CRFIDs. In contrast, restricted by the high latency of software radio [22], the SDR reader is intentionally configured to tolerate higher response latency. Thus in the experiments, while the commodity RFID reader misses the delayed response, the SDR reader "waits" for the reponse.

We measure the response delay of CRFIDs to different amount of requested data in Figure 6. We find that the response delay increases almost linearly with the data size. The timing requirement of C1G2 standard is extremely tight especially for the lightweight CRFID. According to the C1G2 standard, data responses should be strictly within $T_1 = \mathrm{Max}(\mathrm{RTcal}, 10/\mathrm{BLF})$, where RTcal denotes reader to RFID timing calibration duration, and BLF denotes backscatter link frequency [1]. In the experiment, RTcal is 60μs and BLK is 250KHz which yields a T_1 of 60μs. In Figure 6, we see that as more data is requested, the CRFID cannot meet the stringent timing requirement.

C. Root cause of large latency

To investigate the root cause of the lengthy latency, we carefully examine the runtime of CRFID response to the data request. As shown in Table I, a `Read` command specifies the data size. Table II specifies the format of a `Data` packet in response to the `Read` command [1]. To ensure the data integrity, a CRFID appends a 16-bit CRC in the `Data` packet. A CRC is an error detecting code widely used in digital communications and storage systems [7, 21]. The CRC computation overhead increases linearly with the data size.

CRFIDs need to compute CRCs and append to data packets before responding to readers. The response deadline of C1G2

TABLE II
DATA RESPONSE OF RFID DEVICE

Header	Memory Words	RN	CRC
1bit	Variable	16bits	16bits
0	Data	handle	CRC

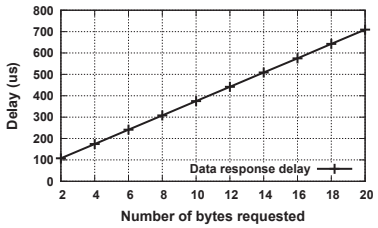


Fig. 6. Data response latency with varied number of requested data bytes.

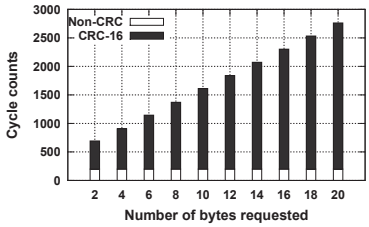


Fig. 7. CPU cycles to finish the computation tasks of data packet response

TABLE III
THE COST OF CPU CYCLES OF THE BITWISE CRC AND THE TABLE-LOOKUP CRC.

Data size (bytes)	2	6	10	14	18	22
Bitwise CRC	493	953	1409	1857	2309	2753
TBL-lookup CRC	125	217	309	401	493	585

standard is extremely tight for CRFIDs with clock frequency of 16MHz. We measure the CPU cycles that CRFIDs take to calculate CRCs for different data sizes. In particular, we analyze the latest CRFID firmwares of both WISP [2] and Moo [30] and count the CPU cycles with offline code execution. Note that derived from the WISP project, the firmware implementation as well as hardware architecture pertaining to the bulk data transfer are essentially the same across all the WISP-based CRFID platforms. For instance, Moo's microcontroller has more memory, but the two MSP430 microcontrollers offer the same clock frequency and instruction cycle time. Figure 7 plots CRC-related computation overhead along with non-CRC computation tasks involved in handling data packets. We find that the non-CRC overhead is small and remains constant across different data sizes. In contrast, the computation overhead of calculating CRC increases linearly, which eventually translates to large response latency. We convert the CRC-related CPU cycles into the runtime according to 62.5ns instruction cycle time of MSP430 specification and find that the processing time roughly fits the response delay (as plotted in Figure 6). As CRFIDs typically adopt ultra-low power microcontrollers for energy efficiency, the mismatch between the limited packet handling capability of CRFIDs and the stringent timing requirement of commodity standard persists across various CRFID platforms. To date, CRFIDs cannot meet the stringent response deadline and transfer large data packets to commodity RFID readers.

A straightforward optimization is to trade memory for time and energy. Current CRFIDs adopt the bitwise CRC calculation [2] which minimizes memory overhead. Nevertheless,

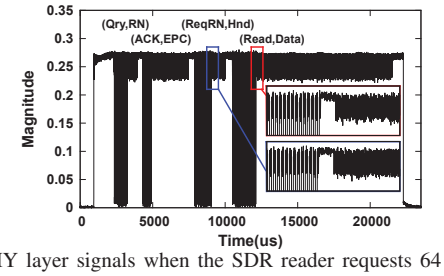


Fig. 8. PHY layer signals when the SDR reader requests 64 bytes of data and the WISP CRFID sends back a Data packet with a precomputed CRC.

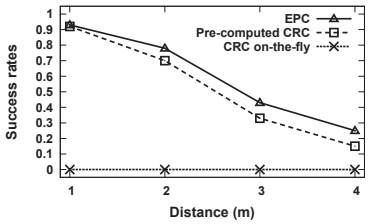


Fig. 9. Transmission success rates when a reader requests 64 bytes.

the bitwise method incurs higher computation overhead as well as higher power consumption. On the other hand, the MSP430 microcontrollers have abundant non-volatile memory (e.g., 8KB flash). We adopt the table-lookup CRC computation which allows CRFIDs to perform bytewise computation and save computation time and harvested energy [8, 21]. Referring to a lookup table, CRFIDs can efficiently shift the raw data in bytes rather than in bits in the CRC computation. Table III compares the computation cost of bitwise CRC calculation and the bytewise table-lookup approach. Although the table-lookup approach reduces the computation time down to approximately 25% of the bitwise approach, the response latency still increases linearly with data size, and the CRFIDs cannot transfer more than 8 bytes per message exchange. Using powerful microprocessors may similarly reduce computation time and deliver slightly more data, but this approach cannot fundamentally solve the problem of data transfer for CRFIDs.

The experiment results suggest that the large response latency of CRFID is the primary reason for data transfer failures, which departs from all existing hypotheses [18, 19, 26]. Although the data transfer primitive is semantically correct, CRFIDs cannot meet the response deadline of commodity standard in practice. The fundamental problem stems from the mismatch between the tight timing demand of commodity standard and the limited packet handling capability of CRFIDs.

IV. SYSTEM DESIGN

We seek an efficient, lightweight, and standard-compliant data transfer scheme. (1) We want an efficient data transfer protocol which allows CRFIDs to quickly offload buffered sensor data and resume sensing tasks. An efficient data transfer protocol also reduces power consumption and saves harvested energy to perform sensing and computing. (2) We want a pure software solution without any hardware extension. Using extra hardware is costly, drains more power, and increases form factors. Powered by harvested energy, current CRFID platforms cannot afford the luxury of dedicated radio chips used in traditional sensor motes with reliable power supply. (3) To leverage commodity RFID infrastructures, we want the data

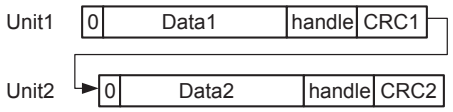


Fig. 10. HARMONY patches intermediate CRCs to concatenate multiple units.

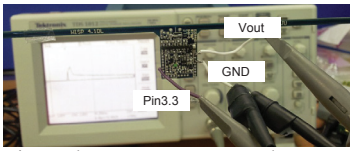


Fig. 11. Capacitor voltage measurement and output pin monitoring.

transfer approach to be compatible with the C1G2 standard. In the following, we explore to schedule workloads, optimize computation, and make the best use of harvested energy.

A. Precomputing CRC

Instead of computing CRC on-the-fly, we explore to precompute CRCs for data packets so as to reduce response latency. We let the SDR reader request 64-byte data from the CRFID. The CRFID transfers data packets appended with precomputed CRCs. Figure 8 plots the PHY layer signals collected by the SDR reader during one data packet transmission. In the figure, we observe that the response latency becomes comparable with other message exchange primitives. Compared with the on-the-fly packet handling as depicted in Figure 5, we see that the response latency can be substantially reduced as in Figure 8.

We further investigate the success rate of data transfer with commodity RFID reader. We let the commodity reader request 64 bytes from the CRFID. The CRFID responds data packets with precomputed CRCs. Figure 9 plots the transmission success rates of using precomputed CRC compared with the success rates of calculating CRC on-the-fly. We plot the success rates of EPC transmission for comparison. According to the results, the CRFID can achieve much higher success rates by precomputing CRC, while computing CRC on-the-fly leads to zero success rates.

The precomputation however requires the prior knowledge of reader command (e.g., WordCount in Table I [1]). Note that a data packet would be decoded incorrectly or directly dropped by commodity reader if the data size does not match the reader command. Using fixed packet size limits the flexibility and efficiency of data transfer. In practice, a CRFID can form a `Data` packet with precomputed CRC beforehand, and overload `EPC` packet to inform RFID readers the packet size during the identification procedure. Then the RFID reader can parse `EPC` and figure out the packet size to request. The RFID reader then specifies the data size in the `Read` command so that reader and CRFID can agree on the packet size. Similarly, a common `handle` can be shared between the CRFID and reader during {`Req_RN`, `handle`} handshakes, as the 16-bit `handle` is determined by the CRFID. By doing this, CRFIDs are able to precompute CRCs and prepare `Data` packets beforehand. We call such self-verifiable `Data` packets as HARMONY units.

B. Exploiting intermediate computations

Precomputing the CRCs and preparing HARMONY units can effectively reduce the response latency. However, the size of each unit prepared by CRFID is fixed. If an RFID reader wants to collect Unit1 and Unit2 from a CRFID as depicted in Figure 10, the two units have to be transferred separately via two data transfer procedures which involves additional communication overhead. Calculating CRC for a new data packet incurs extra computation overhead for CRFIDs and leads to larger latency.

To address this problem, CRFIDs first precompute CRCs and prepare short units, and later concatenate multiple units to form a large data packet according to reader's request. We propose a lightweight approach to concatenate multiple units by reusing intermediate computation results (e.g., CRC1, CRC2 in Figure 10) without calculating a new CRC from every data bit. The key idea is to isolate CRC calculation within each unit and thereby achieve transparent unit concatenation.

Directly concatenating the two units without any changes as in Figure 10 may result in CRC verification failure. When receiving the concatenated packet, the RFID reader parses it (according to Table II [1]) as a leading '0' bit, a data payload starting from Data1 to Data2, a handle, followed by CRC2. In this case, CRC2 cannot serve as a valid CRC for the concatenated packet. The reason is the CRC remainder of Unit1 carries over to Unit2, which invalids CRC2 for the concatenated packet. Therefore, if we isolate the influence of remainder carryover from Unit1 to Unit2, we can reuse the computation result of CRC2 for the concatenated packet.

The C1G2 standard adopts a 16-bit CRC which uses the polynomial divisor (denoted as Poly) of 0x1021. The initial value and final value are both 0xFFFF [2, 8]. The final value of 0xFFFF means that the remainder is bitwise-inverted at the final stage of CRC calculation. Therefore, when the CRC is bitwise-inverted back, we will obtain the remainder. We denote by Rmd the remainder of a message M. In the 16-bit CRC, we have

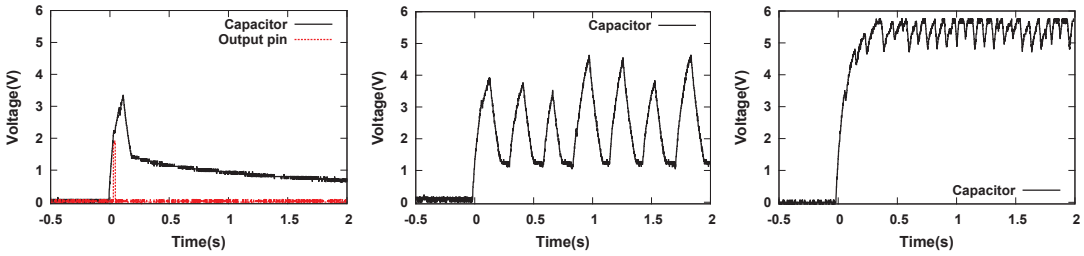
$$\begin{aligned} \mathrm{Rmd} &= (\mathrm{M}|0\mathrm{x}0000) \mod \mathrm{Poly}, \\ \mathrm{CRC} &= \mathrm{Rmd} + 0\mathrm{xFFFF}, \end{aligned} \tag{1}$$

where the notation "|" concatenates message M and 0x0000, both "+" and "−" are bitwise XOR. From Eq.(1), when we append the remainder to the message, we have

$$0\mathrm{x}0000 = (\mathrm{M}|\mathrm{Rmd}) \mod \mathrm{Poly}.$$

Taking the example of concatenating Unit1 and Unit2 in Figure 10, after we bitwise-invert CRC1, the remainder of Unit1 will become 0x0000. Then we treat Unit1 (Data1+handle+CRC1) as a whole data piece and the remainder of 0x0000 will carry over to Unit2. Although the remainder differs from the initial value of 0xFFFF which is used in calculating CRC2 for Unit2, we can leverage the linearity of CRC computation to do an adjustment to convert CRC2 accordingly.

We denote by $\mathrm{CRC}(\mathrm{M}, \mathrm{Ini})$ the CRC result of message M with initial value of Ini. Then the mismatch of CRCs due to different initial values can be described as $\mathrm{CRC}(\mathrm{M}, 0\mathrm{x}0000) \neq \mathrm{CRC}(\mathrm{M}, 0\mathrm{xFFFF})$. Leveraging the linearity of CRC computation, recent theoretical work proposes several approaches to calculate CRC for different initial inputs without recalcu-



(a) Single interrogation (b) Conveyor reading (c) Grid monitoring

Fig. 12. Harvested energy of CRFIDs with different interrogation modes of commodity RFID reader.

lating data content of M [7]. In particular, we have

$$\mathrm{CRC}(\mathrm{M}, 0\mathrm{x}0000) = \mathrm{CRC}(\mathrm{M}, 0\mathrm{xFFFF}) + \mathrm{Patch},$$
$$\mathrm{Patch} = (0\mathrm{x}0000 - 0\mathrm{xFFFF}) \ll |\mathrm{M}| \mod \mathrm{Poly}, \quad (2)$$

where Patch denotes the necessary adjustment to convert $\mathrm{CRC}(\mathrm{M}, 0\mathrm{xFFFF})$ into $\mathrm{CRC}(\mathrm{M}, 0\mathrm{x}0000)$. "$\ll$" represents a bitwise shift and $|\mathrm{M}|$ is the fixed length of message M. In Eq.(2), as all the parameters are known constants, we can compute Patch at compile time and apply it to convert $\mathrm{CRC}(\mathrm{M}, 0\mathrm{xFFFF})$ to $\mathrm{CRC}(\mathrm{M}, 0\mathrm{x}0000)$.

Back to the example of concatenating Unit1 and Unit2 as depicted in Figure 10, a CRFID can first bitwise-invert CRC1 which makes the carryover of Unit1 to Unit2 a constant of 0x0000, and then adjust CRC2 by XORing Patch as computed in Eq.(2). With the adjustment to CRC1 and CRC2, the concatenated packet will pass the CRC test at RFID readers given no errors. Similarly, we can concatenate multiple HARMONY units, at the cost of one 16-bit XOR per unit concatenation.

This patching technique allows CRFIDs to leverage intermediate computation results and concatenate multiple HARMONY units. This approach only incurs small communication overhead (4 extra bytes of `handle` and CRC for every unit). As a matter of fact, the actual transmission time of additional 4 bytes is negligible compared with the control overhead involved in the collision arbitrations (e.g., {`Query`, `RN16`}, {`Req_RN`, `Handle`}). By precomputing and exploiting intermediate results, lightweight CRFIDs can meet the stringent response deadline of C1G2 standard and transfer large data packets to commodity reader without sacrificing the data transfer efficiency and flexibility.

C. Computing with transient power

1) Characterizing harvested energy: Unlike traditional sensor motes with reliable power supplies, a CRFID features transient harvested energy, which comes from RFID reader's interrogation. Although CRFIDs do not actively generate RF signals, the computation tasks may drain onboard energy stored in the tiny capacitor (e.g., 10μF on WISP) and lead to incomplete task execution [9, 19, 26]. The minimum operation voltage is around 1.8V, below which the CRFID cannot work properly, and the over-voltage protection is set to 5.8V for circuit protection purpose [9, 19]. A voltage regulator is used to stabilize alternating capacitor voltages and provide stable power to CRFIDs. As volatile state will be lost if the onboard energy depletes, the computation results need to be properly saved into non-volatile memory. The CRFID adopts 1.8V EEPROM via I2C serial connection, while the flash memory requires a higher voltage of 2.2V. We use an oscilloscope (Tektronix TDS 1012) for power measurement and realtime monitoring. In particular, we toggle Vout (i.e., capacitor voltage) and GND (i.e., ground) pins to monitor the capacitor voltage as in Figure 11.

We measure the capacitor voltage of the CRFID which is in the coverage of a commodity RFID reader. Figure 12 depicts the capacitor voltage over 2s with three typical interrogation modes, i.e., single interrogation in (a), conveyor reading in (b), and grid monitoring in (c). In Figure 12(a), we find that the sporadic interrogation charges the capacitor and the voltage quickly exceeds the operation voltage of 1.8V, which provides sufficient power to perform some computation tasks. When the reader stops RF signals, the capacitor voltage quickly drops due to high power consumption of active microcontroller. When the capacitor voltage drops below 1.5V, the microcontroller transits to sleep mode, followed by the gradual depletion of capacitor voltage due to power leakage. To study whether CRFIDs can compute and store CRCs into non-volatile EEPROM during short energized periods, we measure the runtime by monitoring a pin which outputs indication signals during program execution as plotted in Figure 12(a). According to the result, we see that the CRFID can quickly compute and save computation results into EEPROM. Moreover, the CRFID can potentially compute several CRCs during the short energized period in the single interrogation scenario.

In Figure 12(b), working in the conveyor mode, the RFID reader periodically energizes RFIDs. Thus the CRFID can harvest sufficient energy for sensing and computing. In Figure 12(c), when the reader works in the grid monitoring mode with higher interrogation frequency [3], the capacitor voltage quickly rises up to 5.8V and the voltage protection has to kick in. In the experiment, we see that although the harvested energy exhibits different patterns under different interrogation environments, energy harvesting opportunities are abundant especially when RFID readers frequently interrogate RFID devices in the field.

2) Post confirmation for complete operations: According to the above experimental results, CRFIDs can perform multiple computation tasks during short energized periods. Therefore, we propose to aggressively perform computation tasks when the CRFIDs are powered up. One challenging issue is that the harvested power supply is indeterministic [17], e.g., RFID readers may terminate on-going interrogations, line-of-sight paths between readers and CRFIDs may be blocked, etc. As a

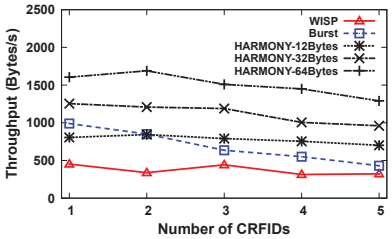


Fig. 13. Aggregated throughput of multiple CRFIDs.

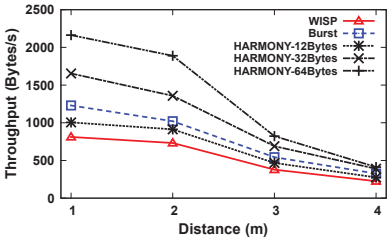


Fig. 14. Single CRFID throughput.

result, CRFIDs may not finish the complete execution of every task and lose intermediate results.

To overcome this problem, CRFIDs must be able to resume incomplete operations. In HARMONY, a successful CRC precomputation finishes with a successful writing operation into EEPROM. While writing data into EEPROM incurs high power cost, reading data from EEPROM is much cheaper. Thus, we propose to use a post confirmation approach to ensure complete task execution. In particular, CRFIDs read the computation results just written to the EEPROM and check whether the computation results have been correctly saved. CRFIDs then update the computation progress if the results are successfully saved. When powered up later, the CRFIDs can resume from the last incomplete operation.

V. Implementation and Evaluation

A. Implementation and experiment setting

1) Commodity RFID reader: We implement the HARMONY data transfer scheme on the Alien ALR-9900+ reader. The commodity reader can work in a variety of configurations, e.g., single reading, continuous monitoring, etc. The implementation overhead of data transfer protocol on the reader side is fairly small, i.e., only slight modifications based on the reader SDK codes are needed. As the HARMONY reader only requires the routine functions of C1G2 standard, it is compatible with other C1G2 standardized RFID readers and thus we believe HARMONY can be implemented on other models of commodity readers.

2) CRFID: We implement HARMONY CRFIDs based on the WISP4.1DL hardware and firmware [2]. The WISP CRFIDs are equipped with MSP430 microcontrollers. The WISP firmware, written in C and assembly codes, has partially implemented the C1G2 standard [1]. We primarily make three extensions to the CRFID firmware: the CRC computation module, the precomputing scheduling module, and the data transfer module. In particular, we adopt the table-lookup CRC algorithm based on the TI specification [8] which roughly saves 75% CRC computation time. CRFIDs cannot transfer more than 8 bytes with the table-lookup approach as the response latency increases linearly with the data size. To reduce response latency, we implement the CRC precomputing algorithm and ensure complete operation with the post confirmation mechanism to allow CRFIDs to make the best use of transient energy. We implement the data transfer module which concatenates multiple HARMONY units to improve data transfer efficiency and flexibility. HARMONY only requires firmware updates, so our scheme can be applied on other WISP-based CRFID platforms [17, 23, 30].

3) Experiment setup: The Alien ALR-9900+ reader uses one ALR-8696-C circular polarized antenna with antenna gain of 8.5dBic for transmission as well as reception. The RFID devices are attached to a poster panel parallel to the antenna 1m away with line-of-sight paths to the reader. We vary the power output from 18dBm to 30dBm to study the data transfer performance with different power conditions. We present experiment results in equivalent distances calculated using free-space propagation for intuitive interpretations compared to different power outputs [9].

B. Performance of HARMONY

We compare the throughput of three data transfer schemes: (1) HARMONY, the proposed scheme in this paper; (2) WISP [2], the default data transfer scheme of WISP that overloads one identification packet in each interrogation round; and (3) Burst [18], the burst data transfer scheme that exclusively occupies backscatter channel and uses multiple identification packets to transfer data in each interrogation round. We study the data throughput under various scenarios with single CRFID, multiple CRFIDs, as well as mixed CRFIDs and commodity passive RFIDs.

1) Single CRFID: We measure the data transfer throughput where single CRFID transfers data without contention or collision from coexisting RFID devices. We use the commodity reader to power up and interrogate the CRFID. The CRFID senses ambient temperature, preprocesses sensor data, and buffers intermediate data in the non-volatile memory. We collect the same amount of data from the CRFID using three different schemes and compare their performance. We repeat experiments at 4 different power outputs. We collect 100 data traces for each scheme immediately one after another with little change to the interrogation environment. We report the average throughput in Figure 14. According to the experiment results, we see that by exclusively occupying the channel and sending multiple identification packets in each interrogation round, the burst transfer scheme can achieve higher throughput than the default data transfer scheme of WISP which transfers only one identification packet in each round. HARMONY can also transfer multiple data packets after the handshake establishment of {`Req_RN`, `Handle`}. Besides, HARMONY gains much higher throughput by transferring larger data packets per message exchange. With large data packets, CRFIDs can effectively amortize the control overhead involved in the handshake establishment. According to Figure 14, HARMONY with 32-byte data packets already exceeds the throughput of burst transfer scheme. With 64-byte data packets, HARMONY yields almost 80% improvement over the burst transfer scheme. According to the C1G2 standard, a

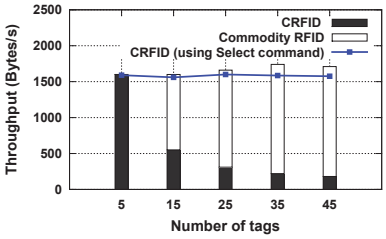


Fig. 15. Aggregated data throughput of mixed CRFIDs and commodity passive RFIDs.

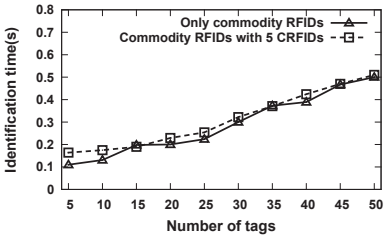


Fig. 16. Time to identify RFIDs.

commodity reader shall be able to read 510 bytes of data per {`Read`, `Data`} exchange [1]. The Alien ALR-9900+ reader, however, only supports a request up to 64 bytes. We expect a higher throughput gain once RFID readers fully support the C1G2 standard [3].

2) Multiple CRFIDs: We study the data transfer performance where a commodity RFID reader collects data from multiple CRFIDs. The RFID reader takes turns to interrogate each CRFID. Each CRFID sends off the bulk data in response to the data collection in its turn. For the rest time, each CRFID senses the environment, stores the data, and performs local computation (e.g., to precompute unit CRCs in HARMONY). Such an iterative sensing and interrogation mode can be applied to many practical applications [10, 23, 29].

In the experiment, the commodity reader requests the same amount of data from varied number of CRFIDs. We report the aggregated throughput in Figure 13. We find that HARMONY with large data packets (e.g., 32-byte and 64-byte) consistently outperforms the benchmark schemes and the aggregated throughput remains stable across different CRFID population. In HARMONY, each CRFID strictly follows the C1G2 standard in channel contention and harmonically coexists with other CRFIDs. Following the C1G2 standard, HARMONY CRFIDs ensure exclusive collision-free channels with lightweight handshakes, and transfer large data packets in their turns. In the default WISP data transfer scheme, the aggregated throughput does not decrease with the increased number of CRFIDs since CRFIDs also follow the C1G2 standard. In the burst scheme, the aggregated throughput gradually decreases as the CRFID population scales up due to severe collisions among the burst CRFIDs.

In summary, the performance gain of HARMONY primarily stems from two aspects. First, HARMONY CRFIDs can effectively transfer larger data packets which fundamentally improves the data transfer efficiency. Second, HARMONY avoids collisions in data transfer among multiple coexisting CRFIDs which guarantees high aggregated throughput.

3) Coexisting with commodity RFIDs: We evaluate data transfer performance of HARMONY coexisting with commodity RFID devices. In the experiment, we evaluate the data transfer performance of HARMONY as well as its impact on commodity RFID devices. We test with 6 different types of commodity passive RFIDs from 2 different RFID manufacturers [3]. The commodity RFIDs are equipped with non-volatile memory which allows an RFID reader to read/write data from/to the RFIDs [1]. We do not compare with other bulk data transfer approaches, because those approaches are not fully compatible with the C1G2 commodity RFIDs.

First, we let the commodity RFID reader request 64 bytes from 5 CRFIDs coexisting with varied number of commodity RFIDs. Figure 15 plots the aggregated throughput of 5 CRFIDs as well as coexisting commodity RFIDs. According to the results, as the number of commodity RFIDs increases, the throughput of commodity RFIDs eventually overtakes that of CRFIDs. This result is expected since the C1G2 standard provides equal chances for HARMONY CRFIDs as well as commodity RFIDs to transfer data.

In some applications, we may want to collect data only from HARMONY CRFIDs but not from coexisting commodity RFIDs. In such cases, the RFID reader can use the optional `Select` command to turn passive RFIDs into sleep mode [1]. We let the reader use the `Select` command to suppress contentions from coexisting commodity RFIDs, and then request 64-byte data packets from CRFIDs. The blue line in Figure 15 plots the aggregated throughput of HARMONY CRFIDs. According to the results, we see that the aggregated throughput remains consistently high since the undesirable contentions from commodity RFIDs can be effectively suppressed.

In the following, we examine how HARMONY CRFIDs influence the identification procedure of commodity RFIDs. In some practical scenarios, a reader may want to identify RFIDs rather than collecting bulk data from them. In Figure 16, we measure the average identification time with varied number of commodity RFIDs changing from 5 to 50. We see that the identification time increases with the number of commodity RFIDs. To examine the influences of HARMONY CRFIDs, we replace 5 commodity RFIDs with 5 HARMONY CRFIDs and repeat the experiment without changing the experiment setting. HARMONY CRFIDs behave in much the same way as the C1G2 compliant RFIDs in the identification procedure. We find that the reader takes almost the same amount of time to identify the mixed RFID devices. This result suggests that HARMONY CRFIDs can peacefully coexist with commodity RFIDs with negligible influence on the identification performance.

VI. Related work

CRFID platforms. Since the pioneering Intel WISP project [2], many CRFID platforms have been developed to enable ultra-low power sensing and computing. We have implemented and tested HARMONY on the latest version of WISP4.1DL CRFIDs. The SoCWISP [29] adopts the system-on-chip design which is sufficiently small to be attached to in-flight insects for bio-signal collection. The Blue Devil WISP [23] uses a v-shape antenna for improved omnidirectional antenna gain in complex interrogation environments. The Moo [30] upgrades the CRFID hardware based on the WISP for larger memory

and flexible general-purpose I/O extensions. The WISP5.0 CRFIDs under development adopt dual orthogonal antennas to improve energy harvesting efficiency and increase communication range [2]. The mismatch between the stringent requirement of commodity standard and the limited packet handling capability of CRFIDs persists across all the CRFID platforms.

Testbed and measurement. The Gen2 project [5] implements an opensource RFID reader based on the GNURadio project [4], which provides the access to PHY/MAC layer and allows fast prototyping of novel RFID protocols. We build our software radio testbed based on the Gen2 project. The Gen2 Monitor [12] uses software radio as a probe to analyze backscatter communications, which allows researchers to reverse-engineer and study commodity RFID systems [19]. The Gen2 Listener [14] decouples the functionality of transmission and reception of RFID readers which simplifies parallel interrogation and cooperative decoding. Buettner *et al.* [11] characterize the PHY/MAC layer of RFID systems, which promotes our understanding of RFID communications.

Data transfer. BUZZ [24] proposes to exploit PHY layer collisions and decode them to improve data transfer efficiency. BUZZ however requires the PHY layer information which is not available on current commodity readers. Flit [18] leverages idle slots and transfers short data packets in bursts. Flit however is not fully compatible with the commodity RFID standard and the CRFIDs experience severe collisions at scale. BLINK [31] proposes an efficient data rate adaptation scheme for RFID communications. We present an efficient, lightweight, and standard-compliant data transfer scheme, which enables CRFIDs to transfer large data packets to commodity readers.

Applications. Buettner *et al.* [10] label everyday objects with CRFIDs and infer daily activities from accelerometer readings, necessitating continuously data collection from multiple CRFIDs. Yeager *et al.* [29] design wearable ultra-small CRFIDs to collect bio-signals from in-flight insects which would also benefit from efficient data transfer schemes. Czeskis *et al.* [13] propose to enhance the security of IC cards with CRFID enabled secret handshakes. All such applications benefit from the efficient bulk data transfer with HARMONY.

VII. Conclusion

In this paper, we describe the design and implementation of HARMONY, an efficient bulk data transfer scheme for CRFIDs. We conduct comprehensive experiment study for a better understanding on the CRFID data transfer. The fundamental problem of data transfer stems from the mismatch between the tight timing demand of commodity standard and the limited packet handling capability of CRFIDs. We combine a set of simple yet effective techniques to enable CRFIDs to meet the stringent response deadline and transfer large data packets to commodity RFID readers. In particular, we propose to preprocess data packets to reduce response latency, efficiently reuse intermediate computation results, and resume incomplete tasks to aggressively exploit ephemeral power. We implement a prototype on top of the WISP CRFIDs and a commodity RFID reader. Experiment results show that HARMONY substantially improves the data transfer efficiency.

Acknowledgement

We acknowledge the support from Singapore MOE AcRF Tier 1 grant MOE2013-T1-002-005, and NTU Nanyang Assistant Professorship (NAP) grant M4080738.020.

References

[1] EPC C1G2 Standard. http://www.epcglobalinc.org/standards/uhfc1g2.
[2] WISP. http://wisp.wikispaces.com.
[3] Alien Tech. http://www.alientechnology.com.
[4] GNU Radio. http://gnuradio.org.
[5] Gen2 project. https://www.cgran.org/wiki/Gen2.
[6] Ettus Research. http://www.ettus.com.
[7] A. Kadatch and B. Jenkins. Everything we know about CRC but afraid to forget. http://crcutil.googlecode.com/files/crc-doc.1.0.pdf.
[8] MSP430 CRC. http://www.ti.com/lit/an/slaa221/slaa221.pdf.
[9] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: an energy-aware runtime for computational rfid. In *USENIX NSDI*, 2011.
[10] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall. Recognizing daily activities with rfid-based sensors. In *ACM Ubicomp*, 2009.
[11] M. Buettner and D. Wetherall. An empirical study of uhf rfid performance. In *ACM MobiCom*, 2008.
[12] M. Buettner and D. Wetherall. A 'gen 2' rfid monitor based on the usrp. *SIGCOMM Computer Communication Review*, 40(3):41–47, 2010.
[13] A. Czeskis, K. Koscher, J. R. Smith, and T. Kohno. Rfids and secret handshakes: defending against ghost-and-leech attacks and unauthorized reads with context-aware communications. In *ACM CCS*, 2008.
[14] D. De Donno, F. Ricciato, L. Catarinucci, A. Coluccia, and L. Tarricone. Challenge: towards distributed rfid sensing with software-defined radio. In *ACM MobiCom*, 2010.
[15] W. Gong, K. Liu, X. Miao, and H. Liu. Arbitrarily accurate approximation scheme for large-scale rfid cardinality estimation. In *IEEE INFOCOM*, 2014.
[16] W. Gong, K. Liu, X. Miao, Q. Ma, Z. Yang, and Y. Liu. Informative counting: Fine-grained batch authentication for large-scale rfid systems. In *ACM MobiHoc*, 2013.
[17] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan. On the limits of effective hybrid micro-energy harvesting on mobile crfid sensors. In *ACM MobiSys*, 2010.
[18] J. Gummeson, P. Zhang, and D. Ganesan. Flit: a bulk transmission protocol for rfid-scale sensors. In *ACM MobiSys*, 2012.
[19] B. Ransford, J. Sorber, and K. Fu. Mementos: system support for long-running computation on rfid-scale devices. In *ASPLOS*, 2011.
[20] 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 Transactions on Instrumentation and Measurement*, 57(11):2608–2615, 2008.
[21] D. V. Sarwate. Computation of cyclic redundancy checks via table look-up. *Communications of the ACM*, 31(8):1008–1013, 1988.
[22] T. Schmid, O. Sekkat, and M. B. Srivastava. An experimental study of network performance impact of increased latency in software defined radios. In *WinTECH*, 2007.
[23] S. Thomas, J. Teizer, and M. Reynolds. Smarthat: A battery-free worker safety device employing passive rfid technology. In *IEEE RFID*, 2011.
[24] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. In *ACM SIGCOMM*, 2012.
[25] R. Want. An introduction to rfid technology. *IEEE Pervasive Computing*, 5(1):25–33, 2006.
[26] X. Xu, L. Gu, J. Wang, and G. Xing. Negotiate power and performance in the reality of rfid systems. In *IEEE PerCom*, 2010.
[27] L. Yang, J. Han, Y. Qi, and Y. Liu. Identification-free batch authentication for rfid tags. In *IEEE ICNP*, 2010.
[28] L. Yang, J. Han, Y. Qi, C. Wang, T. Gu, and Y. Liu. Season: Shelving interference and joint identification in large-scale rfid systems. In *IEEE INFOCOM*, 2011.
[29] D. Yeager, F. Zhang, A. Zarrasvand, N. T. George, T. Daniel, and B. P. Otis. A 9 μA, addressable gen2 sensor tag for biosignal acquisition. *IEEE Journal of Solid-State Circuit*, 45(10):2198–2209, 2010.
[30] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational rfid and sensing platform. *UMASS Tech Report UM-CS-2011-020*, 2011.
[31] P. Zhang, J. Gummeson, and D. Ganesan. Blink: a high throughput link layer for backscatter communication. In *ACM MobiSys*, 2012.