Flit: A Bulk Transmission Protocol for RFID-Scale Sensors

Jeremy Gummeson
Universty of Massachusetts Amherst, Dept. of Computer Science
gummeson@cs.umass.edu

Pengyu Zhang
Universty of Massachusetts Amherst, Dept. of Computer Science
pyzhang@cs.umass.edu

Deepak Ganesan
University of Massachusetts Amherst, Dept. of Computer Science
dganesan@cs.umass.edu

Abstract

RFID-scale sensors present a new frontier for distributed sensing. In contrast to existing sensor deployments that rely on battery-powered sensors, RFID-scale sensors rely solely on harvested energy from sources such as RF from a reader and ambient light. Their small form factor, negligible weight, and long deployment lifetimes, makes them ideal for several indoor and urban tagging and tracking applications. These devices sense and store data when not in contact with a reader, and use backscatter communication to upload data when a reader is in range. A key challenge is that unlike existing RFID tags that only transmit identifiers, RFID sensors need to transfer more data to a reader during every contact event. In this paper, we propose several optimizations to the RFID network stack to support efficient bulk transfer while remaining compatible with existing EPC Gen 2 readers. Our key contribution is the design of a coordinated bulk transfer protocol for RFID-scale sensors that maximizes channel utilization and minimizes energy lost to idle listening while also minimizing collisions. We present an implementation of the protocol for the Intel WISP, and describe how several protocol parameters can be determined using empirical measurements that characterize the wireless channel. Our results show that our burst protocol vastly improves goodput to a reader in comparison with vanilla EPC Gen 2 tags, improves energy-efficiency and better use of a small energy buffer for data transfer, allows multiple RFID sensors to share the channel, and also coexists with passive, non-sensor tags.

1 Introduction

A wide range of sensing applications require miniature, ultra-low power, and sensing in urban and indoor areas. This has led to an increased interest in the design of small, cheap, harvesting-based devices that are attached to common everyday objects (e.g., books, furniture, walls, doors, produce, etc), which can be used for tracking these items [8]. One example of such devices are RFID-scale sensors that exploit ambient light or RF for energy, and use backscatter communication with an RFID reader for data transfer.

RFID-scale sensors, also referred to as Computational RFIDs or CRFIDs, present a new frontier for distributed sensing [5]. These devices are distinct from existing battery-powered sensor platforms as well as commercial RFID tags. They are designed for continuous sensing, however, unlike existing continuous sensing devices (e.g. Motes), they use small capacitor buffers, rely solely on energy harvesting, and use more power-efficient backscatter communication for data transfer. CRFIDs are also distinct from commercial RFID tags in that they use hybrid harvesting to enable continuous sensing, computation, and storage rather than just vanilla identification.

In this paper, we investigate how to efficiently utilize the energy buffer of an energy harvesting CRFID node for enabling communication to a reader. CRFIDs are largely mobile and their movements result in two communication states: connected and disconnected. While tags are moving through their environment they perform any number of sensing and computation operations. As time elapses, these devices buffer some amount of data during disconnected operation; occasionally they encounter a reader, resulting in a period of connected operation of non-deterministic length, during which the tag may transmit buffered data.

Our goal is to optimize bulk data transfer from a CRFID sensor to an RFID reader. Because CRFID sensors buffer small amounts of energy and potentially buffer large amounts of data, it is imperative that communications maximize the goodput while minimizing the amount of energy per unit data. This presents several challenges. First, commercial RFID readers follow the EPC Gen-2 protocol which is optimized for large numbers of tags that each transfer a small amount of data (tag identifier). This design makes it inefficient when operating with CRFID sensors that are fewer in number but need to transfer large amounts of buffered data to a reader. While a complete re-design of the protocol stack is possible, this would mean that CRFIDs cannot take advantage of existing commercial RFID readers, making them far less attractive for widespread use. Rather, we seek to support efficient bulk data transfer while still being compatible with commercially available EPC Gen 2 RFID readers. Second, in comparison with other sensor platforms, CRFIDs have different hardware components, use different energy sources, use different energy buffers, and follow a different communication protocol. Thus, designing an energy-optimized bulk transfer protocol for RFID sensors requires an entirely new set of bandwidth and energy-optimization mechanisms. Third, CRFIDs present different usage scenarios since they are largely deployed indoors, and often on mobile objects or people. Thus, any data transfer protocol should operate effectively under such scenarios where there are short contact durations with readers, and considerable changes in link characteristics during mobility.

Our protocol, Flit, provides a fast and efficient alternative to the existing EPC Gen 2 protocol for bulk data transfer from sensors, while still remaining compatible with existing RFID readers. Flit makes three fundamental changes to the protocol stack. First, it enables each sensor to transfer data in a burst by responding to

all slots in a query round rather than just its assigned slot. This design choice improves goodput and energy-efficiency by reducing wasted slots, and takes advantage of extended query rounds with less control overhead. Second, Flit coordinates across sensors by using explicit burst notifiers that are echoed by RFID readers, rather than allowing devices to randomly pick a slot and transmit. This approach serializes burst transfers across nodes, thereby allowing greater goodput while reducing potential for collisions. Third, Flit improves energy-efficiency by duty-cycling the RFID sensor when another CRFID is in the middle of a burst. This avoids wasted energy due to overhearing of reader messages during the burst, thereby enabling better use of a small buffer of stored energy on the CRFID sensor.

Our results show that:

- Flit achieves 60% greater goodput than EPC Gen 2 for a single tag at different distances from the reader, and for different mobility conditions. A breakdown shows that much of these gains are due to the use of larger query rounds, and avoiding wasted slots in the round.
- Flit achieves 4.5x more goodput than an EPC Gen 2 tag when three tags are transferring data concurrently, and 9.2x goodput when five tags are transferring simultaneously. In addition, Flit has considerably higher fairness than EPC Gen 2, which is skewed towards the node with highest SNR to the reader. This allows sensors to take advantage of shorter contact durations with readers.
- Duty-cycling in Flit acheives 34.6% better energy efficiency than a non-duty cycled implementation. These energy savings are acheived by minimizing energy consumptions from listening to other CRFIDs' transmissions.

2 An EPC Class 1 Gen 2 Primer

The EPC Class 1 Gen 2 protocol is widely used by passive RFID tags and readers using far-field communications in the 900 MHz ISM band. Because RFIDs are primarily used for inventorying applications, the protocol is optimized for interrogating large numbers of tags that each report a small amount of data. In this section, we provide a brief introduction to the implementation details of Gen 2 that are relevent to this work.

2.1 Gen 2 Protocol Commands

The Gen 2 protocol for RFID tags is designed to inventory large tag populations over a number of communication rounds. To realize this protocol, a CRFID must traverse a simple state machine and respond appropriately to a set of reader commands. Throughout this discussion, refer to Figure 1 to understand how a sequence of reader commands and tag responses are used to transmit data to a reader. The critical subset of EPC commands a CRFID must implement are:

Query, QueryRep, and QueryAdjust: A *Query* message (1) is used to initiate a round of communication with tags. This message is 22 bits long and contains several parameters to be used for this round of communication such as the modulation and frequency to be used for the backscatter communication link, and the number of communication slots that will follow the query. The parameter critical to this discussion is the number of subsequent slots, Q, which defines the number of slots to be $2^Q - 1$ where $0 \leq Q \leq 15$. Tags generate a random slot counter within the range of available slots determined by Q, while the reader chooses Q such that the probability of tag collision is minimal.

Slots after a *Query* are filled with a series of *QueryRep* (10) and *QueryAdjust* messages. A *QueryRep* message indicates a successive slot in the current round of communication, while a *QueryAdjust* both indicates a new slot and the tag should increment or decrement its Q value. Upon receiving either of these messages, a tag decrements its slot counter; if this value becomes 0, the tag proceeds with communication.

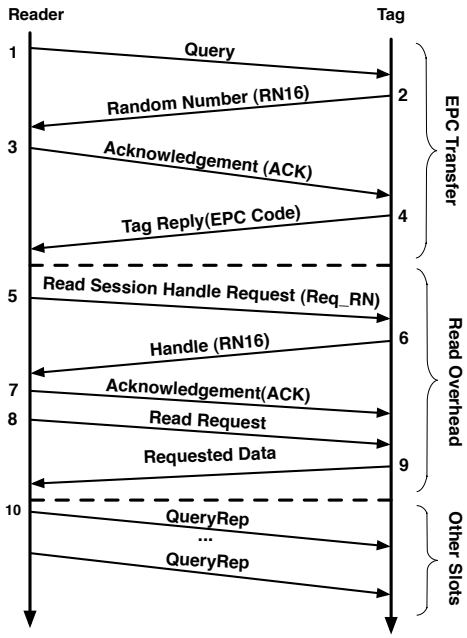

Figure 1. An RFID reader initates a round of communication with a Query message. A query is followed by a number of Rep messages depending on the number of slots in a round chosen by the reader. Each tag responds in one randomly chosen slot and can optionally complete a read operation.

Ack(RN16): A tag could directly reply to a query message, but this is problematic because multiple tags could potentially choose the same slot counter with probability $\frac{1}{2^Q-1}$ and collide; this is especially likely for small Q values. The reader must have a way to disambiguate a specific tag to avoid this issue; for this purpose, an *RN16* (2) message is used. The RN16 message contains a 16-bit random number generated by the tag. Upon decoding an RN16 from tag(s), the reader will echo back the RN16 as an *ACK* (7) it received or the one it reconstructed or captured in the case of a collision.

EPC: A tag can determine its RN16 was chosen by the reader if it matches what was sent; if not, the tag gives up on this round of communication to avoid further collisions. After receiving its own RN16, the tag may backscatter its *EPC* (4) code to the reader. This EPC code typically corresponds to a statically assigned Identifier. An EPC code consists of 2 bytes of header, 12 bytes of data and a 2 byte CRC. After sending its EPC, the tag will not respond to subsequent QueryReps or QueryAdjusts during this round of communication.

Req_RN: A reader that wants to further investigate a tag's state after receiving its EPC code has the option to establish a communication handle using the commmand *Req_RN* (5). A Req_RN command consists of 40 bits of data: one byte of command data, two bytes that contain the previously established RN16, and two bytes of CRC. A tag responds by sending a handle and 2 bytes of CRC; the reader ACKs (7) this message in the same way that precedes the EPC.

Read: After establishing a session handle, the reader may send a *read* command (8) to read a segment of the tag's memory. This

command is 52 bits long: it contains one byte of header, 2 bits that indicate a region of memory, a 2 byte address pointer, 1 byte containing the number of 16-bit words to be read, the previously defined handle, and 2 bytes of CRC. After receiving a Read command, the tag responds with the requested data (9). The RFID appends the session handle and a two byte CRC computed across the payload.

2.2 EPC Gen 2 Physical Layer

In addition to establishing reader and tag command sequences, the Gen 2 protocol also defines a specification for the physical layer used for communication between a reader and tags. Similar to active radios, communication between a reader and tag is half-duplex, meaning that tags cannot demodulate reader information while backscattering requested data. Additionally, a tag relies completely on the carrier wave transmitted by the reader to both harvest energy and send information. Based on these unique channel characteristics, Gen 2 specifies the physical layer modulation and encoding schemes used for the communication links between tags and readers:

R→T modulation: The reader sends information to a tag by modulating the RF carrier with amplitude shift keying (ASK). The reader has the option to use one of three types of amplitude shift keying (ASK): double-sideband ASK, single-sideband ASK, or phase-reversal ASK. The motivation behind using ASK for signaling is because of the simplicity of the decoding process; tags need only measure the amplitude of the carrier wave to demodulate reader information. Readers use a fixed modulation format and data rate for the duration of an inventory round, although tags are capable of demodulating all three schemes. A reader will initiate any signaling with either a preamble or a frame-sync bit sequence. A preamble shall precede a Query command and denotes the start of an inventory round. All other signaling begins with a frame-sync bit sequence.

R→T encoding: Reader to tag communications use pulse-interval encoding (PIE). A logical 0 is encoded by the reader as one high amplitude pulse and one low amplitude pulse; a logical 1 is encoded as three high amplitude pulses and one low amplitude pulse. Tags need only measure the amplitude and width of pulses to decode bits information; in practice the amplitude is measured by an analog comparator circuit and the tag only considers binary amplitutdes. The main shortcoming of PIE is the bit error rate introduced by quantization error. For example, if the second pulse of a logical 1 is quantized as low, a logical 1 will be interpreted as two 0s.

T→R modulation: A tag backscatters data by detuning its antenna; it uses fixed data encoding and data rate for the current inventory round as specified by the reader's query message. Tags can select ASK or PSK modulation for its response; the reader is capable of demodulating either type of modulation. Because tags rely on the reader's carrier wave to backscatter data, the modulation scheme of a tag is limited by the modulation used for the carrier wave. Therefore frequency based modulation schemes, such as FSK, cannot be used.

T→R encoding: One option for the tag to reader communication link, and the one the Intel WISP uses, is the miller-4 encoding scheme. When using this scheme, a bit of information will be encoded as normal high-low pulse combinations. For example, Miller-4 encodes a logical 0 as 4 high-low pulse combinations; however, the phase between two sequential 0s is inverted. A logical 1 is encoded as a phase inversion within a pulse. Because the reader has more computational resources and power, more complex encoding schemes such as Miller encoding, as opposed to PIE, can be used to adapt the channel characteristics. Miller-4 encoding is widely used in encoding radio signals because the frequency spectrum of the encoded signal contains less low-frequency energy than a conventional non-return-to-zero signal and less high-frequency energy than a biphase signal.

Protocol Operation	bits	Active Time	Idle Time	Energy
Query	22	983 μs	52 μs	648 nJ
QueryRep	4	273 μs	50 μs	210 nJ
QueryAdujst	9	415 μs	51 μs	319 nJ
Read	52	2100 μs	50 μs	1615 nJ
RN16	16	641 μs	2390 μs	422 nJ
Ack	18	660 μs	36 μs	508 nJ
Req_RN	40	1616 μs	51 μs	1241 nJ
EPC	128	2450 μs	2360 μs	1615 nJ
CRC16	–	452 μs	–	307 nJ

Table 1. An CRFID must emulate a protocol in software; this leads to widely varying amounts of energy consumption depending on the command message the sensor decodes or the data it sends. Decoding a read message consumes twice the energy of decoding a query.

3 The Implications of EPC for Sensors

In this section, we discuss several implications the EPC Gen 2 protocol has on designing a bulk data transfer protocol that transfers data from a CRFID sensor to a reader.

3.1 Read vs EPC for burst data transfer

The first question in designing a burst data transfer protocol is which EPC Gen 2 message primitive to use as the building block for transferring data. In the previous section we, introduced the EPC protocol and showed how tags use protocol messages to backscatter a 12-byte EPC identification code. Two options present themselves in terms of adapting the EPC Gen 2 protocol for bulk data transfer from the sensor to the reader. The first option is to use the *EPC* message, and send the data instead of the 12 byte identifier within this message. The second solution would be to use the EPC read command. The read command allows an arbitrary amount of data to be sent from a tag to a reader after it is selected in a normal communication round.

At first glance, the Read message appears to be the best choice. It can support variable length messages, thereby easily supporting variable amounts of sensor data transfer from the tag to the reader. In contrast, an EPC message is always 12 bytes. In addition, Read messages are directly compatible with the EPC Gen 2 protocol and therefore needs no further modifications to use on existing RFID readers. But, as we show below, Reads are inefficient in terms of energy consumption and channel utilization, making this the wrong choice for CRFIDs.

Energy comparison: We compare the energy efficiency of reads vs EPC messages using a set of energy benchmarks. The energy efficiency of the read command varies with the length of the data sent in response to the read request, while an EPC message is always 12 bytes. Table 1 shows a breakdown of the energy consumed for each message in the EPC Gen 2 protocol exchange. These benchmarks were captured using the Intel WISP tag [15] (more details in §5).

From this breakdown, we can compute the amount of energy consumed per byte of data transferred. Each Read command incurs a total energy overhead of 5163 *nJ* for the steps 1–8 from Figure 1 that precede the read payload. The energy consumed for each EPC command varies a small degree based on whether it is in response to a Query, QueryRep or QueryAdjust since they have different sizes; it is 1885 *nJ* for a Query, 1447 *nJ* for a QueryRep and 1556 *nJ* for a QueryAdjust.

We can see from the numbers that for a payload size of 12 bytes, an EPC message in sent in response to a QueryRep is 3.7 $\times$ more

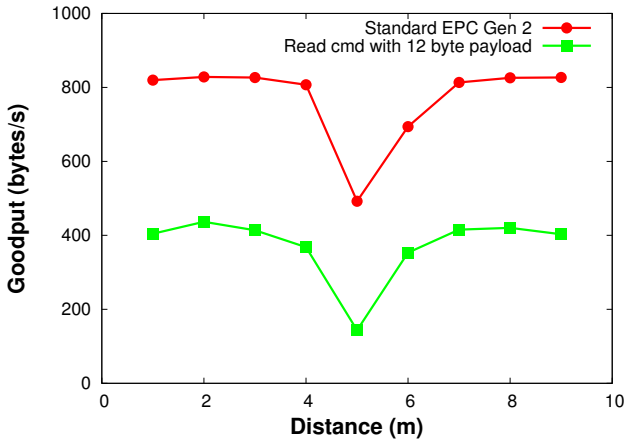

Figure 2. A read command that requests 12 bytes achieves half the throughput of EPC-based messages as a result of additional protocol overhead.

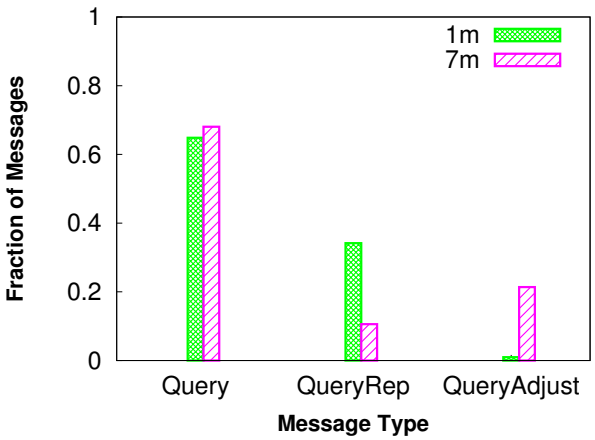

Figure 3. Communication slots are biased towards Queries at short range (1m). At a longer range (7m) losses result in the reader sending more QueryAdjust messages.

efficient than a read of the same size; EPCs are more efficient because the number of message exchanges in the protocol are a strict subset of those that occur in a read as shown in Figure 1. In order to break-even, a read message would need to have a payload greater than 57 bytes.

Goodput comparison: We now turn to the goodput achieved by transferring data using Reads vs EPC messages. There are two factors to consider: a) Read messages require more time due to additional rounds of initiation, and b) Read messages are longer than EPC messages and experience different loss rates.

First, we look at the break-even point between EPC and Reads due to protocol overhead. As shown in Table 1, the overhead for setting up each read command is 11.425 ms whereas the overhead for setting up a Query, QueryRep or QueryAdj is 4.762 ms, 4.05 ms, or 4.193 ms respectively. Thus, assuming that both Reads and EPC messages experience the same loss rate, a read message would need a payload larger than 30 bytes to obtain the same goodput as of that using EPC messages.

We now empirically validate these results. In Figure 2, we plot measured goodput of standard EPC Gen 2 messages compared to the goodput of 12 byte Read payloads. These experiments were conducted within line of sight of an Impinj Speedway reader across varying distances. Our results match the performance predicted by our previous calculations, showing that Reads consistently get only half the throughput of EPCs for similarly sized messages.

Longer read messages: While using long read messages might seem like a solution, we run into several other considerations that make this difficult to implement. On an Intel WISP, we found that reliability dropped to be close to zero once the read payload exceeded 16 bytes. While lack of visibility on the reader side makes it difficult to precisely attribute the sharp goodput drop to a specific cause, we hypothesize that this results from a combination of channel error and timing drift for long messages. In addition, message length has a dramatic impact on loss rate — even a 12 byte message incurs about a 10% loss rate on a backscatter link; increasing the message size by 5× to get to the breakeven point for a read command is certain to dramatically increase losses.

In conclusion, these results show that the Read command is a bad option for designing a data transfer protocol from CRFIDs to a reader, and implies that we should build a protocol using EPC messages.

3.2 EPC Protocol Inefficiency

The Gen 2 RFID protocol is designed around inventorying large numbers of tags that need only report a static identifier. It is therefore primarily focused on collision avoidance when there are large numbers of passive tags.

A key parameter that controls the efficiency of the EPC Gen 2 protocol is the window size, Q. The window size is a parameter that is set by the reader based on the tag population that it observes, as described in §2; during a round a number of slots are chosen such that the probability of collision between two tag responses is negligible. The EPC Gen 2 standard provides some general guidelines as opposed to a specific algorithm for how to set Q, so the specifics of the algorithm depends on the vendor and is not available to us. In addition, there is often no way to control the Q values set by a reader since modern RFID readers are designed with convenience of operation in mind and abstract low-level protocol parameters from the operator. This means that the window size, Q, cannot be controlled through the exposed API, which in the case of the reader we used was the Low Level Reader Protocol (LLRP).

The efficiency of the EPC protocol clearly depends on the value of Q set by a reader in each round. For example, if a reader picks $Q = 3$ and there is only one tag present, then there are 8 slots in a round, including the Query, of which only one slot is responded to by a tag. In addition to the obvious throughput inefficiency, this is also inefficient energy-wise as a tag incurs the energy overhead of listening to the QueryRep or QueryAdjust messages for the empty slots, which incurs non-trivial energy cost as shown in Table 1.

To quantify EPC Gen 2 inefficiencies, we place a single tag (Intel WISP) programmed to respond with its EPC code at two different distances from the reader (1m and 7m). We then measure the number of Query messages, QueryRep messages, and QueryAdjust messages received by the tag. Clearly, the throughput and energy efficiency is maximized if the reader chooses $Q = 0$ since there is only one tag, and this avoids any wasted slots. Figure 3 shows a breakdown between the number of Query, QueryRep, and QueryAdjust messages received by the tag at the two distances; if $Q = 0$, the tag would only receive Query messages, therefore the fraction of other messages received gives us an estimate of the inefficiency.

The results show that only about 65% of the messages are Query messages, and the remaining messages are slots wasted for QueryRep and QueryAdj messages due to the reader choosing $Q > 0$. A log of the Q value chosen by the reader shows a dis-

triution between 1 and 6, with a mean of 2.5. Based on numbers from Table 1, we can see that this translates to a 33.2% reduction in throughput and 17.5% increase in energy consumption. This design is unfortunate for CRFIDs that may have large amounts of data to transmit. Because reader contact events last only a few seconds, each communication slot should be utilized to send data. If tags implement the protocol verbatim, throughput and energy-efficiency will be poor.

Thus, one of the central challenges that we face is designing an efficient burst transfer protocol while not being able to control low level operational parameters within the RFID protocol.

3.3 Asymmetric Link Quality

The RFID communication channel consists of two components: the forward link (reader to tag) and the backscatter link (tag to reader). One communication round consists of multiple message exchanges in both the forward and backscatter links, as shown in Figure 1. An understanding of the forward and backscatter links is critical to a link quality estimation algorithm that is at the core of any data transfer mechanism.

Intuitively, it would appear that the backscatter link is the bottleneck for an RFID data transfer protocol. The signal to noise ratio (SNR) for typical backscatter communication decays with the square of distance for the forward link and to the fourth power of distance for the backscatter link [19]. The significantly lower SNR at the reader would suggest that the backscatter link should be the key emphasis for a link quality estimation mechanism for CRFIDs.

But there are two other factors to consider: a) The antenna sensitivity at the receiver, and b) The modulation and encoding scheme chosen for the communication links.

Antenna sensitivity: To understand antenna sensitivity, we compare an Impinj Speedway RFID reader and an Intel WISP as the tag. The Impinj reader uses a mono-static antenna for sending and receiving data, which has a sensitivity of -80 dBm. In contrast to the highly sensitive antenna used by the reader, the WISP uses a dipole antenna for data transfer because of its simplicity. The dipole antenna found on the WISP comprises two horizontally aligned copper wires, embbeded in a printed circuit board. While it is hard to measure the precise sensitivity of the WISP antenna, it is clear that the sensitivity of the Impinj reader's mono-static antenna is much higher than the dipole antenna of WISP, which causes asymmetry in the perceived signal of the forward and backscatter links. Because the reader has a more sensitive antenna, it can receive a backscatter signal that has a considerably lower SNR than what the RFID tag can decode.

PHY encoding: In addition to different antenna sensitivities, the physical layer encoding scheme also has significant impact on channel quality. As mentioned previously in §2.2, pulse interval encoding (PIE) is used in sending information from the reader to the tag. While PIE is simple to implement on a CRFID, it lacks robustness in dealing with interference. If the width of a pulse is smaller than the defined pulse width because of interference, the decoding process will fail and result in an increase of the overall bit error rate.

In contrast, Miller-4 encoding is used for the backscatter communication link. signal voltage levels to distinguish a logical 0 and 1. As compared to PIE, more pulses are required to encode one bit of information. In addition, multiple rounds of voltage detection are required, which introduces additional complexity and energy consumption. Considering the additional level of coding, the backscatter link should have higher link quality than the forward link, because of its insensitivity to pulse timings.

Comparison of forward and reverse links: To provide an empirical comparison of the effects of the above-described factors, we now measure the packet loss rate of the forward and backscatter link at different distances. A 22 bit query message is used to measure the packet loss rate of the forward link and the 12 byte EPC code, which includes 2 bytes of header and a 2 byte CRC, is used to measure the packet loss rate of the backscatter link. Although the EPC response is much longer than a query message, as shown in Figure 4, the loss rate of the forward link increases with distance while the loss rate of backscatter link remains below 10% across distance. Considering the relative link qualities of the forward and reverse channel links, the bottleneck of goodput in our system lies in the forward link. This observation impacts our system design, as we focus our efforts on measuring the forward communication link and assume the backscatter link is strictly superior.

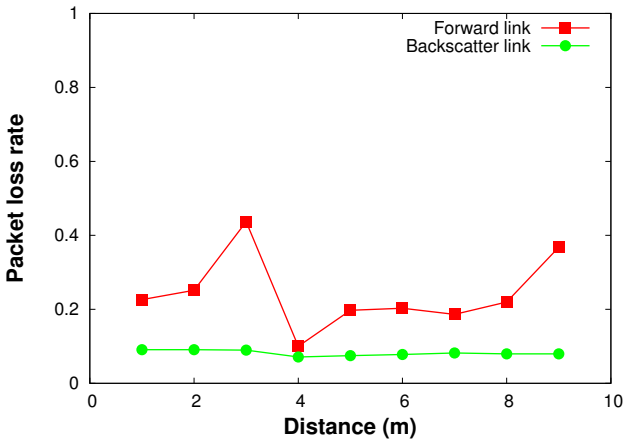

Figure 4. The bottleneck of communication for the Intel WISP is the forward link, whose loss rate increases with distance. The backscatter link loss late remains constant as a result of reader antenna sensitivity.

4 Design Methodology of a Burst Protocol

There are a number of factors to consider when designing a data transfer mechanism around the use of EPC messages. A protocol built around the EPC message response as a communication primitive must strive to: 1) maximize data transfer rates so that sensor tags can transfer their data quickly and efficiently to a reader during short contact events, 2) minimize power consumption so that an CRFID sensor can maximize the amount of data transferred using the small energy buffer, and, and 3) inter-operate with standard commercial RFID readers, so that our tags can leverage existing RFID reader infrastructure.

To address these goals, we present the design of a burst protocol for CRFID sensors. First, we demonstrate how sensors can achieve high levels of goodput using burst-mode data transfer that leverages unused slots in the EPC Gen 2 protocol. Second, we show a coordination mechanism that uses burst notifiers to avoid collisions among bursting tags. Third, we present a duty-cycling mechanism that minimizes the energy lost to idle listening. Finally, we discuss implications of the design choices that we make when there are a mix of sensor tags that are bursting and standard EPC Gen 2 tags that are only transmitting their identifier.

4.1 Burst-Mode EPC Transfer

Previously, we showed that a single tag responding in only one slot within a round of communication achieves goodput proportional to the number of slots chosen by the reader. Thus, a key question is: *how can we improve protocol efficiency when a tag needs to respond across several rounds of communication*. This is particularly important since messages are limited to 12 byte payloads, necessitating a large number of rounds before several hundreds of

bytes of sensor data can be transferred.

To address the problem of minimizing protocol efficiency, we use a simple approach wherein a CRFID sensor uses all slots within one round of communication. In other words, a tag responds to each Query, QueryRep, and QueryAdjust message regardless of the value of its slot counter. We now discuss the implications of such a strategy on energy-efficiency and goodput.

Implications on energy-efficiency: To understand how energy efficiency is affected by the number of slots reserved for communication, we must understand the implications of a single tag responding within all communication slots. As specified, a reader implementing the EPC protocol expects an individual tag to occupy a single communication slot. As a result of seeing many slots filled by different EPC messages, the reader decides that many tags are present and increases Q to avoid collisions amongst the perceived tags. The reader will continue to increase Q to its maximal value of 15, as it always finds the slots it previously allocated filled to capacity.

One may expect that causing the reader to react in such a way could result in confusion and result in poor efficiency; however, we argue that increasing the number of communication slots also increases energy efficiency. We justify this claim using Table 1. As we previously showed, a query message will be followed by a combination of $2^Q - 1$ reps and adjusts. A subtle benefit of QueryReps and QueryAdjs, as opposed to queries, is their brevity. Based on protocol specs query, rep and adjust have lengths of 4, 9, and 22 bits respectively. As Q grows, the energy expended during a round of communication becomes dominated by round trips involving reps and adjusts; this cost is quantified in the following equation:

$$\mathrm{E}_{\mathrm{round}} = \mathrm{E}_{\mathrm{query}} + \left(2^{Q+A} - 1\right) \cdot \mathrm{E}_{\mathrm{rep}} + n \cdot \mathrm{E}_{\mathrm{adjust}} \qquad (1)$$

Equation 1 shows that the total energy spent on a round of communication ($\mathrm{E}_{\mathrm{round}}$), is the sum of the energy spent on listening and replying to queries($\mathrm{E}_{\mathrm{query}}$), reps($\mathrm{E}_{\mathrm{rep}}$), and adjusts($\mathrm{E}_{\mathrm{adjust}}$). In the simplest case, the reader assigns Q within the intial query message, followed by the corresponding number of reps; however, the reader may decide to send an adjust message to increment or decrement Q after a round has already begun. In Equation 1, n represents the number of adjust messages sent, while A indicates these messages' net impact on Q.

Taking into account the bits required to decode the initial command (Query, QueryRep, or QueryAdj), the RN16 value transmitted and received by the tag, and finally the transmission of the EPC code itself, we find that rep and adjust round trips take 90.2% and 92.9% of the energy of a query round trip. Thus, for sensors that cause Q to become large by adopting burst transmissions should potentially acheive 9.8% energy savings as compared to a round containing only a query.

Implications on Goodput: A similar equation may be derived that quantifies the goodput, in bytes per second, of the channel as a function of the distribution of messages sent by the reader within a round:

$$\mathrm{Goodput}_{\mathrm{round}} = \frac{12}{\mathrm{Time}_{\mathrm{query}}} + \frac{12 \cdot \left(2^{Q+A} - 1\right)}{\mathrm{Time}_{\mathrm{rep}}} + \frac{12 \cdot n}{\mathrm{Time}_{\mathrm{adjust}}} \qquad (2)$$

By similar analysis, we look at the distribution of queries, reps, and adjusts within a round; this time, we take the 12 bytes of data transmitted by the sensor (Length of EPC message) and divide by the total time each message type requires. Because inter-frame times are fixed independently of Q, the shorter messages will result in slightly improved goodput. For large Q, sensors using burst transmissions can achieve a potential 8% higher goodput over a round containing only a query.

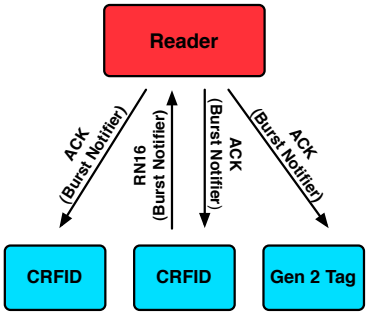

Figure 5. A tag that wishes to send a burst sends a burst notifier as its RN16. The reader broadcasts this value to all tags in range.

4.2 Coordination via Burst Notifier

Responding in every slot has a severe limitation: if more than a single sensor is present, many sensors will suffer from collisions and see reduced energy efficiency and goodput, instead of the improvements we previously quantified. To address this issue, we design a coordination mechanism that avoids most collisions introduced by our burst-mode data transfer mechanism. Because RFID sensors are highly constrained in terms of hardware resources and energy, this coordination strategy needs to be effective while remaining simple. The primary challenge for such a coordination scheme is staying within the constraints of the Gen 2 protocol so existing hardware may be used for implementation. Thus, we ask the question: *How can CRFID sensors use the existing EPC protocol to efficiently coordinate bursts?*

A system using an active radio could solve this problem using control messages such as RTS/CTS or an overhearing-based approach such as CSMA to coordinate transfers between peer nodes. These approaches are not suitable for backscatter communication circuits found in RFIDs because RFIDs are unable to decode messages originating from peers. Therefore, any co-ordination mechanism should rely on the reader, and in particular, should use an existing message from the EPC Gen 2 protocol.

Due to this limitation, we use an approach designed to be compatible with EPC Gen 2. As we showed in Step 3 of Figure 1, the reader echoes the RN16 of the RFID it chooses to occupy a particular communication slot. Our strategy is to overload the RN16 to signify that a particular CRFID is currently bursting. We accomplish this by providing a special interpretation of a segment of reserved RN16s; we partition the space of RN16s as $0 < n < 2^{16}$, where n is the number of CRFID sensors deployed and values less than n are considered burst notifiers. A sensor that wishes to send a burst of EPC messages will use its statically selected burst notifier chosen from the available pool, instead of a random value. This mechanism is illustrated in Figure 5, where we show that a burst notifier sent by a CRFID is overheard as the reader's acknowledgement by other tags in the reader's RF field. Note that the sensor selects a notifier just once for an entire burst, rather than once per slot as is done by a standard tag.

The RN16 burst notifier is used in the following way: prior to initiating a burst, a CRFID sensor listens to the channel after decoding a query, rep, or adjust message. If the sensor observes an RN16 within the range of burst RN16s, it should remain silent to avoid colliding with an ongoing burst. If the slot contains an RN16 outside of this range, it can go ahead and start a burst transfer after the current round using its own burst notifier, as non-burst EPC

messages occupy only one slot.

It is, of course, possible that another CRFID sensor is in the middle of its burst and either the reader might have missed the burst notifier or the listening sensor might have not received the notifier echoed by the reader due to channel error. Both cases would lead the listening sensor to conclude that the channel is free and start to burst, resulting in collisions at the reader. A collision at the reader typically results in the reader receiving the stronger signal among the colliding tags due to capture effect. The reader echoes the burst notifier that it receives, which results in only the sensor with stronger signal continuing to burst. While a collision could also result in neither signal being received by the reader, we find that this is extremely uncommon due to the high sensitivity of the reader antenna.

To prevent the sensor from holding the channel indefinitely, the burst will terminate after a small, fixed amount of time; this time should be set to a value large enough to amortize coordination overheads, but small enough to allow mobile tags with limited communication opportunities a chance to offload a burst of data to the reader.

4.3 Duty-cycled Coordination

Employing burst notifiers minimizes collisions between bursting nodes, but a burst tag expends energy by listening idly during another node's burst. Because nodes will occupy hundreds of slots during a single burst, they can incur significant overhead and is a problem that needs to be addressed.

The evident solution to this problem is to duty-cycle RF subsystem of a sensor tag when it is waiting for another tag to complete its burst. The RF subsystem comprises two components: a) the analog comparator that senses the channel to detect the presence of a bit, and b) the microcontroller that wakes up upon each interrupt from the comparator to process the bit and check if a valid message is present. The duty-cycling strategy is straightforward — shutting off the comparator avoids any energy lost from responding to interrupts and therefore idle listening.

If a waiting sensor tag knew precisely how much longer a burst from another tag would last, it could sleep for exactly that duration. However, this information is unavailable since a sensor tag relies on the burst notifier from the reader to detect a burst, which provides no information on the time remaining for the burst. Thus, a tag needs to periodically wakeup to check the channel and detect if a burst has ended. Thus, a key challenge for a duty-cycling strategy is to efficiently find the end of a burst so that sensors can capture the channel from another sensor between bursts and react quickly to mobility dynamics while avoiding most of the energy wastage caused by overhearing.

Thus, there are two questions that remain regarding how to duty-cycle an CRFID sensor: a) how much amount of time a sensor should probe the channel and b) how much time a sensor should sleep.

Probe Duration: The probe duration should be long enough such that a tag can detect whether another tag is continuing to burst. This duration is equivalent to a single slot in a query round. A sensor tag wakes up, listens to the first Query, QueryRep, or QueryAdjust slot, and sees whether a burst notifier is echoed by the reader during this slot. If so, it concludes that another tag is bursting and goes to sleep; if not, it concludes that it can initiate its own burst and starts transmission in the next slot. In the middle of a burst, if a tag detects that the reader has echoed a different burst notifer it concludes that another CRFID sensor is bursting and goes to sleep to save energy.

A potential issue here is that the the duration of a slot can vary because a) Query, QueryRep, and QueryAdjust messages are of different lengths, b) a slot can terminate at different times depending on whether the reader times out after the RN16, Ack or EPC steps

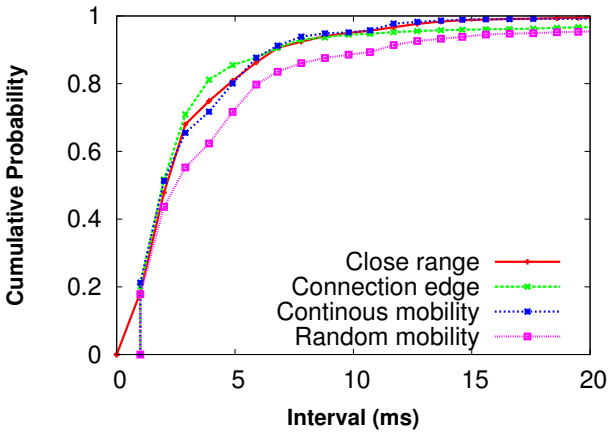

Figure 6. The vast majority of valid message frames arrive within intervals smaller than 5 ms for a variety of mobile and static communication scenarios.

in its state machine and c) mobility can introduce additional dynamics. To address this, we look at the probe duration empirically by measuring the duration between Query, QueryRep or QueryAdjust messages for a continuous exchange between an Intel WISP programmed with the EPC Gen 2 protocol and a reader. We look at this distribution for different distances from the reader, as well as for different mobility patterns. Figure 6 shows the CDF of the inter-message duration.

The results show that the inter-query intervals do not depend significantly on the distance, and are impacted a little ,but not a lot, by mobility. The knee of the curves are in the 3-6ms range; we pick 5ms as our probe duration since in 90% of the cases, this results in a query being received by the CRFID.

Sleep Interval: There are several considerations in determining the comparator sleep interval. First, the sleep interval must be long enough that we get significant energy benefits from duty-cycling. Second, it should be short enough that a tag can quickly react to mobility-induced channel dynamics. Third, it should have sufficient randomization so that we avoid unwanted synchronization issues that can result from multiple tags waking up at the same time.

In terms of the energy consumption, we want a duty-cycle of lower than 10%, hence the sleep duration should be at least 50ms when the probe duration is 5ms. To understand the typical contact duration at walking speed, we use 1 reader in a corridor, and walk in circles around it. The resulting contact duration CDF is shown in Figure 7. As can be seen, a typical contact duration is a few seconds, hence the sleep duration should be much smaller than this number. To prevent synchronization issues, the tag can randomize the sleep time within a tolerable range that provides a desired amount of energy savings and reactivity to expected mobility patterns for a given deployment.

4.4 Co-existence with Non-burst Tags

While our discussion thus far has assumed the tag population comprises solely of sensor tags that have to transfer data in a burst, we now look at the implications when a mix of sensor tags and standard EPC Gen 2 tags are communicating with the same reader infrastructure. Not surprisingly, the net effect is that standard EPC Gen 2 tags incur more delay in communicating with a reader infrastructure. However, there are mitigating factors that can enable better co-ordination across tags.

The burst mode transfer mechanism that fills up all slots of a round impacts standard Gen 2 tags in two ways. First, a standard

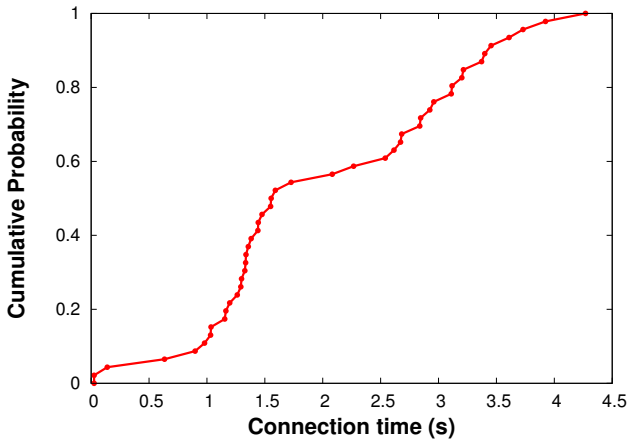

Figure 7. For human-scale mobility rates, the connection time between a tag and reader typically lasts several seconds.

tag which picks a slot within a round will collide with a burst tag, resulting in loss of one of the messages. In practice, we find that because of reader sensitivity, the reader gets one of the CRFID's messages with high probability (capture effect), hence it is still possible that the standard tag gets its data through. However, if the burst tag has the stronger signal, the standard tag suffers. Second, the burst transfer approach results in large Q values, which makes a round long; since standard tags only respond in one slot within a round, this makes their response slow. This effect is mitigated by the fact that a burst lasts only for a short duration of time (1s in our implementation), after which a tag goes to sleep for a short window of time before bursting again. The duration between bursts is sufficient for a few short communication rounds, enabling standard tags to get their data through.

The use of burst notifier facilitates co-ordination across CRFIDs, however, passive tags are free to choose any value from 0 to 2^{16} as its RN16, so it is possible that a passive tag could choose an RN16 that conflicts with a burst notifier. However, we choose a small part of the space for burst notifiers since we expect the number of sensors in the vicinity of a reader to be in the tens (equivalent to the number of objects in the vicinity of a reader) as opposed to thousands. Thus, the probability of collision is low. In addition, the RN16s are chosen anew in each round, hence a standard tag would likely choose a non-colliding RN16 in the next round.

5 Implementation

Our bulk transmission protocol is well suited for implementation on CRFID sensors because it is a modification of the EPC Gen 2 protocol they already support. CRFID sensors that want to implement the protocol need only modify their state machine to properly handle burst-mode transmission, burst notifers, and duty cycle appropriately in response to received message frames from a reader. In this section, we show the state machine we used to implement our Bulk Transmission Protocol for the Intel WISP. Next, we describe how state machine parameters can be defined based on results from channel measurements and mobility experiments. Finally, we describe the methodology used for performing channel measurements, as well as the strategies used for testing and evaluating our firmware modifications.

5.1 State Machine

Figure 8 shows the state machine used to implement our Bulk Transmission Protocol for the Intel WISP. Sensors that have data to send initialize a timer interrupt and begin operation in the *Sleep* state. After this timer expires, the sensor activates its comparator and enters state *Frame Check*; after initializing another timeout value, the microcontroller enters a low-power mode, only waking up to handle an incoming message frame. Upon receiving a valid message frame, the sensor will enter state *RN16 Probe*; else, if the sensor does not hear a valid delimiter from a reader, it goes back to state *Sleep* after timing out. While in state *RN16 Probe*, the sensor initializes its timer with another timeout value; after hearing at least one empty frame from the reader, in which the sensor does not hear another sensor's BURST_NOTIFIER, it will send its own BURST_NOTIFIER in response to a Query, QueryRep, or QueryAdjust message. If the sensor hears its own BURST_NOTIFIER, it enters state *Burst*; if another sensor's BURST_NOTIFIER is heard, instead of an empty slot or if the timeout value is reached, the sensor enters state *Sleep*. Upon entering state *Burst*, the sensor will again initialize a timer, then begin transferring the contents of its buffered data as an EPC message in response to Query,QueryRep, or QueryAdjust messages; the sensor uses its BURST_NOTIFIER to send every message within the burst. Upon completion, timeout, or detecting 4 slots during which it finds no acknowledgement, the sensor will disable its comparator and return to state *Sleep*.

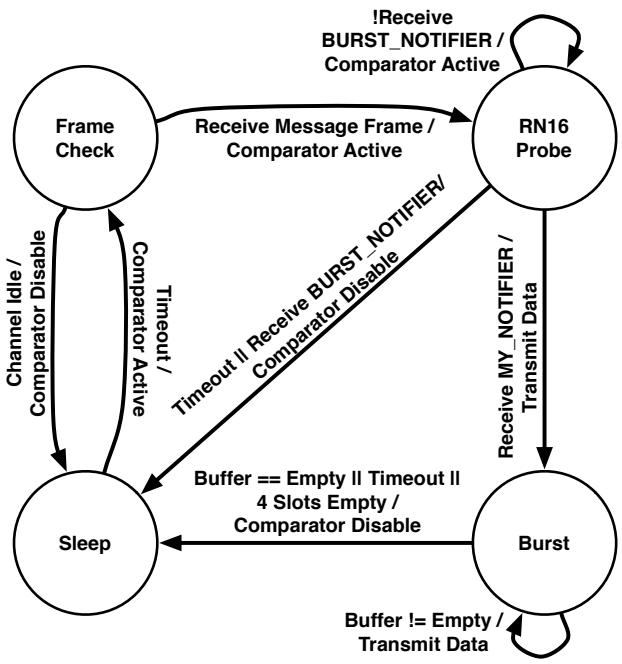

Figure 8. A coordinated bursting protocol for CRFID sensors can be implemented as a state machine. The protocol uses sleep states to both avoid contention and achieve energy efficiency.

5.2 Parameter Selection

While the state machine we previously described is a useful framework to constuct our protocol, it would not function well if the parameters were blindly selected without regard to real system constraints or channel characteristics. Here we provide some intuition in how to choose parameters based on experience with a real implementation:

Timeouts: The timeout values used in our state machine are chosen based on Figures 6 and 7 in §4 that give good insight into expected connection intervals and message inter-arrival times respectively. In practice, these timeout values are used as comparison values for Timer A_1 on the WISP's MSP430 microcontroller. When considering hardware constraints and initialization overheads, one

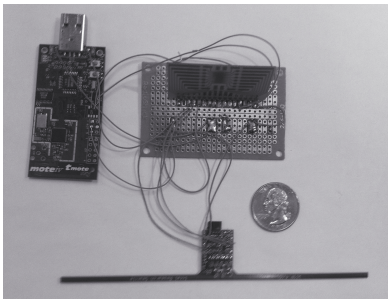

Figure 9. A Telos mote is used to gain visibility into communication measurements obtained from a mobile WISP

must also be careful to not choose a set of timeout values that generate too many interrupts that interfere with the WISP's ability to timely respond to reader messages. In practice, timeout values > 2 ms give the WISP sufficient time to listen for messages, while also providing the time needed to for timer initialization.

Burst Length: The maximum time that allow a WISP to continuously burst data in reader slots is another parameter that depends on hardware constraints. When considering how to bound the amount a tag can burst, we considered two strategies: one that bounds the total amount of data to burst, and another which bounds the time a tag can burst. Beause tags can see vastly different versions of the wireless channel, in practice the amount of time required to send a fixed amount of data varies widely. In a benchmark experiment we found that it took anywhere from 7.1 and 11.7 seconds to send 12kB of data. Conversely, by fixing the amount of time a tag can burst causes the amount of data sent will vary. Because tags use time constants to coordinate, it is important to keep the amount of burst time fixed to minimize the likelihood of overlapping bursts. Although this could cause different tags to deliver data at different rates, they will still get equal opportunities to use the channel and their data delivery rates will vary according to their respective channel conditions.

Burst Notifiers: The final parameter we consider is the burst notifier used by tags to coordinate their burst transfers. When modifying the WISP firmware, we found it can be difficult to get the state machine to stay within the tight timing constraints specified by EPC Gen 2 protocol. Complex operations in the firmware will diminish responsiveness and in the end manifest as a reduction in goodput. For example, choosing a poor ordering of comparisons while looking for a burst notifier can lead to a 30% reduction in goodput. We also found that messages sent from the reader to the WISP can contain bit errors; one example is the RN16 field, which in actuality contains only 15 bits of consistent data.

5.3 Debugging and Evaluation Methodology

Finding a set of good parameters for use in our protocol is part of a challenging development and evaluation cycle because of the Intel WISP's limited visibility, energy, and tight timing constraints. We highlight how we overcame each of these challenges below:

Visibility: To overcome visibility issues, a WISP can be tethered to a JTAG debugging tool to observe its internal state. In many mobile scenarios, JTAG tethering in not a viable option; in these cases, software state of interest can be transmitted to a reader as an EPC code. To understand timing-related phenomena, such as inter-message times, Timer A1, which is unused by the firmware, can be used to capture the intervals which again are sent as EPC messages. In many cases, we found the cycle overhead in initializing a timer when decoding reader message frames was large enough to violate the timing requirements of EPC Gen 2 and result in the reader being unable to decode tag message frames. To solve this particular problem, we use a Telos mote [3] to count the time between two GPIO interrupts generated by the WISP and log the resulting timing information to its internal flash using a simple application written for the TinyOS-2.x [12]. The Telos platform is well-suited for the scale of timings typically present between protocol messages; the platform itself does not significantly impact mobility because of its small size.

Energy Limitations: Evaluating a burst mode transfer mechanism with a limited energy supply is difficult. Because our work assumes tags have some amount of buffered energy when initiating contact with a reader, it is difficult to refresh a capacitor-based energy store to a consistent level while running experiments; this is especially bothersome when multiple tags are involved. To remove measurement dependencies on harvested power, we power tags directly with batteries; this setup assumes they have a plentiful store of energy while in contact with a reader. Seperately, we collected a series of energy benchmarks that carefully measured the amount of energy required for sending and receiving protocol messages and quantified non-trivial computation overhead. By combining these two mechanisms in our evaluation, we were able to design a burst mechanism that minimzes energy consumption, while adapting accurately because performance measurements depended only on channel conditions and not energy.

Channel Measurements: Another evaluation challenge lies in only having the ability to measure the channel indirectly. The protocol used to communicate with the Impinj reader, LLRP, does not provide an interface to directly modify and observe messages or set message rates sent by the Impinj reader. This makes it impossible to get a completely accurate picture of channel characteristics, such as loss rate, from the reader to a tag. To overcome this obstacle, we embed counters that indicate the number of Query/QueryRep/QueryAdjust messages received into EPC messages, giving us limited visibility of the forward link. These message counters assist in determining the rate of messages sent in the forward link, as well as the contribution each type of message has on data delivery. Because an EPC code transfer happens directly after receiving ACK messages from the reader, the ratio of ACKs to EPC codes received at the reader gives us a measure of the losses present in the backscatter link. Additionally, the counter information we send allows us to align messages recorded at the reader, with GPIO timing triggers on the Telos mote.

6 Evaluation

In this section, we evaluate the implementation of our bulk transmission protocol. The evaluation consists of four parts: 1) quantifying the goodput achievable by burst-mode EPC transfer, 2) showing that our burst notifier based coordination mechanism retains most of the goodput achieved by bursts by avoiding collisions, 3) demonstrating the energy benefits of duty-cycling, and 4) evaluating the interaction between bursting tags and standard EPC tags.

6.1 Burst mode transmission

The burst mode transmission protocol that we described in Section 4 ensures that all slots created for a round of communication are utilized by CRFIDs. In this section, we evaluate our burst transmission protocol in three ways: 1) We measure the window size allocated by Impinj reader when burst mode transmission is utilized, 2) We evaluate the goodput benefit gained by burst mode transmission, and 3) We provide a breakdown to show where the goodput benefits comes from.

Window size: In §4.1, we argued that the burst mode transmission strategy results in an RFID reader choosing a large window size (Q value) within a round of communication, leading to greater opportunities for using shorter messages, such QueryRep or

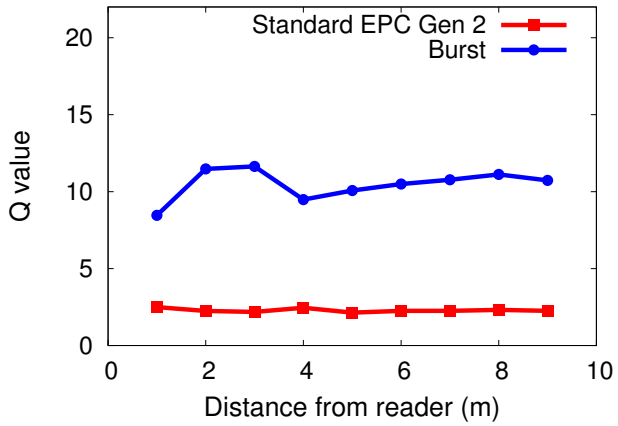

Figure 10. Window size parameter Q is heavily impacted by burst mode operation. The reader chooses larger Q in response to bursting tags, leading to more opportunities to use shorter messages for transfer.

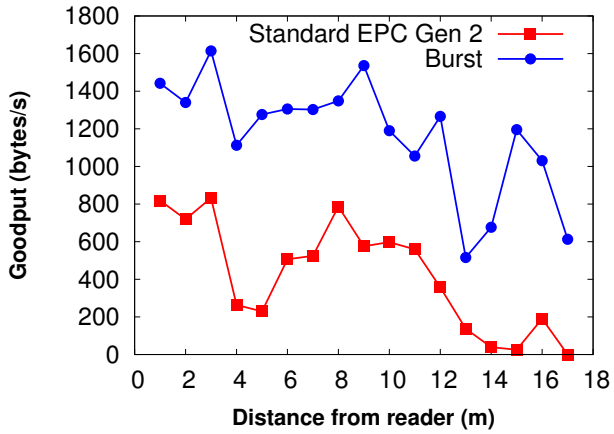

Figure 11. RFID-scale sensors can achieve a 60% improvement in goodput by utilizing all slots in a communication round. The amount of improvement that can be extracted depends on the window size chosen by the reader.

QueryAdjust, for data delivery. To validate this argument, we design an experiment that compares the Q value chosen by an Impinj reader for standard EPC transfer vs burst transfer. Our experiment setup places an Intel WISP in line-of-sight of an Impinj reader's antenna. The WISP piggybacks the Q value in place of the EPC code that it backscatters to the reader.

Figure 10 shows that the Impinj reader consistently selects a considerably larger Q value ($Q \approx 10$) for burst transmission independently of distance. As described in §4.1, a large Q value is good for burst transmission. In contrast, the Impinj reader selects $Q \approx 2$ when the standard EPC Gen 2 protocol is used. Any Q value larger than 0 leads to un-utilized slots, therefore, this choice results in a significant fraction of wasted slots.

Goodput: To quantify how burst-mode transfer of EPC codes better utilizes slots, we design an experiment that compares the goodput of a burst-optimized version where a tag responds within every slot versus EPC Gen 2. In this experiment, we measure the goodput of an Intel WISP tag programmed to act as a conventional EPC Gen 2 tag vs a WISP programmed for burst mode operation. We log the average throughput observed by the reader at different distances for several minutes and compare the results. Figure 11 shows that burst mode communication achieves 60% higher goodput than standard EPC Gen 2, and that these benefits are sustained across different distances. It is also notable that, in practice, the benefits of burst transfer are considerably larger than those that we predicted in §4.1. These additional gains are a result of the reader over-allocating slots for passive tags, resulting in comparatively low goodput.

Goodput breakdown: The increase in goodput for burst transmissions stems from two factors. First, we utilize every slot rather than one slot in each inventory round to transmit data. As shown in Figure 12, a standard EPC Gen 2 protocol only utilize only 64.84% of slots in each inventory round at 1m and 68.05% at 7m since it responds only to either a Query message or a QueryRep message but not both. In contrast, a burst sensor tag utilizes 100% of the slots. Second, in burst mode, we get more opportunities to exploit shorter inventory messages, QueryReps and QueryAdjusts, for data delivery. Shorter messages in the forward link have lower loss rates; this leads to tags succesfully capturing slots for data transfer with higher probability. For large Q, communication is dominated by slots that are initiated by QueryRep and QueryAdjust messages; as shown

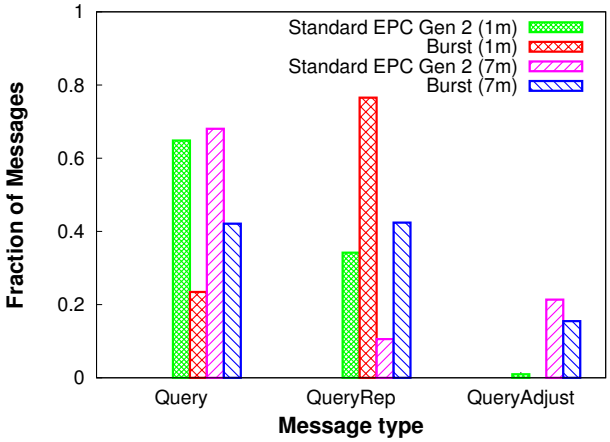

Figure 12. Bursting CRFIDs cause the number of slots in a round to increase. We observe this as an increase in the fraction of QueryReps received.

in Figure 12, the percentage of QueryReps at 1 m while bursting increases to 76.56%, as compared to the 34.18% slots in standard EPC Gen 2. A similar breakdown holds true for QueryReps at 7 m and QueryAdjusts across both distances. With more QueryRep and QueryAdjust messages exploited, tags get more opportunities for data transfer that contribute to higher goodput.

6.2 Coordinating bursts

While burst-mode transfer provides significant improvements to goodput, it also introduces problems when multiple RFID sensors wish to use the channel simultaneously. In this experiment, we evaluate how much the co-ordination mechanism benefits goodput when multiple tags are transmitting.

We consider a baseline case when a single CRFID is present, a low-contention case when three CRFIDs are simultaneously transferring data and a high-contention case when five tags are simultaneously transferring data. All tags are placed in a linear configuration parallel to the Impinj reader within line-of-sight of the antenna at a distance of 1 meter. Figure 13 shows a breakdown of the goodput for three protocols: a) standard EPC Gen 2, b) Burst mode without coordination, and c) Burst mode transfer with cooordination.

Single node case: First, we look at the case where there is a single tag. We see that the standard EPC Gen 2 tag performs significantly worse than the bursting sensor tags, as expected. We also see that the co-ordination scheme performs about 9.17% worse than the case without co-ordination. This is because of the overhead required for coordination. After finishing a burst transmission, a coordinated tag releases the channel for a 50 ms to allow other waiting tags to acquire the channel, which reduces goodput. Also, the code required to check for burst notifiers results in slightly slower reaction times of the state machine running on Intel WISP which occasionally results in violating the timing requirements specified by Gen 2. We believe that the latter problem can be overcome with a careful re-structuring of the Intel WISP radio stack to minimize timing jitters.

Three node case: When we increase the tag population to three, we see a similar behavior in terms of overall goodput, but the split across nodes is very different. Overall, we see that burst mode transfer with coordination is still a little lower (7.67%) in total goodput than the uncoordinated case. However, the un-coordinated case gives out more than 87.3% of the channel to one of the three tags. The higher goodput achieved by one of the tags is a result of the capture effect. This tag's RSSI is 7.5 dBm higher than other tags in the experiment, and it therefore dominates transmissions when there is no co-ordination mechanism. Because of their lower RSSI values, other tags cannot capture the channel effectively, leading to highly skewed goodput distributions across the three nodes. The burst protocol with co-ordination is considerably fairer — all three tags receive a good chunk of the overall goodput.

Five node case: When the tag population reaches five, the impact of collisions becomes significant and adversely impacts the performance of the non-coordination mechanism. For bursts without co-ordination, the total goodput reduces by 65.7% as compared to the single tag and the three tag cases. In addition, the uncoordinated scheme is highly unfair and allocates 89.98% of the goodput to one of the five tags.

Table 2 shows the Received Signal Strength Indicator (RSSI) values logged at the reader for the backscattered data from the five tags, and how that influences goodput. Even though all tags are placed at the same distance from the reader with similar orientation, different tags observe different RSSI values as a result of small, uncontrollable deviations in antenna orientation. It can be seen that for uncoordinated bursts, the tag with strongest RSSI captures the channel entirely and completely swamps the other tags. One particular tag, Tag A, achieves the highest RSSI of -39 dBm, which in turn has the highest measured goodput of 540.5 bytes/second. The other four tags cumulatively get 10.02% of the total goodput, each less than 30 bytes/second.

In contrast, coordinating bursts results in much better performance as the tag population increases to five. The total goodput remains almost exactly the same as the totals for the three and one tag case, showing that co-ordination manages to avoid collisions and serialize transfers across nodes. The RSSI breakdown in Table 2 shows the co-ordination mechanism doesn't necessarily favor the tag with highest RSSI — in fact although Tag D's RSSI is lower than Tag A, it gets a large fraction of the goodput. Overall, the co-ordination mechanism also results in fairer allocation across nodes.

To understand the dynamics of co-ordination, and how frequently nodes switch among each other, we look at the a breakdown of the duration that each tag holds the channel in a burst. We define a burst period as a contiguous period when a single tag is bursting; at the end of the period, some other tag acquires the channel and starts bursting. These results are plotted in Figure 14. The results show that bursts are not always 1 second, even though that is the maximum length of a burst according to our parameter settings.

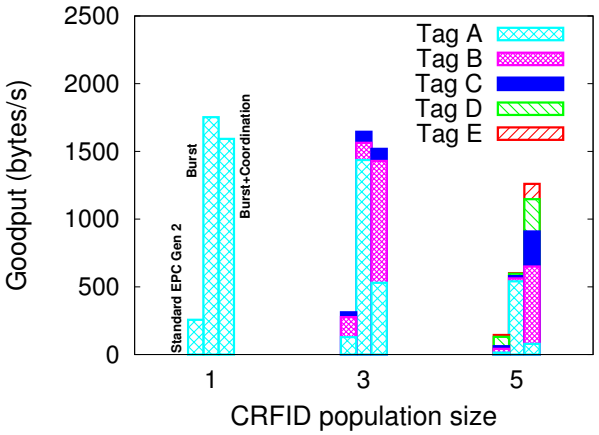

Figure 13. Coordination improves throughput and fairness as more bursting CRFIDs compete for the channel. For small tag populations, coordination incurs a small protocol overhead. burst

Tag ID	Coordination		No Coordination	
	RSSI (dBm)	Goodput	RSSI (dBm)	Goodput
A	-41.7	78.9Bps	-39.6	540.5 Bps
B	-37.7	571.1Bps	-45.1	21.5 Bps
C	-39.9	258.8Bps	-47.2	16.9 Bps
D	-46.0	238.2Bps	-45.9	17.8 Bps
E	-50.1	112.8Bps	-54.9	4.0 Bps

Table 2. Breakdown of the five tag results from Fig 13, showing RSSI and goodput for each tag for the two burst cases (with and without co-ordination).

In fact, nodes switch transfers at a considerably finer granularity. Such switching can happen for several reasons. Sometimes a query message is not received by a bursting tag leading to an empty slot; another tag which listens in this slot assumes that the channel is empty and starts bursting. This results in capture effect resulting in a switch between nodes. Despite such switching across nodes, we see that overall goodput does not suffer at all, and the protocol gracefully splits channel time across the five tags.

6.3 Coordination-Aware Duty-Cycling

Coordination allows tags to avoid collisions between each others bursts, but does nothing to prevent energy consumed due to idle listening. We now evaluate our duty cycling mechanism which duty-cycles the comparator to reduce energy consumed due to idle listening. Our evaluation answers two questions: a) how much energy is saved due to duty-cycling? and b) does duty-cycling result in degradation in goodput?

To quantify duty-cycling benefits, we use a simple experiment where five tags transmit a fixed amount of data of 3000 EPCs (36 kB) using different strategies. We then look at the energy consumed by the different strategies.

Figure 15 shows that duty-cycling reduces energy consumption by 22.13% to 45.58% across the five tags. The benefits are different across different nodes because they complete their transfers at different times and their idle listening overhead varies. Tag A finishes last, and therefore it has the maximum energy consumption, whereas Tag C finishes first and has the least energy consumption. The use of duty-cycling makes all five nodes have similar energy consumption since it dramatically reduces the idle listening overhead across the nodes. Thus, a CRFID sensor that employs duty-cycling can use its energy buffer more judiciously.

While duty-cycling improves overall energy consumption, a nat-

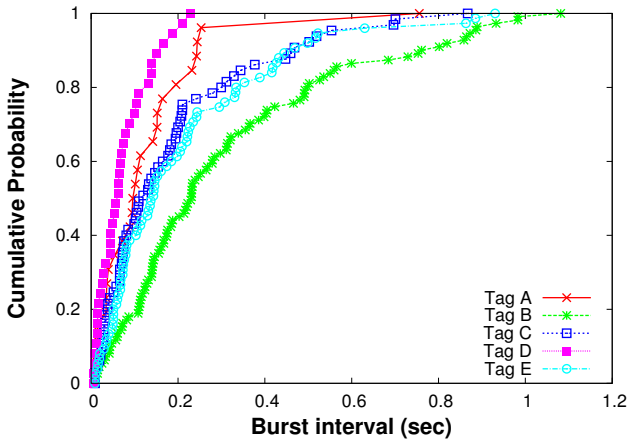

Figure 14. Breakdown of burst time for each of the five tags when coordination is used.

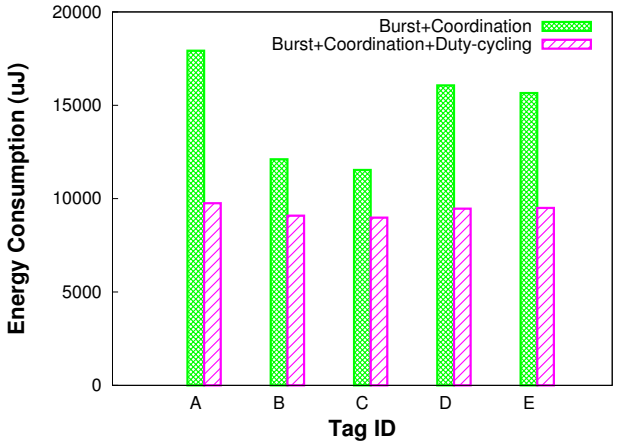

Figure 15. Energy benefits of duty-cycling by reducing idle listening overhead.

ural question is whether these gains come at the cost of a degradation in goodput. Figure 16 shows that duty cycle based coordinated bursting tags have the shortest time to finish data transfer, and therefore have the highest goodput. This demonstrates that our duty-cycling mechanism does not sacrifice goodput to achieve its energy gains.

6.4 Co-existence with passive tags

In this section, we investigate the interaction between our burst protocol and standard EPC Gen 2. We answer two questions when both bursting tags and standard EPC tags are transmitting to an Impinj reader: 1) How does the goodput of CRFIDs change when they compete for the channel with multiple standard EPC tags? 2) What is the time between inventorying standard EPC tag when CRFIDs are bursting? We setup experiments where both bursting tags and passive tags are collectively within the field of an Impinj Reader's antenna. We measure the goodput achieved by the tags that run the Flit protocol, and the interval required to inventory passive Gen 2 tags.

Goodput: Our goal in this experiment is to understand how increasing passive tag populations impact the goodput of bursting CRFIDs. Since we did not have enough Intel WISPs to use as passive tags, we use commercial passive tags for this experiment. We deploy five CRFIDs that are continually bursting to a reader, and increase the commercial passive tag population from 5 to 30.

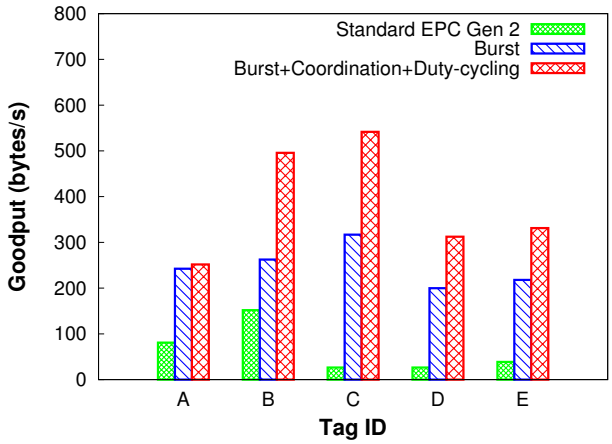

Figure 16. Goodput is highest (and time to completion lowest) for the duty-cycled transfer case.

Passive Tag Num	Avg Goodput (B/s)	Std of Goodput (B/s)
0	264.4	226.7
5	222.1	228.2
10	238.0	294.9
15	270.7	359.5
20	253.3	307.8
25	184.2	214.0
30	184.9	228.7

Table 3. This table shows the impact of increasing passive tag population on goodput of bursting CRFID tags.

All commercial passive tags and CRFIDs are placed 1 meter from reader and spread in a line parallel with the reader antenna. Table 3 shows the average goodput as the passive tag population increases. The results show that there is only a small effect until about 20 tags (the average goodput is 253.3 bytes/second which is only a bit lower than 264.4 bytes/second when there are no passive commercial tags). For larger populations of passive tags, there are more collisions in slots which impacts goodput of burst CRFIDs. However, we see that despite a relatively large passive tag population, CRFIDs continue to perform well while bursting.

Time between inventory: In this experiment, we look at how the time to inventory a passive tag is impacted by the presence of burst CRFIDs. We use two types of passive tags in our experiments, commercial passive tags and Intel WISPs which follow the EPC Gen 2 protocol. In the first experiment, 4 Intel WISPs are configured with Flit and one Intel WISP is configured as a passive tag. All tags are deployed 1 meter from reader and aligned in a linear arrangement, parallel to the antenna. As Figure 17 shows, although 4 bursting tags are trying to use all slots in each inventory round, there are still opportunities for a standard tag to be inventoried. As explained in §4.4, this is because there are gaps between bursts where passive tags can transmit. As expected, however, passive tags incur longer inventory time than they would otherwise, with the median time of roughly 1 second.

In our second experiment, we look at the inventory time as we increase the population of commercial passive tags. We consider two variants of Flit — one with the standard 50 ms sleep period between bursts, and one where this duration is increased to 100 ms. Figure 18 shows the average inventory time when there's a population solely comprising commercial passive tags vs a mix of passive tags and bursting CRFIDs. While the average inventory time increases with increasing population, the increase is greater when bursting CRFIDs are in the mix. However, the total time is still

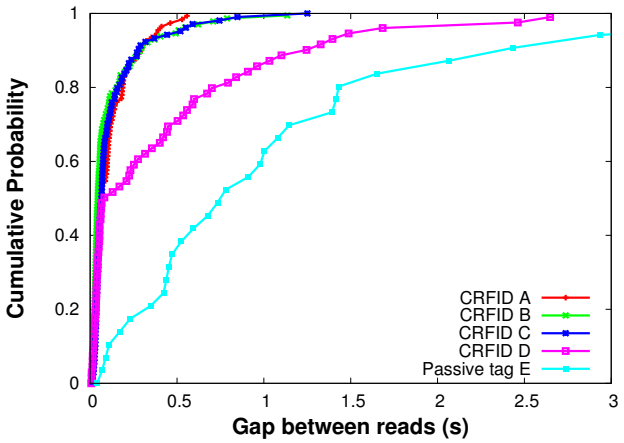

Figure 17. The interval for reader to read bursting tags and a passive Intel WISP.

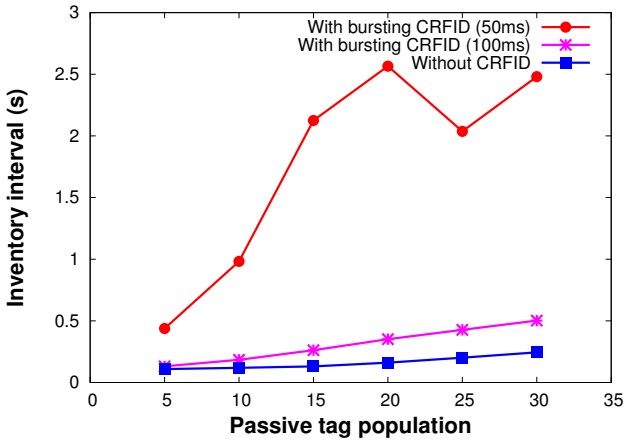

Figure 18. The average interval for a passive tag to be read increases with the tag population.

of the order of a few seconds, showing that passive tags still see opportunities to get data through. In addition, increasing the sleep duration between bursts to 100 ms dramatically reduces the inventory time to be only slightly larger than the case when there are only passive tags. This provides a simple knob in deployments where inventorying time for passive tags needs to be low.

7 Discussion and Future Work

Although our results show that Flit can provide significant benefits for bulk data transfer, there are several possible improvements and potential opportunities that we have not explored in this paper.

Duty-cycling the hardware subsystem: Our hybrid-powered CRFID (using the Intel WISP) currently powers the microcontroller from ambient energy + RF energy, whereas the comparator is powered solely by RF energy. The advantage of this approach is that it eliminates static power draw when the device is disconnected from a reader; the analog comparator is not turned on, and therefore does not have static draw during times when there is no reader present. The disadvantage of this approach is that this reduces range, since it only works for distances where there is enough RF energy to power the comparator. We believe that enabling the comparator to be powered by hybrid power can enable another significant bump in the range of a hybrid-powered CRFID.

However, powering the comparator using ambient energy will result in the static power draw being a concern when a sensor is not in the presence of a reader. In this case, it will be critical to duty-cycle the comparator even when a reader is not present, in contrast to our current design where we duty-cycle only when other nodes are bursting. Having a very short wakeup time is critical to duty-cycling efficiency. In our current work, we use a valid query message as the duration of wakeup; for duty-cycling in disconnected mode, we can use a different strategy. EPC Gen 2 uses a delimiter to identify the beginning of a message frame. A valid delimiter consists of a pulse of a pre-specfied length; if the pulse length is outside of a specific range around this lengh, it is considered invalid. After a valid delimiter, a tag can decode a message frame containing one of the commands we explained in §2. Our measurements show that looking for a *delimiter* from the reader provides a tighter and shorter wakeup interval, thereby improving efficiency.

Reliable transport: Our focus in this paper is on the design of a burst-mode alternative to the standard EPC Gen 2 protocol for high-throughput transfer. This layer is unreliable, however, in practice, we find that Flit loss rates are low even when several sensors are bursting to a reader at the same time. For example, in the experiment with five tags described in §6.3, we found the loss rates to be only around 10% (similar to the backscatter link loss rate shown in Fig 4).

While the loss rates will depend on the tag population, environment, distance, and other factors, we think that Flit provides an excellent building block for a reliable transfer protocol. A simple window-based reliable protocol over Flit would work as follows: After each burst of messages from the sensor to the reader, the reader sends a read command that has a bitmap of the messages that were received by the reader. In the next burst, the sensor can re-transmit these missing frame in addition to adding other data. We are currently implementing this protocol over Flit.

8 Related Work

Empirical Wireless Measurements: In recent years there has been significant work in measuring the characteristics of wireless communication channels for a variety of communication mechanisms (e.g. [17]). Perhaps most relevant to our work is [7], which quantifies wireless performance of several state-of-the-art reader and passive tag technologies through an empirical study that looks only at messages sent by a reader. In contrast, our focus is on improving tag to reader communications for CRFIDs. In addition, our measurements provide visibility into the backscatter link by embedding packet statistics in EPC codes and provides timing statistics by measuring message inter-arrival times using a mobile datalogger.

Bulk Data Transfer: The bulk transmission of data through a wireless channel has been studied in a variety of contexts including hardware and software systems. In an 802.11 setting [13] describes several mechanisms that together provide a transport layer optimized for bulk data transmission. Our work looks at RFID backscatter as opposed to 802.11; specifically, we focus on optimizations at the link layer for improving the throughput of bulk data transfer. Also of interest is [11], which provides a bulk transport protocol for 802.15.4 based wireless sensor networks. This work uses an end-to-end acknowledgement-based protocol to provide reliability, a rate control mechanism to minimize transfer time and a metric derived from combined signal strengths to avoiding hidden terminal issues. A vastly different approach is needed for CRFIDs that use EPC Gen 2 for communications, since tags can only hear a reader and not each other's transmissions. Therefore, we focus on detecting other CRFID bursts using reader messages rather than explicitly avoiding them with a collision avoidance metric; hidden terminals are a non-issue because tags cannot independently initiate data transfers.

Hardware approaches towards bulk data transmission at the link layer are used to improve throughput in in 802.11n [4] and to improve energy-efficiency in low-power radios [1]. Our state machine is as simple as the EPC Gen 2, so an ASIC version of a burst tag is certainly feasible. In this work, however, CRFIDs emulate EPC Gen 2 in software.

Energy Management: There has been some recent work on energy management or minimizing energy wastage in CRFID systems. The work presented in [14] instruments code at compile time with the ability to checkpoint software state to non-volatile storage; the goal is to avoid wasting the energy spent on work that was lost to power outages. In [6], a run-time system is presented that adaptively schedules a task to avoid energy wastage. The wastage is caused by work that does not complete because of energy limitations, as well as by energy lost to a saturated energy buffer. [9], looks at tradeoffs when ambient harvesting is used with RF harvesting, and explores the choice of hardware components to satisfy application requirements given an anticipated amount of harvested energy. Our work is complementary to all these efforts in that we improve the bandwidth and energy-efficiency of communication whereas other work largely focuses on energy management in the OS/run-time system.

Looking beyond CRFID research, there has been substantial work on energy management in harvesting-based sensors. For example, [16] achieves perpetual operation by scheduling tasks to match predicted energy harvesting rates, [10] and [18] looks at adaptive duty-cycling strategies for harvesting-based systems, [8] looks at using efficient solar harvesting in combination with an ultra-wideband impulse radio to balance energy usage at an even smaller scale, and commercial efforts have looked at micro-energy harvesting from miniature solar panels, thermal differences, vibrations, and wireless power-over-distance technology (e.g. Powercast [2]). Such techniques may be useful to ensure a suitable amount of buffered energy for bursts is available during reader contact periods, and is complementary to this paper.

9 Conclusion

In this paper we presented the design, implementation, and evaluation of Flit, a bulk transmission protocol for RFID-scale sensors. Through a careful analysis of the EPC Gen 2 protocol for passive RFIDs, we identified several opportunities for improvements to both goodput and energy efficiency when considering small numbers of CRFIDs that have large amounts of data to send. Through empirical evaluation, we demonstrated that significant gains in goodput are possible over a variety of distances when compared to a tag that implements vanilla EPC Gen 2. To enable the simultaneous bulk transfer of data from multiple CRFIDs, we design a simple coordination mechanism that works well in practice and through an experimental evaluation, show the complete system retains most of the performance improvements we observed for a single, uncoordinated CRFID. Finally, we argue that burst mode transmission should allow for coexistence between passive RFIDs and CRFIDs bursting data. Our results show that populations of passive RFIDs can still be read, but are inventoried with increased latency.

10 References

[1] Nordic nrf24ap2 user guide. `http://www.sparkfun.com/datasheets/Wireless/Nordic/ANT-UserGuide.pdf`.

[2] Powercast website. `http://www.powercastco.com`.

[3] TelosB Datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`.

[4] 802.11 specifications amendment 5: Enhancements for higher throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages c1 –502, 29 2009.

[5] M. Buettner, B. Greenstein, A. Sample, J. R. Smith, and D. Wetherall. Revisiting Smart Dust with RFID Sensor Networks. In *Proc. 7th ACM HotNets Workshop*, October 2008.

[6] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: An energy-aware runtime for computational rfid. In *NSDI*, 2011.

[7] M. Buettner and D. Wetherall. An empirical study of uhf rfid performance. In *in Proc. 14th ACM Int. Conf. on Mobile Computing and Networking (MobiCom*, pages 223–234.

[8] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman. Challenge: Ultra-low-power Energy-harvesting Active Networked Tags (EnHANTs). In *Proceedings of the 15th annual international conference on Mobile computing and networking (Mobicom)*, 2009.

[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 *MobiSys*, pages 195–208, 2010.

[10] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power Management in Energy Harvesting Sensor Networks. *ACM Transactions Embedded Computing Systems*, 6(4):32, 2007.

[11] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. E. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: a reliable bulk transport protocol for multihop wireless networks. In *SenSys*, pages 351–365, 2007.

[12] 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.

[13] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched networks: A new paradigm for wireless transport. In *NSDI*, pages 423–436, 2009.

[14] B. Ransford, J. Sorber, and K. Fu. Mementos: system support for long-running computation on rfid-scale devices. In *ASPLOS*, pages 159–170, 2011.

[15] J. R. Smith, A. P. Sample, P. S. Powledge, S. Roy, and A. Mamishev. A Wirelessly-Powered Platform for Sensing and Computation. In *Proc. 8th International Conference on Ubiquitous Computing (UbiComp)*, 2006.

[16] 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 *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2007.

[17] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low-power wireless. *TOSN*, 6(2), 2010.

[18] C. M. Vigorito, D. Ganesan, and A. G. Barto. Adaptive Control of Duty Cycling in Energy-harvesting Wireless Sensor Networks. In *The IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 21–30, 2007.

[19] R. Want. RFID Explained. In *Synthesis Lectures on Mobile and Pervasive Computing*. Morgan & Claypool Publishers, 2006.