A "Gen 2" RFID Monitor Based on the USRP

Michael Buettner*
*University of Washington
buettner@cs.washington.edu

David Wetherall†*
†Intel Labs Seattle
djw@cs.washington.edu

ABSTRACT

We have developed a low cost software radio based platform for monitoring EPC Gen 2 RFID traffic. The Gen 2 standard allows for a range of PHY layer configurations and does not specify exactly how to compose protocol messages to inventory tags. This has made it difficult to know how well the standard works, and how it is implemented in practice. Our platform provides much needed visibility into Gen 2 systems by capturing reader transmissions using the USRP2 and decoding them in real-time using software we have developed and released to the public. In essence, our platform delivers much of the functionality of expensive ($>$ \$50,000) conformance testing products, with greater extensibility at a small fraction of the cost. In this paper, we present the design and implementation of the platform and evaluate its effectiveness, showing that it has better than 99% accuracy up to 3 meters. We then use the platform to study a commercial RFID reader, showing how the Gen 2 standard is realized, and indicate avenues for research at both the PHY and MAC layers.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Experimentation, Measurement

Keywords

RFID, EPC Gen 2, Software Defined Radio

1. INTRODUCTION

Radio Frequency IDentification (RFID) is an emerging wireless technology that allows small, inexpensive computer chips to be remotely powered and interrogated for identifiers and other information. While there are many kinds of RFID, e.g., HF RFID in credit cards, recent advances in RFID have focused on passive UHF RFID as standardized by the EPC Class-1 Generation-2 (Gen 2) specification in 2004 [4].

Gen 2 RFID was originally developed as a replacement for barcode identification systems. It provides key advantages such as read ranges of more than 10 meters, non-line-of-sight operation, high inventory rates, and rewritable product IDs. It has seen widespread deployment for tracking pallets in the supply chain and is rapidly expanding to new applications such as large scale, item-level tracking of consumer goods [14, 13]. In 2008, the US Passport Card and the New York and Washington state Enhanced Drivers Licenses were introduced. They contain embedded Gen 2 RFID tags that can be read from up to 50 meters away [11]. The cards were ostensibly introduced to reduce delays at border crossings. New uses that stretch the initial vision of RFID are emerging, as the capabilities of RFID devices advance to include computation, storage and sensing [16].

Given this rapidly growing application space, understanding how Gen 2 RFID operates in practice is of significant interest to improve current deployment practices, and to identify research challenges in RFID protocols and systems. The Gen 2 standard allows for great flexibility in terms of both physical layer configuration and the sequence of reader commands that can be used to inventory tags. It is important to understand the impact and trade-offs of these implementation decisions. Unfortunately, gaining visibility into RFID deployments is difficult because existing RFID readers are black box systems. They allow limited configuration and return high-level results that simply list tags in range of the reader. They do not expose low-level behavior such as PHY layer configuration, MAC protocol realization, or error rates and timing information. This makes diagnosing problems difficult, and gives researchers little insight into the challenges faced by real-world RFID deployments.

Because the fine-grained behavior of current systems is poorly understood, existing analytical [9, 12] and simulation studies [6] of RFID are based on idealized versions of both the Gen 2 standard and RFID device performance. Though tools for detailed evaluation of RFID readers and tags exist, they are expensive ($>$\$50K) as they are built using high-end equipment such as vector signal analyzers and proprietary software [8, 1]. Additionally, they are targeted at conformance testing in a lab setting and are ill-suited to analyzing real-world deployments in situ. As a result, existing studies of deployed systems have generally been coarse grained and derive performance metrics from simple read rate [2, 7]. We aim to provide a tool whereby researchers can conduct low-level studies of deployed systems at a reasonable cost.

In this paper, we present a Gen 2 RFID monitoring platform that provides deep visibility into the operation of RFID systems. To achieve this, we use the Universal Software Radio Peripheral 2 (USRP2) to capture RFID signals and software developed using GNU Radio to decode reader commands. Commercial solutions are built upon signal analyzers capable of GS/s sampling rates. However, the 25 MS/s sampling rate of the USRP2 is sufficient for capturing the complete 25 MHz RFID band with high fidelity. This

makes our solution cheap and portable as it consists of only a USRP2 and a laptop. By using the open source GNU Radio toolkit to process signals on the host, the software is free and can be easily modified by researchers.

Our monitoring system decodes all RFID reader transmissions in the RFID band and outputs a message level trace of reader behavior. These traces show the design decisions made by vendors when realizing the Gen 2 protocol, and illuminate real world dynamics such as error rates and protocol timing; such factors are critical for realistic analytical and simulation studies. We expect that our platform will be helpful for identifying bottlenecks in RFID systems and thus avenues for further research; our previous, less capable version has already proven useful in this capacity [3].

We make three main contributions in this work. First, we present the design and implementation of our system, showing how a low-cost SDR platform and open-source software can be used to create a powerful Gen 2 RFID monitoring tool. We show that our tool can decode reader commands across the complete RFID band with better than 99% accuracy at 3 meters. Second, we use our monitor to study a commercial RFID reader and, to the best of our knowledge, present the first description of how the Gen 2 protocol is realized in practice. As part of this study, we find that algorithms for PHY layer rate adaptation and cross-layer techniques that factor the capture effect into MAC behavior are rich areas for further research. Lastly, this paper can be viewed as a companion document to our monitor system software which can be downloaded from the Comprehensive GNU Radio Archive.[1] We hope that by demonstrating how our monitor can be used to explore existing systems, researchers will use our platform as a foundation for the development of new tools for research.

2. GEN 2 RFID BACKGROUND

In this section we introduce the Gen 2 PHY and MAC layers to motivate the design of our monitoring system. Gen 2 tags are fully passive, which means they harvest the entirety of the operating energy from nearby RFID readers. The specification was designed with two major constraints in mind. First, it must be implementable on very low cost RFID tags (each costing a few cents) that are wirelessly powered and computationally weak. Second, it must operate in a regime where readers can communicate with tags and vice versa, but tags cannot communicate with each other or even hear other tag transmissions.

2.1 Gen 2 Physical Layer

The Gen 2 physical layer has the dual purpose of maximizing harvestable power at the tags, and facilitating downlink and uplink communication. While a reader is communicating with tags it must transmit a continuous RF wave (CW); tags power themselves by harvesting the energy in this RF signal. Down-link communication uses On/Off Keying where bit boundaries are indicated by brief pulses of zero amplitude in the CW, and Pulse Interval Encoding (PIE) where the time between zero amplitude pulses differentiates a zero or a one. Depending on PIE durations, downlink rates range from 27 kbps to 128 kbps.

Uplink data rates are between 5 kbps and 640 kbps, and communication is achieved via "backscatter" transmission.

[1]https://www.cgran.org/wiki/Gen2

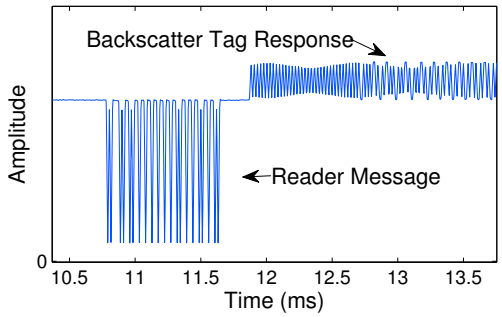

Figure 1: Reader message and tag response

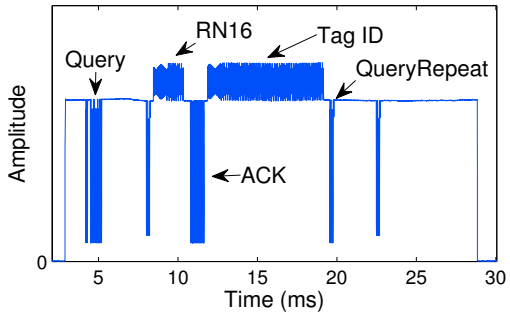

Figure 2: Message exchange for a Query round

Passive RFID tags do not technically transmit any energy. Instead, they manipulate how well they reflect (backscatter) the incident CW. This reflected signal, though weak, is received by the reader which decodes the modulated data. Uplink modulation is determined by two parameters specified by the reader; uplink frequency and data encoding. Gen 2 specifies four encodings schemes that differ in the number of cycles per symbol (varying from one cycle to eight). For a given link frequency, an encoding using more cycles will have a lower data rate, but will be more robust to noise.

Figure 1 shows communication between a reader and tag as captured by a USRP. The CW results in the DC offset of the received waveform, with the series of low amplitude pulses being a reader transmission. The backscattered tag response can be clearly seen as a combination of the incident CW and the reflected CW from the tag. It should be noted that the signal seen in the figure was captured with the USRP antenna placed inches from the RFID tag. This accounts for the prevalent backscatter signal.

To limit interference with other devices in the 902–928 MHz ISM band, FCC regulations stipulate that UHF RFID systems frequency hop across fifty 500 kHz channels, with dwell times no longer than 400 ms. However, because tags do not "tune" to different channels, when multiple readers are active in an area their transmissions will collide at tags even if the readers are on different channels.

2.2 Gen 2 MAC Layer

Gen 2 tags decode reader transmissions using a simple edge detector that detects the conspicuous zero amplitude pulses in the CW. However, tags are unable to decode, or even detect, the backscattered signals of other tags. This is in contrast to Ethernet or 802.11 where nodes can hear each other, at least when they are not transmitting. Conse-

quently, the Gen 2 MAC protocol is based on Framed Slotted Aloha [15] which was designed to operate in a context where transmitting nodes cannot hear each other.

The general model of RFID is that readers are continuously "inventorying", i.e., looking for tags that are in range. An inventory round begins with the reader transmitting a *Query* command that indicates the number of slots in the frame. The number of slots in a frame is a power of two in the range [1, 32768]. After receiving a *Query*, tags randomly choose a slot in which to reply, and transmit a 16-bit random number (*RN16*) in that slot. If the reader receives an *RN16* in a slot, it *ACKs* the *RN16* and the tag that transmitted it replies with its 96-bit identifier. Tags that collide in a slot are not *ACKed* and respond again after the next *Query*.

Figure 2 shows a message exchange with only one tag present. In the example, the reader powers up and transmits a *Query* message that specifies four slots in the frame. The first slot immediately follows the *Query* command, and is empty in this example. The second reader command is a *QueryRepeat* which indicates the beginning of a new slot. In this case, the tag had chosen the second slot and so it transmits its *RN16* in response to the *QueryRepeat*. It is then *ACKed* by the reader, and transmits its *ID*. Because the reader specified four slots in the *Query*, it sends two additional *QueryRepeats* looking for any remaining tags. In this case, there is only a single tag so the last two slots are empty. The reader then powers down. This example illustrates a very simple Gen 2 message exchange and does not describe every message defined by the protocol. We will examine more complex scenarios when we look at how the complete MAC is implemented in a commercial reader.

3. SYSTEM DESIGN AND EVALUATION

Our monitoring platform aims to provide much of the functionality of commercially available RFID analysis tools, but with greater flexibility and at a fraction of the price. One key insight is that commercial RFID analytics solutions are generally add-on modules to high-end signal analyzers capable of many gigasamples per second. This is far more than is necessary to capture the 900 MHz ISM band.

By using lower end hardware and open source software, we give researchers a low cost tool for monitoring RFID traffic that can be completely modified by the user. While our monitoring system does not currently provide all the functionality of commercial systems, the hardware platform can theoretically support a complete suite of analysis tools. This includes decoding tag transmissions and messages from multiple, simultaneously transmitting readers. This would be useful for studying existing large scale deployments [19]. By releasing our software to the community, we hope that researchers will continue to build upon our work.

The hardware platform we use is the Universal Software Radio Peripheral 2 (USRP2) developed by Ettus Research. Signal processing is performed on the host using software developed using the GNU Radio framework. The overall system architecture is shown in Figure 3. Our monitoring system is placed close to an RFID reader, and the USRP2 captures the reader transmissions. The digitized signal is streamed to the host PC over GigE and processed using both stock and custom GNU Radio signal processing blocks. Our custom blocks are shaded in the diagram. The signal undergoes initial signal processing steps, reader transmissions are decoded by the *Edge Detector* and *Bit Decoder* blocks, and

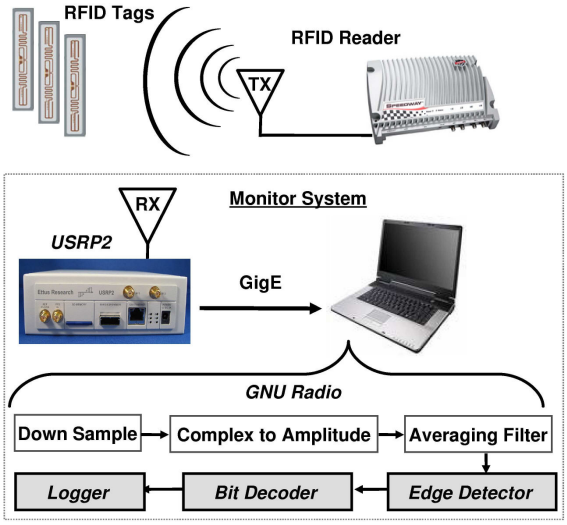

Figure 3: Monitor system diagram

reader traffic is logged. The following subsections detail the salient features of these components.

3.1 Capturing RFID Signals

The USRP2 is a more powerful version of the original USRP, and has a complex sampling rate of up to 25 MS/s. This is just sufficient for capturing the entire UHF RFID band. The USRP2 is combined with an RF front-end that downconverts signals from the 900 MHz ISM band to baseband, and an onboard FPGA samples this baseband signal and transmits it over GigE to the host PC.

Though the USRP2 can sustain 25 MS/s over the GigE interface, GNU Radio cannot process a 25 MS/s signal using currently available host machines without falling behind and dropping samples. For example, we were unable to simply write the signal to disk for offline processing, or convert the stream from complex I/Q data to amplitude data, without losing large swaths of data.

Without modifying the FPGA code of the USRP2, the sample rate over the GigE interface can only be reduced via *decimation*. Decimation consists of an initial *low-pass filter* followed by *downsampling* to reduce the sample rate. The low-pass filter is generally needed because simply downsampling a signal results in *aliasing*, where non-existent low frequency components are introduced into the signal because the Nyquist rate of the high frequency components is violated. However, when decimating at the USRP, the low-pass filter reduces the width of the frequency band and the host does not receive the complete RFID band.

Because Gen 2 uses On/Off Keying, frequency aliasing does not effect demodulation. Hence, we reduce the sample rate while capturing the entire 25 MHz band by downsampling by a factor M at the host as the first step in the processing graph. This allows us to tune the USRP2 to the center of the ISM band, and still capture traffic at the edges.

In our implementation, M is tunable so users can reduce the sample rate until their host can keep up with the signal stream. The sample rate must remain high enough to accurately represent the signal, with higher sample rates being more robust to noise. We find that a sample rate of 3 MS/s, which can easily be supported by most host computers, is sufficient to capture the highest rate reader transmissions.

Time (s)	Command	Bits	Length (us)	Inter-Arrival
20.294363	P-DWN	NA	NA	592
20.299102	QUERY	0x8D4064	955	3556
20.300731	QUERY	0x8D4250	955	748
20.301414	QREP	0x0	222	540
		...		
20.309600	QREP	0x0	222	535
20.311117	ACK	0x7EF84	872	746
20.314011	QREP	0x0	224	2614

Table 1: Reader message trace

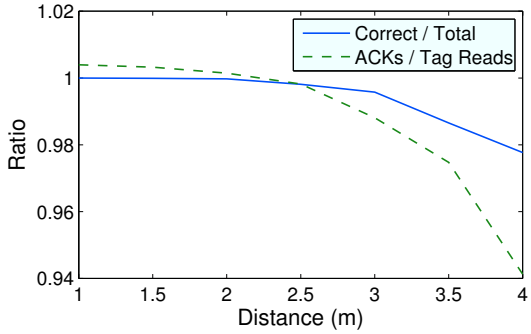

Figure 4: Monitor accuracy

3.2 Decoding Reader Messages

Decoding reader transmissions is relatively straightforward. In essence, our monitor needs to solve the same decoding challenge as RFID tags, using hardware costing a few thousand dollars instead of a few cents. The signal is conditioned for decoding by converting the complex I/Q samples to amplitude samples, and then passing the signal through an averaging filter to smooth out noise. An example of the resulting signal is shown in Figure 1.

Decoding consists of two parts. First, the low amplitude pulses of reader transmissions are detected by finding negatives edges in the signal. This is done by comparing each incoming sample against a running average of the signal amplitude. Negative edges due to noise are ignored by only triggering on edges that drop below 90% of the average amplitude; we found this threshold to work well in practice.

The second part is determining the bits in the signal by inspecting the duration between low amplitude pulses. Each reader message is preceded by a preamble which encodes, among other things, the duration of a "0" and a "1", with a "1" being approximately twice as long as a "0". [2] The monitor extracts this timing information from the preamble, and uses it to distinguish the subsequent bits in the message body. End of message boundaries are detected when no negative edge is seen for a duration greater than twice a "1" bit period. The decoded bits are then logged.

3.3 Logging Message Traces

Decoded reader messages are timestamped and written to a trace file on the host. All bits and preamble timing data are logged. This allows us to determine details such as the PHY layer parameters, the number of slots specified by the *Query* command, and the RN16 sent in the *ACK* command. If the decoded bits do not match a valid Gen 2 command, a decoding error is noted in the trace.

The duration of each message is noted, beginning with the first negative edge and ending with the last, as is the interarrival time. These values are calculated based on the number of samples between these points and the time per sample. This results in a complete trace of all reader traffic in the RFID band, with microsecond resolution.

Table 1 shows a section of a trace file. In the interest of space, PHY layer parameters gleaned from message preambles have been elided. Both *Length* and *Inter-Arrival* are shown in microseconds. Inter-arrival timing is useful for filling in blanks in the trace. For example, we can determine the approximate duration of the *ID* backscattered by the tag by inspecting the *Inter-Arrival* value for the last *QueryRepeat*(QREP) as this measures the time from the last edge of the reader *ACK* to the first edge of the *QueryRepeat*. This gives us a good estimate of the duration of tag transmissions, as the periods between reader commands and tag responses (and between tag responses and reader commands) is constrained by the Gen 2 standard to within a few microseconds.

[2]For more details, we direct interested readers to the Gen 2 standard [4].

3.4 Evaluating Monitor Accuracy

In this section we evaluate how accurately our monitor decodes reader messages. We require that it perform well out to at least a few meters, because it may not always be feasible to place our monitor directly beside the reader.

As we do not have "ground truth" data showing what messages the reader actually transmitted, we use two heuristics to evaluate our monitor. First, we look at how often the monitor self-reports errors in decoded messages. Reader messages begin with a command code that identifies the Gen 2 command, and each command has a unique length. When a reader transmission is detected by our monitor, the subsequent message must match both of these features for a known Gen 2 messages, otherwise an error is logged. For the second heuristic, we compare the number of *ACKs* detected by the monitor with the number of tag reads reported by the reader, as there must be at least one *ACK* transmitted per tag read reported.

For our experiment, we deployed one Impinj Speedway RFID reader and our monitor, along with one Alien "Omni-Squiggle" RFID tag. Both the monitor antenna and the RFID tag were placed approximately one meter from the transmitting antenna of the reader. The reader inventoried the tag continuously for one minute while the monitor gathered trace data. The reader initially transmitted at 30 dBm, the maximum allowed by the FCC. After each one minute run the transmit power of the reader was reduced. This approach keeps the multipath environment stable between experiments. We present results in terms of equivalent distance using free-space propagation, as we find this more intuitive than results in terms of transmit power.

Figure 4 shows the results when the reader was configured with a 27 kbps downlink. We experimented with other configurations, and found similar results. The data consists of over 50,000 logged events at each distance. The solid line shows the ratio of error-free messages to the total number of messages detected, with the y-axis spanning approximately 94%–100%. Out to three meters, more than 99% of detected messages were decoded without errors.

The figure also shows the ratio of *ACKs* decoded compared to the number of tag reads reported by the RFID reader. By this metric also, our monitor is approximately 99% accurate at three meters. One thing to notice is that this ratio can be slightly greater than 1. This is because when a tag ID is received with bit errors, the reader must retransmit the *ACK*. However, the read rate was constant from 1 to 4 meters and we found that bit-errors did not appreciably increase; our monitor just becomes less able to decode the *ACKs*. Overall, our monitor has an accuracy well above 90% to approximately 4 meters, which is sufficient for the intended deployment scenarios.

4. EXPLORING THE STATE OF THE ART

In this section, we use our monitor to closely examine the behavior of a commercial reader at the message level. It has been difficult to know how Gen 2 is implemented in commercial systems as many implementation details are left to the vendor. For example, choices at the physical layer can result in downlink rates that vary by more than a factor of four, and uplink rates can vary by more than two orders of magnitude. At the MAC layer, the precise sequence of commands used to inventory tags is not specified by the standard, and it is unclear how real systems choose frame sizes to efficiently read a given tag population. These implementation choices made by vendors are key to reader performance, and the design trade-offs of these choices are poorly understood.

4.1 Experimental Setup

The data presented in this section was gathered using an Impinj Speedway reader, our monitoring system, and up to 64 Alien "Omni-Squiggle" RFID tags depending on the experiment. The monitoring system consisted of a USRP2 and a Lenovo T61 laptop running 64-bit Linux. The antenna for our monitor was placed inches from the reader antenna to generate the most accurate trace possible. The traces were seen to have an error rate no greater than 1%. This error rate was calculated as described in the previous section. Unless otherwise specified, tags were placed on a sheet of poster board located one meter from the reader. The transmit power of the reader was varied and equivalent distance calculated based on free-space propagation.

4.2 Physical Layer Rate Adaptation

The Gen 2 specification allows for a wide range of PHY layer configurations. Manufacturers are free to choose downlink rates from 27 kbps to 128 kbps and uplink rates from 5 kbps to 640 kbps. There are four uplink encodings (FM0, and Miller-2/4/8) that vary in how many cycles they use per symbol, with more cycles reducing the data rate but being more robust to noise. By carefully selecting these parameters, manufacturers can tune their systems to balance data rate against robustness and spectrum usage.

RFID readers do not generally allow users to set PHY parameters independently. Instead, they provide a small set of preconfigured options which are statically configured at deployment. The Impinj reader provides 5 "modes", with the highest rate option using a 112 kbps downlink and a 640 kbps FM0 encoded uplink, and the lowest rate using a 27 kbps downlink and a 32 kbps Miller-8 encoded uplink.

In our previous work [3], we showed how different PHY configurations impact MAC behavior and overall reader performance. We also proposed a technique where readers dynamically adapt their PHY layer configuration to increase performance. The Impinj reader provides something very similar with their "AutoSet" mode, and using our monitor we can determine how their approach is implemented.

To determine how "AutoSet" mode adapts to varying conditions, we had the reader continuously read one tag as equivalent distance increased from one to six meters, and reader messages were logged using our monitor. According to the Gen 2 protocol, if bit errors are detected in the tag ID, the reader must send a *NAK* message to tell the tag to retransmit. Consequently, *NAK* messages are a good proxy for uplink error-rate, though bit errors in the *RN16* and messages below the noise floor are not detected. We found that the incidence of *ACKs* immediately followed by a *NAK* increased with distance from less than 1% at one meter to more than 75% at six meters. Thus, the reader would be expected to adapt to the changing link quality.

Examining the details of the *Query* messages, we found that only three distinct PHY configurations were used. All three used a 35 kHz downlink with the uplinks being 320 kbps/FM0, 68 kbps/Miller-4, and 20 kbps/Miller-8. When no tags are present, the 68 kbps uplink is used. When a tag is placed close to the reader, the reader generally operates using the 68 kbps configuration, but just before the reader powers down it transmits one last *Query* using the 20 kbps configuration; ostensibly to "pick up" any remaining tags with weak uplinks.

One surprising find was that the 320 kbps uplink is only used when the tag is far from the reader and the error rate is very high. In this case, the reader begins most rounds using the 320 kbps configuration, but continues to end each round with the more robust 20 kbps rate. We do not entirely understand this behavior, particularly because at 6 meters 97% of the IDs sent at 320 kbps had errors, whereas only 19% had errors when sent at 20 kbps. Even considering the faster uplink, the protocol overhead associated with errors would seem to make this strategy far from optimal.

We expect that we did not capture all the behavior of the "AutoSet" mode with our simple experiment. However, our findings suggest that there is potential for the further development of dynamic PHY layer algorithms. Our monitor can be used to build realistic error models for readers in their deployed scenarios, and these can be used to inform analytical and simulation studies of such algorithms.

4.3 MAC Layer Frame Size Selection

One key factor in reader performance is accurate frame size selection. The number of slots in the frame is indicated in the *Query* message, and this controls the probability of tag collisions. It has been shown that the optimal frame size is when the number of slots is equal to the number of tags [17], but estimating the number of tags in the population is non-trivial.

While the Gen 2 specification gives an example algorithm for selecting the frame size, there has been significant work targeted at improving this estimation [5, 10]. We set out to determine how the Impinj reader chooses frame sizes to match the tag population. For this, we had the reader inventory a set of tags placed 2 meters away that varied from 1 to 64 tags.

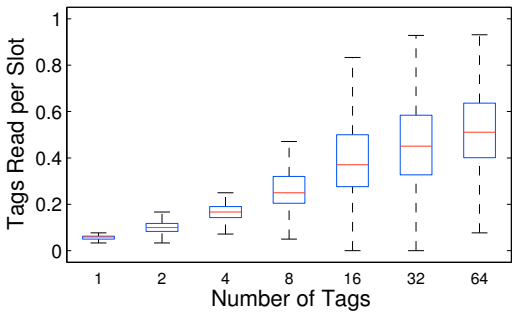

Figure 5: Distribution of tag reads per slot

Immediately after powering up, the reader transmits a *Query* that specifies only one slot. If no tags are present, the reader power down for 2 milliseconds and tries again. The first *Query* is effectively just a check to see if any tags are present. If at least one tag is present, the next *Query* always specifies 16 slots. Beginning frames with no less than 16 slots was suggested in [18], though this is shown to be less than ideal when more than 27 tags are present.

We also found that the reader does not generally iterate through all slots in the frame. If, for example, no tags reply in the first five of the sixteen slots, the reader aborts the frame, as the empty slots indicate that the frame size is too large. Estimating population size on a per slot basis, and aborting frames when necessary, has been shown to increase Gen 2 performance [5].

Because tags randomly choose a slot in which to respond, some slots will be empty, some slots will have only a single tag response, and some slots will have multiple, colliding, tag responses. When the number of slots in a frame matches the number of transmitters (the ideal case), the ratio of slots with a single transmitter approaches e^{-1} (≈ 0.37) [17]. Consequently, if the reader accurately selects the best frame size, we would expect to see an average of 0.37 tags read per slot.

Figure 5 shows the distribution of the average number of tags read per slot for different sized tag populations. We calculated this by comparing the total number of slots in each round to the number of slots where an *ACK* was transmitted. We found that, in practice, the efficiency of the protocol is much higher than e^{-1}. When reading 64 tags, the median number of tags read per slot is > 0.5. This is due to the capture effect, where a strong signal is decoded if it collides with a weaker signal.

Our results shows that capture can be a significant component in MAC performance and should be taken into account. However, the extent to which capture happens, and the extent to which this affects protocol performance, has not been studied in depth. As a result, most frame size selection algorithms seen in the literature do not explicitly model the impact of the capture effect. Additionally, the uplink physical layer configuration will impact the prevalence of the capture effect, and consequently the efficiency of the MAC protocol. To the best of our knowledge, this type of cross layer relationship has not been explored.

5. SUMMARY

In this paper, we present a Gen 2 RFID monitoring platform that provides deep visibility into the operation of RFID systems. Our platform uses the USRP2 to capture the complete Gen 2 RFID spectrum, and reader transmissions are decoded using software we developed using the GNU Radio toolkit. By using a low-cost SDR platform and open-source software, we have built a system that is flexible and low cost, but quite powerful for exploring Gen 2 systems. We evaluated the performance of our monitor and found that it was 99% accurate when placed within 3 meters of the reader. This is sufficient for our target scenario of monitoring deployed Gen 2 systems in situ.

To demonstrate the usefulness of our monitor, we studied a commercial RFID reader. We presented details showing how the Gen 2 protocol is realized in practice, and also indicated avenues for further research. Specifically, we feel that algorithms for PHY layer rate adaptation and cross-layer techniques that factor the capture effect into MAC behavior are rich areas for further exploration. Our system software has released as an open-source project, and is intended as a foundation to be used by other researchers.

6. REFERENCES

[1] Agilent Technologies. 89600 VSA RFID modulation analysis. www.agilent.com/find/89600.
[2] S. Aroor and D. Deavours. Evaluation of the state of passive UHF RFID: An experimental approach. In *IEEE Systems*, 2007.
[3] M. Buettner and D. Wetherall. An empirical study of UHF RFID performance. In *MobiCom*, 2008.
[4] EPCglobal. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz-960 MHz version 1.0.9. 2005.
[5] C. Floerkemeier. Bayesian transmission strategy for framed ALOHA based RFID protocols. In *IEEE RFID*, 2007.
[6] C. Floerkemeier and R. Pappu. Evaluation of RFIDSim - a physical and logical layer RFID simulation engine. In *IEEE RFID*, 2008.
[7] S. Hodges et al. Assessing and optimizing the range of UHF RFID to enable real-world pervasive computing applications. In *Pervasive Computing*. Springer-Verlag, 2007.
[8] Intermec Technology Inc. Using national instruments software and hardware to develop and test RFID tags. http://sine.ni.com/cs/app/doc/p/id/cs-10511.
[9] Y. Kawakita and J. Mitsugi. Anti-collision performance of gen2 air protocol in random error communication link. In *SAINTW*, 2006.
[10] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in rfid systems. In *MobiCom*, 2006.
[11] K. Koscher, A. Juels, V. Brajkovic, and T. Kohno. EPC RFID tag security weaknesses and defenses: passport cards, enhanced drivers licenses, and beyond. In *CCS*, 2009.
[12] P. Nikitin and V. Rao. Performance limitations of passive UHF RFID systems. In *IEEE Antennas and Propagation Symposium*, 2006.
[13] RFID News. Portuguese book retailer rolls out item-level rfid deployment. http://www.rfidnews.org.
[14] RFID Update. Why metro's item-level rfid deployment matters. http://www.rfidupdate.com.
[15] L. G. Roberts. Aloha packet system with and without slots and capture. *SIGCOMM Comput. Commun. Rev.*, 1975.
[16] A. P. Sample et al. Design of an rfid-based battery-free programmable sensing platform. In *IEEE Transactions on Instrumentation and Measurement*, 2008.
[17] F. Schoute. Dynamic frame length aloha. *IEEE Transaction on Communications*, 1983.
[18] H. Vogt. Efficient object identification with passive RFID tags. *Pervasive Computing*, 2002.
[19] E. Welbourne et al. Longitudinal study of a building-scale RFID ecosystem. In *MobiSys '09*, 2009.