

A Conjunctive Middleware Adaptation of Active and Passive RFID Technologies

A Thesis
Presented to
The Academic Faculty

by

Brent Rowswell

In Partial Fulfillment
of the Requirements for the Degree
Bachelors of Science in the
School of College of Computing

Georgia Institute of Technology
May 2010

Abstract

RFID technology has been split into two branches. Active RFID and passive RFID behave in very different manners, yet the theory behind them is practically the same. If the two branches were able to be merged into use under a single software system, then a number of use cases can be made that mitigate each technology's weakness with the other. The only major obstacle to using them interchangeably is the lack of a standard messaging system. Since we can't create a standard for industry, we plan on using a middleware to translate to our own standard.

Introduction

Radio Frequency Identification (RFID) has been in use since the early seventies, and yet we still have many problems using it. RFID technology has as many applications as can be thought of, where the hardware handles the job of tracking specified RFID tags, giving a data packet back if the tag is seen and none otherwise. Currently there have been several uses that have spawned significant interest in the technology, particularly for shipping and keeping inventory. In warehouses now there are many problems keeping inventory lists and counts straight, a problem which costs quite a bit of money. The current solution is to tag an item and track movement, which would theoretically end any sort of shipping mishaps, making cargo manifests simple to produce and ordering

items even easier to handle. Most of the RFID systems proposed use passive RFID tags, which are a very inexpensive type of tag that has an extremely long lifetime.

Unfortunately passive tags are also notoriously unreliable, causing severe amounts of delay due to the need to validate their messages. Furthermore their effective range is quite small, making many of these systems extremely large, and keeping tags very close together.

There is an alternative form of tags, active RFID tags. These also have their own problems, since they are more expensive, larger, and have shorter lifetimes than their counterparts. The benefit from using active tags is that they are significantly more reliable, have much larger reading ranges, and can have limited communication on the fly to change some of their settings. Both of these types of tags have significant drawbacks, making one or the other difficult to implement in many situations.

As of yet there is no way to merge the use of both types of tags, as most RFID tags address messages completely differently. Even if these RFID tags were to be deployed in an environment, current solutions have difficulties scaling up from the local level to the global scale. Our software solution tries to handle all of these problems in a robust and extensible way without becoming too time-intensive.

There are very well known physical issues regarding the characteristics of RFID technology. For instance, it is well known that passive readers have problems gaining reliability in being read due to various field nulls, such as multipath fading or signal degradation due to the surroundings.[2] However, what is not as well known is how these RFID signals work in a mixed environment of both active and passive tags.

As discussed before there exist several RFID manufacturers, each of which uses different proprietary communications. There has been some research into multi readers, or a component that allows a system to understand messaging from multiple manufacturers. Furthermore there is promising data that systems can detect the difference between brands of readers, and can identify what type of reader is being used.[6] We have many known similarities between the functionality of active and passive readers, and the technology is physically similar[7] enough that we should be able to compile a list of methods that readers should be able to do. And yet when we have all of these similarities between the two technologies, this is the first study of a mixed environment of active and passive readers using multireader technology.

Background information on RFID middlewares

Middlewares are software systems that tie two software applications together. In this particular instance our middleware ties into the physical RFID readers and the given applications. Ideally an RFID reader will come with its own software package that lets a middleware keep its distance from the hardware. This lets us create an environment in which we can filter the proprietary commands of the RFID readers into a uniform command from the middleware to an application. A uniform messaging system is helpful for the extensibility of the project by letting one piece of middleware tie into many different applications.

The various aspects of RFID middleware have already been divided by many middleware developers into several categories. ISO/IEC 9126 factors help us determine

if a middleware will be functionally viable in a development environment[1]. Our study uses the categories of efficiency, portability, usability, reliability, and functionality to gauge if multireader approach to incorporating both active and passive RFID technologies will become an effectively viable approach to middleware solutions. Most RFID middlewares focus on a single type of RFID tag, generally passive tags.[2, 3] It has become common practice to use stochastic methods to overcome the passive tags' inherent unreliability.[2] This practice allows even unreliable signals to become statistically reliable to a degree, depending on the amount of time spent reading information from tags. However even with stochastic methods to generate reliable data from these tags, they still incur a large time component to correctly sample data, making them have a large room for improvement. One of the interesting questions in the field is if adding in the more reliable active RFID tags will help alleviate the time constraint placed on the passive tags.

The OSGi framework has been used to create a reconfigurable RFID middleware that can incorporate a variety of different embedded devices. It maintains its uses throughout various embedded devices by separating the interface and the implementation and by creating a standard that separates embedded components.[4, 5] The RF2ID framework establishes a connection between the spatial placement of readers and their respective data load. This setup allows users to monitor the discrepancies in loads at certain spatial or temporal points, allowing for easy load balancing, which is a known issue in the primary warehouse use case.[3] Due to the use cases being looked at and their noticeably temporal and spatial components as well as the necessity of load balancing issues, we have chosen to use the RF2ID framework for testing.

Our Project

Current middleware solutions have problems scaling up, making a ubiquitous solution difficult to manufacture. Currently there are several different types of RFID tags and readers, and just as many different forms of messaging going out from the readers to any sort of middleware. This poses a problem to the community of developers since they then create applications that are RFID specific, since the messaging and available options change from model to model. Our solution is to create a middleware that ties in between several RFID readers to create a common message that can then be reinterpreted in the use case. Creation of this middleware involved three steps.

Project Overview

1. The first step involved figuring out which reader is actually plugged in. This determines what format the messages need to be sent to, at what rate, and what to expect back. This is particularly hard and vital as these messages need to be correct in order that the reader will function. In this step we determined what reader we are dealing with.
2. The second step focuses on the existing RFID readers and tags, specifically to catalog what sort of messages are being used, what a user would have access to normally, and what to expect from the reader as a response. Examples of this category include

acknowledge messages, group identification, and commands to receive packages. In this step we determined how to communicate with the reader.

3. The final step handles the outgoing messages. That involves figuring out what applications want, what can be generated from the information coming into the middleware from the readers, and what solutions can be made along the way to help out developers for various use cases. Commonly found uses for this include path finding, data mining, and load balancing. In this step we determined how to let a user communicate with the reader, and facilitated further development.

Step 1: Finding which reader

Each reader has its own unique set of characteristics dealing with operation. Baud rate, power usage, interface, and opcodes all differ based on models. Most readers allow for relatively easy identification via communication after the reader has been initialized. Unfortunately, if the reader isn't known then neither is the format, and thus communication with the reader becomes prohibitively difficult.

Therefore this step handles the initialization of a generic reader into a state where the identity of the reader can be determined. With our test case we use the message format from the more temperamental reader first, and if an error message appears, switch to that of the second reader based on the type of error message. This allows us to determine which of the two readers is being used with less likelihood of the reader needing to be reset.

Step 2: Cataloging functionality

During this step we cataloged various functionalities between active and passive readers. Specifically we found the methods that are common between them, such as sampling data, reading data, reading tag identification, and sampling baud rates. Even if we know what reader we are dealing with from step 1, we still need to know what functionality comes with most readers, and what to expect.

We chose to look at two readers, one active and one passive reader, with common functionality due to time constraints, so the results may include more methods than are actually found between all readers. The available functionality of readers has been relatively straightforward to come across from the manufacturers, although the discrepancies between implementation styles are noticeable on most if not all levels of the platforms architecture. It is because of these differences in messaging and internal architectures that we want to categorize the functionalities so that we can merge all of the separate yet common functions under a single message type. Without this common starting point we will not be able to use more than at most a single brand of manufactured RFID reader.

Step 3: Handling messages to the readers

This step is effectively the reverse engineering of step 2: that is we are trying to send messages to the readers in a format that they understand. To handle messages to

and from the readers we needed to know the specific message formats. These are also found with little effort from the manufacturer, although they may need to be imported at a disadvantageous stage in the architecture. Common uses of messages back to the reader involve changing sampling rates and ranges (if possible), and changing identification groups. We included the ability to change range, power level, and identification group into our middleware as a proof of concept.

When determining how the readers handle incoming messages there were several snafus that came about. These readers are quite temperamental in their state space, so without proper initialization steps a generic solution isn't feasible. Since this is beyond the scope of the problem, we decided to assume that the readers were initialized beforehand.

Deployment Architecture

There are five distinct layers of architecture that were addressed when looking at this problem. The lowest is the reader and tag sets, or the physical hardware. By nature of the project these are able to be changed in and out according to each specific site and shouldn't be considered to be a problem. The next level up is the interfaces to the physical readers, which change by brand. These need to be integrated into a certain framework so that like behaviors can be grouped together and used regardless of model. Above each reader sits a virtual reader. A virtual reader is a simulated reader that will act as a reader in all respects except that it doesn't physically exist. Instead it collects data from a number of physical readers, checks for any errors and miscounts, checks along the

network of virtual readers to make sure the counts are correct, and compiles this data in a meaningful format for the next layer up. Next is the path and name servers, which maintain the site in question.

We think of the problem in terms of graph theory, so that each path would be an edge between named readers, or vertices. With this graph model in mind we see the control flow of the system via the weights reported from the intercommunication between virtual readers, allowing for real time tracking of tags through a chaotic system. Furthermore we're able to use this path data to possibly communicate back and alter flows, allowing a significant amount of control over the system by making the path weights dynamic and in the control of the name and path servers. These physical servers have an interface so that their use case can be altered to fit any sort of applications needs, both of which are the final levels of this architecture. Now the scaling ability of this architecture comes from the name and path servers, which can be abstracted out into layers of name and path servers, each handling their own graph level, where the first would deal with nodes, the next up would deal with graphs, and the next dealing with graphs of graphs, etc. This sort of modeling works extremely well in conceptualization, and scales remarkably well.

Testbed

To test the idea of integrating active and passive RFID opcodes, we chose the M200 Mantis active reader from RFCODE and the M5e-compact passive reader from ThingMagic. These two readers were chosen due to their supposed compatibility

between their architectures. Both are mobile readers connected over a serial port. These initial similarities let us use the same device drivers to connect to the device to send information, as well as use them in the same testbed.

Again, we wanted to gauge effectiveness in the categories of efficiency, portability, usability, reliability, and functionality. The readers we chose were primarily chosen due to convenience, portability, functionality, and efficiency. That meant that usability and reliability were less important to the initial characteristics of our readers, specifically because we wanted to see if our middleware would improve reliability and create a usable framework.

Our testing area dealt with two stages, one for the active and one for the passive tags. The passive tags were placed on a grid along the carpet at 50 centimeter intervals. The active tags were placed on the large objects in the room, specifically a desk and the corners of the testing area. We tested each reader individually to determine that each could successfully detect tags.

For software usage, we decided to use the basis of the M5e-compact passive reader, the ARBSER package that came with it. This gave us a C++ implementation of a serial driver, the CRC code that the M5e used, and several data structures that were useful in determining the format of messages to and from the reader.

During development of the middleware we created some messaging systems for both the active and passive readers that was able to interchange between both systems in the same run. While this wasn't completely finished, it was able to interact with one or the other reader depending on which reader took the dominant USB port. Such a system

would be able to interchange between readers on the same station, allowing for the desired level of interchangeability.

Experimental Analysis

During testing of the readers we did an interesting range-finding test on the active reader. While these values may change based on the antennas being used by the reader, the analysis of the range patterns of the active reader are still valid. At 53 cm the reader held between 50 and 60 dB with the outliers dealing with the left side of the reader 30 degrees forward and back of center. At 100 cm we see those values increase to the mid 60 dB range, but we also saw the first lost tags, at 330, 150, and 180 degrees. This expresses a deficiency in our reader on the left side, and on the diagonals.

During these experiments there were a number of concerns that came up dealing with the hardware. The first of these is that the active reader has an interesting reading band in which the reading rate drops to 0%, which is placed between 1 and 1.2 meters. After that distance the reader continues to read until the final range distance, around 2 meters. The next hardware issue that we came across was the messaging system of the passive reader. If the passive reader handles the initial bootup process incorrectly then the state of the reader becomes lost, forcing it to require restarting. Another issue with the passive reader came in the reading range, which we had initially expected to be small, but ended up being absolutely minuscule, down to the tens of centimeters. While all of these problems did impede progress, there were ideas for solving all of these problems.

While the active reader may have an usual reading band, the idea is that the reader is in motion and thus will be able to pass through the area of uncertainty. The passive reader was dealt with first before anything with the less temperamental active reader, so the state of the passive reader on bootup was preserved. In so far as the passive reader's reading range, there is the possibility that may be the problem of the passive tags being used instead of the reader. Common profiles of the difference between the output of passive readers and passive tags shows an extreme difference between the activation power of the tags based on the reader. This may be able to be mitigated by switching out tags, but that remains to be tested.

Results

The ThingMagic reader uses a completely different architecture of opcodes from the RFCode Mantis series, meaning that any sort of similarities in the hardware were offset by the opcode design. The Mantis reader used very simple plaintext transfer protocols which allowed for easy setup, input and output from the reader. On the other hand the M5e series used a very complicated series of opcodes for setup, involving variable input and output fields, and didn't allow for any form of simple setup, tear down, or reading without using the built-in executables. This posed a major problem for the project, as it expressed the need for a standards body in the fields before any sort of integration efforts between active and passive RFID technologies can be seriously implemented. Without a standard of opcode between the same technologies there is

literally a nearly infinite combination of different opcodes that can be used for the same results.

Thus the initial usability problems were discovered, giving even more rise to the need of an effective middleware solution for rapid development for these products. This also effectively demonstrated that any middleware solution could not scale nicely without a standard, but scalability issues based on discrepancies in the opcodes are easy to handle in the long run.

Furthermore, the M5e series demonstrated how temperamental the hardware can be, as any faulty opcode transmitted on startup caused it to lose state, forcing it to be manually restarted. This sort of behavior caused any sort of device determination attempts to quickly result in failures, and would have to be a separate study in and of itself. While this is problematic for step 1, and ultimately led towards dropping step 1 as a major branch of the project, it didn't dismiss the primary issue, if active and passive RFID could be used with the same middleware. Again we chose to assume that the devices were determined beforehand and began testing.

However our study did lead to a few interesting discoveries. No matter what sort of RFID device is involved, we have determined that it will express at least the following fields, as stated in step 2.

M5e series	Mantis II series
Hardware specifics	
<ul style="list-style-type: none"> • 1 monostatic antenna • 10 – 23 dBm output • serial communications 	<ul style="list-style-type: none"> • 2 mono or bistatic antenna • Text oriented command base • serial or bluetooth communications

<ul style="list-style-type: none"> • Standard baud rate of 9600 dB • Binary oriented command base • Requires an appended CRC for proper communication • Returns tags on request 	<ul style="list-style-type: none"> • Standard baud rate of 115,200 dB • Returns tags as they arrive at set intervals
Bootup specifics	
<ul style="list-style-type: none"> • Requires bootloader • 5 instructions that must be put in order or else the reader loses state and needs to be restarted 	<ul style="list-style-type: none"> • Specify output format and start reading
Application specifics	
<ul style="list-style-type: none"> • Tag filtering and singulation • Get reader information • Read single or multiple tags • Write, lock, or kill tags • Get or Set antenna power • Get or Set power mode • Split read and returning tag data • Get or Set country settings 	<ul style="list-style-type: none"> • Group tags • Get reader information • Read tags • Filter tags by signal strength • Reset reader • Set timeout • Set output message format • Get direction information • Communicate over bluetooth • Get or Set country settings
Possible reporting fields	
<ul style="list-style-type: none"> • Tag ID • Read count • Signal Strength • Antenna ID • Frequency • Timestamp (Time it took to read) 	<ul style="list-style-type: none"> • Tag ID • Group ID • Payload • Lost tag • Signal Strength • Antenna ID • Timestamp • Tag movement

Furthermore there are some commonalities between setting these various fields, as well as some features allowing for filtering various tags. Unfortunately, even these discoveries come with some setbacks. The architectures of the two readers differed in not only the format of messaging, but also the method of reporting. Where the passive reader uses a windowing approach for collecting multiple tags data, the active reader uses

a streaming buffer to send out each tag as it is read. This caused several problems with synchronizing reading times, as well as creating a discrepancy between the format of how to read new tag data. While the read times didn't need to be synchronized, it certainly helps. As for the format of the tag data, fixing this proved to be a bit of a problem, as it led the project towards a mash up of two separate middlewares instead of creating a conjunctive middleware. This meant that we had to create two separate data structures for each hardware, as well as having separate opcodes and messaging systems.

Another interesting difference is in the timestamp field. While both readers use a timestamp, only the active reader uses it as an offset from the initial read-time. The passive reader instead uses the field as a timer between the beginning and the end of the read of a single tag, giving no information about the amount of time between reading different tags, nor giving a good estimate of when a tag was read in relation to another tag. This causes problems for any sort of model involving time-dependency, as there is no good frame of reference between the time of two reads. While this can be mitigated to some extent between read buffers, inside a buffer it is indeterminable as to which tag was read first.[8]

Conclusions and future work

The proposed idea is that active and passive RFID technologies are not as different in the software layer as they are in hardware. The idea is that there is a certain degree of overlap in the expectations of the technology that can be represented by a single layer of abstraction, allowing for multiple readers to be used under an umbrella

middleware. The key ideas of sending and receiving tag data is essentially the same between architectures, but the implementation differences turn out to be extremely difficult to mitigate. The fact that any RFID manufacturer can create their own unique opcode system and architecture limits the amount of flexibility and robustness of any sort of library of common functionality. In fact, the limits posed by the manufacturers even limit the ability to test which reader is currently connected, nevertheless test multiple readers on the same system.

Theoretically it is still possible to use multiple readers under one middleware, but getting a single middleware that is able to handle any type of reader is a problem that needs an agreed upon standard. Thus the proposed solution to integrating active and passive RFID technologies is to create an agreed upon standard that industry will adhere to in their architecture design.

Once an agreed upon standards can be maintained, changing between readers is fairly straightforward so long as there is a way of detecting which reader is currently connected. Thus future work would be along the lines of hardware detection and representation in the middleware layer. Since the capabilities of readers along the same family are fairly similar, a standard is not difficult to create. However, enforcing a standard upon industry is difficult even when each body agrees, and this project has clearly shown that industry is not in agreement of the current standards of opcodes. Another possible way of going about this research lies in using a different software package. The idea of using a simpler C# middleware still allows for viable usage, creating more unique data structures, and will handle communications in a more straightforward manner. That would also allow for more usability and modifiability for

the middleware, and faster development. The only downside to using a C# middleware, and the reason we didn't use one is because it would mean we would have to throw away all of the software we had.

Another field of future work involves the proposed step 3, with the creation of usability for other developers. While we created some rudimentary controls for other developers, such as changing power levels and reading ranges, we certainly can't predict all of the uses that can be made from the technology. Therefore there is a wide berth of opportunities towards developing new use cases and uses of the middleware, and all of the various changes that can be made to the readers capabilities while on the fly. Of particular note is the ability to gauge and measure frequencies of tags, a relatively simple tweak that may be of use.

References

- [1] Gi Oug, O., Doo Yeon, K., Sang Il, K. and Sung Yul, R. *A Quality Evaluation Technique of RFID Middleware in Ubiquitous Computing*. City, 2006.
- [2] Floerkemeier, C. and Lampe, M. RFID middleware design: addressing application requirements and RFID constraints. In *Proceedings of the Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies* (Grenoble, France, 2005). ACM.
- [3] Ahmed, N. and Ramachandran, U. Reliable framework for RFID devices. In *Proceedings of the Proceedings of the 5th Middleware doctoral symposium* (Leuven, Belgium, 2008). ACM.

- [4] Jie, W., Dong, W. and Huanye, S. *A Component-based Reconfigurable RFID Middleware*. City, 2007.
- [5] Liu, F., Han, L., LinKai and Ruan, Y. *OSGi-based Reconfigurable RFID middleware*. City, 2008.
- [6] Hashim, S. Z. M., Mardiyono, Kadir, W. M. N. W. and Anuar, N. *Multi Readers Detection in Adaptive RFID Middleware*. City, 2008.
- [7] Won-Ju, Y., Sang-Hwa, C., Seong-Joon, L. and Young-Sik, M. *Design and Implementation of an Active RFID System for Fast Tag Collection*. City, 2007.
- [8] ThingMagic, *RF Signal discussion*,
<http://www.thingmagic.com/forum/viewtopic.php?f=3&t=145&sid=746f1fd8837cdabf2f6cd615b083e668>