Facilitating RFID Development with the Accada Prototyping Platform

Christian Floerkemeier, Matthias Lampe and Christof Roduner
Institute for Pervasive Computing, Department of Computer Science,
ETH Zurich, 8092 Zurich, Switzerland
{floerkem|mlampe|rodunerc}@inf.ethz.ch

Abstract

The proliferation of radiofrequency identification systems in application domains such as supply chain management requires an IT infrastructure that provides RFID device and data management and supports application development. In this paper, we discuss the requirements towards such an infrastructure and present the freely available Accada RFID Prototyping Platform. The Accada platform manages readers, filters and aggregates RFID data, and helps to interpret the captured RFID data in an application context. Our RFID infrastructure implements the specifications defined by the EPCglobal RFID community, such as the reader protocol, the application-level-events specification and the EPCIS capture and query interfaces. We believe that this freely available RFID infrastructure will allow the research community to evaluate novel concepts and applications more quickly by significantly lowering the barrier for large-scale, real-world testing.

1 Introduction

Radio Frequency Identification (RFID) technology has recently seen growing interest not just from the research community, but also from a wide range of industries such as retail, pharmaceutical, and defense [6]. In these domains, RFID technology holds the promise to eliminate many existing business problems by bridging the economically costly gap between the virtual world of IT systems and the real world of products and logistical units. Common benefits include more efficient material handling processes, elimination of manual inventory counts, and the automatic detection of empty shelves and expired products in retail stores [1].

In traditional RFID applications, such as access control, there was little need for an RFID middleware because the RFID readers were not networked and the RFID data were only consumed by a single application. In novel application domains, such as supply chain management and logistics, there is no longer a 1-to-1 relationship between reader and application instance, however. In these domains many readers distributed across factories, warehouses, and distribution centers capture RFID data that need to be disseminated to a variety of applications. This introduces the need for an RFID infrastructure that hides proprietary reader device interfaces, provides configuration and system management of the reader devices and filters and aggregates the captured RFID data. The result is that applications no longer need to maintain connections to individual reader devices or even need to know how to trigger a read cycle at a particular RFID reader device.

From an application development perspective, it is also important to abstract from the low level RFID data captured and translate them into more meaningful application. The detection of an RFID tag with tag ID 3455.3454656 by reader 8745653 would thus result in the corresponding business event that a shipment of razor blades arrived at dock door 14 of the warehouse. Many benefits commonly associated with RFID require sharing these business events across the supply chain [1].

In this paper, we analyse these application requirements in detail and present Accada[1], an RFID prototyping platform that aims to address these requirements. Accada is an open source RFID infrastructure released under a BSD-style license which implements the interfaces defined in the EPC Network specifications of EPCglobal[2]. The EPC Network, originally proposed by the Auto-ID Center [16] and further developed by the members of EPCglobal, is currently the predominant standardization effort of the RFID community. The objective behind the Accada project is to provide a common codebase for the exploration of novel applications and educational projects. We believe that the availability of a free RFID infrastructure that implements current industry standards is of great importance for universities and research institutions because it facilitates the testing of novel concepts and advances the development of what has been coined “an Internet of Things”.

[1] www.accada.org
[2] www.epcglobalinc.org

The rest of this paper is structured as follows. In Section 2, we detail the requirements toward an RFID infrastructure. In Section 3, we describe our Accada RFID prototyping platform and show how our implementation addresses these application needs. In Section 4, we review limitations of our implementation and present future work. Before we conclude in Section 6, we discuss related work in Section 5.

2 Application Requirements

Based on an analysis of different RFID applications and the study of other work on RFID middleware [3, 5, 9, 14, 15] we identified the following requirements an RFID middleware should meet:

RFID data dissemination. The information captured by a reader is usually of interest not only to a single application, but to a diverse set of applications across an organization and its business partners. The captured RFID data must thus be broadcast to the entities that indicated an interest in the data. Due to the event-driven nature of many processes observed with the help of RFID systems, there is a need to support asynchronous messaging as well as a query-response model. Different applications also require different latencies. Applications that need to respond immediately to local interaction with the physical objects require a short notification latency that is comparable to the observation latency. Legacy applications that are not designed to handle streaming data might need to receive batched updates on a daily schedule.

RFID data aggregation. RFID systems generate significant amount of data that can be aggregated in a number of different ways. RFID data can be aggregated in the time domain, e.g. by generating entry and exit events, and in the space domain, e.g., by combining data from different readers and reader antennas that observe the same location or by detecting the movement of a tagged object. Since RFID permits identification at the instance-level rather than at the class-level, there is also the possibility to report the quantity of objects belonging to a specific category.

RFID data filtering. A common feature of all applications that make use of the captured data is the desire to receive filtered RFID events rather than all RFID data captured. Different applications are interested in a different subset of the total data captured, based on the reader, reader antenna, and tag involved.

Writing to a tag. Some tags feature not only memory space for an identifier, but for additional data. RFID middleware should thus provide means to write to and read from this additional memory. This additional memory can then be used to store application data such as expiry dates in order to facilitate data exchange, where no network access is available. In a broader sense, writing to a tag also includes the initial write to the tag to program its ID, and killing a tag to permanently disable it.

Trigger RFID readers by external sensors. In many applications it is not mandatory to operate RFID readers continuously. Due to the limited bandwidth available, it is even undesirable to have readers transmit, while no tags are present [9]. To initiate the tag inventory process at a reader when there are tagged objects arriving in the read range, external sensors, such as motion sensors, should thus be able trigger the RFID readers.

Fault and configuration management. The proliferation of readers mandate fault and configuration management. This includes monitoring the health of RFID readers and accessing the RFID reader configurations remotely.

Heterogeneous reader landscape. The diverse computing and networking capabilities of readers is also an important RFID consideration when developing RFID infrastructure support. Low cost readers usually support only a single antenna and a serial RS232 interface. These reader types are connected to a computer which hosts the application directly or forwards the captured data over a network connection. More sophisticated reader devices support several antennas, a TCP host interface, and ample computing resources for on-device data processing, such as filtering and aggregation.

RFID data interpretation. From an application perspective, it is also desirable to provide a mechanism that interprets the captured RFID data in a given business context and that turns the low level RFID event into the corresponding business events. The detection of a number of tags at a dock door can thus be translated into a shipment complete event automatically.

Sharing of RFID triggered business events. Many benefits commonly associated with RFID require data sharing across the supply chain [1]. In order to realize those benefits, an infrastructure that captures RFID triggered business events and makes them available to authorized parties is essential.

Other application requirements that relate to security and performance are not discussed here in detail, since they are not unique to RFID. In the following section, we show to what extent our Accada RFID Prototyping Platform addresses these RFID requirements.

3 Accada – an open source RFID prototyping platform

The goal of the Accada project is to develop a RFID prototyping platform that meets the application needs mentioned in the previous section and thus provides the research community with a testbed for RFID experiments. Our implementation is based on the standardized interfaces published by the predominant RFID standardization body –

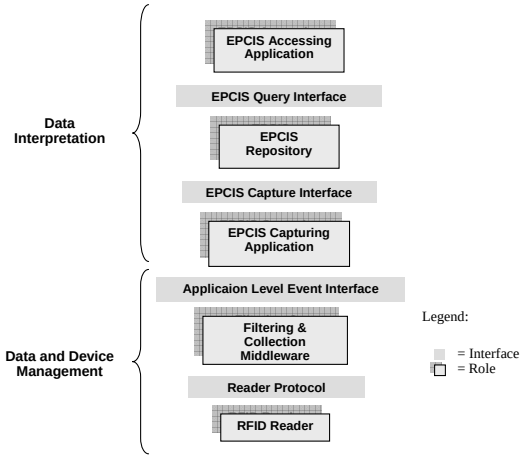

Figure 1. *EPC Network Roles and Interfaces [2] (Source: EPCglobal)*

EPCglobal. The Accada platform consists of three separate modules: the reader, the filtering and collecting middleware, and the EPC information service (EPCIS) module. These modules implement the corresponding roles in the EPC Network (cf. Figure 1), which is named after the standardized tag identifier code, electronic product code (EPC), which is stored on every tag. We begin our description with the reader module, which performs data dissemination, filtering, and aggregation at the reader level. We then continue with our filtering and collection middleware, which decouples readers and applications and provides additional aggregation and filtering functionality. The third module relates to the EPCIS part of the EPC Network, which deals with an application framework to interpret the captured RFID data in an application context.

3.1 Reader module

The reader module of the Accada platform implements the EPCglobal Reader Protocol [8]. Our implementation features all mandatory and optional features defined in the Reader Protocol specification. This includes filters on tag ID and reader antenna, time aggregates, such as entry and exit events, and space aggregates, where multiple reader antennas can be logically grouped to a single source. There is also support for writing to tags and external triggers such as motion sensors. The data captured can be disseminated via a query-response and a publish-subscribe mechanism using different message transport bindings (MTB) as shown in Figure 2. Our reader implementation also features a Reader Proxy that implements the host side of the EPCglobal Reader Protocol. This facilitates application development because interaction with the resource limited reader device is now as straightforward as a remote procedure call.

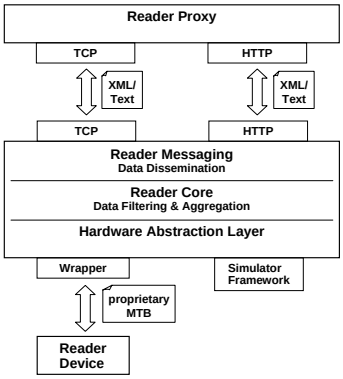

Figure 2. *Accada reader implementation.*

The Accada reader implementation can be used in three different modes. In the first mode, the reader implementation is deployed on a separate server and wraps a proprietary RFID reader protocol using the built-in hardware abstraction layer (cf. Figure 2). Our implementation currently supports a variety of reader devices from different manufacturers. To facilitate testing of RFID applications, when there is no reader connected, the Accada reader can also be used in a simulation mode. The built-in simulation framework that supports this mode includes a graphical user interface that allows the developer to drag and drop tags over different reader antennas (cf. Figure 3). The simulation framework also provides a mechanism to schedule the detection of a tag at different times on different reader instances. This allows a developer to simulate the movement of a tag population through the supply chain. In the third mode, the Accada reader implementation can also be deployed on an RFID reader itself to provide data dissemination, filtering, and aggregation capabilities.

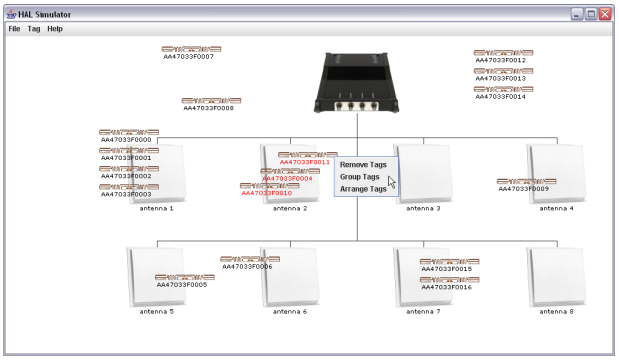

Figure 3. *Screenshot of Accada reader simulator.*

3.2 Filtering & Collection Middleware module

The Accada filtering & collection middleware represents a single interface to the potentially large number of readers that make up an RFID system deployment. This allows applications to define a subscription, which is then used to configure the corresponding reader devices using the EPCglobal reader protocol (cf. Figure 4). Once the readers capture relevant tag data, the readers notify the middleware, which combines the data arriving from different readers in a report that is sent according to a pre-determined schedule to the subscribed applications. Since the middleware receives data from multiple readers, it can provide additional aggregation functionality. Redundant read events from different readers observing the same location can thus be omitted, reducing the amount of filtering and aggregation required in any application interpreting the captured RFID data. The interface between the filtering & collection middleware and a host application is based on the EPCglobal Application Level Events (ALE) Specification [7]. The ALE specification defines a SOAP MTB for the subscription communication channel and a XML and TCP/HTTP MTB for the notification channel.

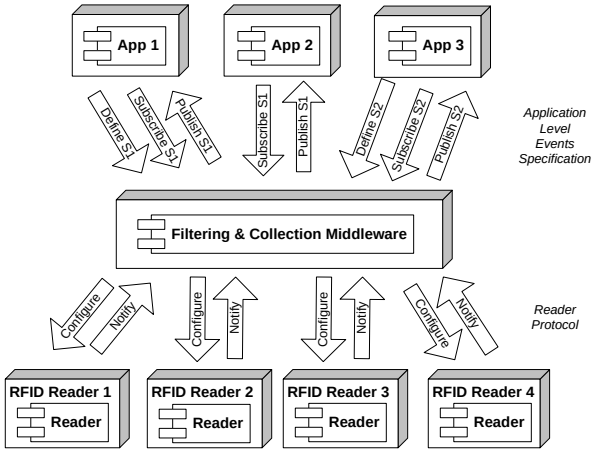

Figure 4. *Filtering&Collection Middleware.*

3.3 EPCIS module

The EPC Information Service (EPCIS) [2] of the EPC Network is the component that is responsible for receiving the application-agnostic RFID data from the filtering&collection middleware, translating them into the corresponding business events, and making those business events available. The EPC Information Service itself consists of three parts (cf. Figure 1): an EPCIS capture application that interprets the captured RFID data, an EPCIS repository that provides persistence, and EPCIS query application that retrieve the EPCIS events from the repository.

The Accada EPCIS module provides sample capture and query applications that implement the corresponding interfaces and and EPCIS repository that uses a relational database to store the EPCIS events(cf. Figure 5).

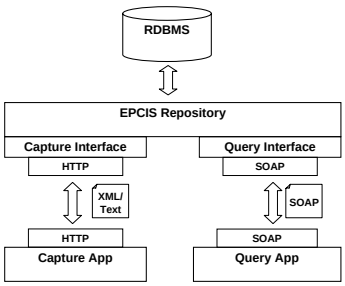

Figure 5. *Accada EPCIS module featuring an EPCIS Repository, and Capture and Query Applications.*

4 Discussion, Limitations and Future Work

The Accada reader implementation addresses the majority of the requirements outlined in Section 2 that are applicable to a reader. This includes data dissemination, filtering, and aggregation, writing to tags and external triggers. Due to our surrogate concept, where the Accada reader software is deployed on a separate (embedded) computer and connects to reader devices via our hardware abstraction layer, our implementation also addresses the heterogeneity of the reader landscape and in particular the reader devices with proprietary MTBs and limited resources. While we support a number of different reader devices in this mode, there is still a large number of reader devices which we do not support. We hope that other researchers will see the benefit of the existing open source Accada implementation and contribute drivers for the reader devices they use.

There are currently only limited fault and configuration management capabilities. We are currently implementing the EPCglobal Reader Management Protocol, which provides this functionality. Once implemented, a Accada reader can be managed and configured just like any other IT equipment via the Simple Network Monitoring Protocol. The Accada reader implementation also uses the standard edition of SUN's Java Virtual Machine rather than a microedition. The result is that the Accada reader implementation can only be embedded on those reader devices that feature significant computing resources. In the future, we

might consider porting the software to be compliant with one of the Java profiles for limited devices.

The EPCglobal Reader Protocol Version 1.1. currently consists of a large number of optional features. This design choice is a direct result of the heterogeneity of the reader landscape, where reader devices with significant computing resources were envisioned to provide the majority of the optional features and low-end reader devices would only support the mandatory features and possibly a small number of the optional features. However, this makes application development a significant challenge, since there is no service discovery implemented, which lists the supported features. Future versions of the EPCglobal Reader Protocol might want to address this issue by providing different reader profiles with corresponding feature sets. The EPCglobal Reader Protocol Version 1.1, which we implemented, also lacks configuration support for air interface settings, such as transmit power, noise levels, and data rates.

The XML MTB of the EPCglobal Reader Protocol also leaves room for improvement. The message binding does not return serialized objects to the host, but forces the host to request one class variable at a time. There is also no way to configure a reader with a single command.

The Accada filtering&collection middleware supports data dissemination, filtering and aggregation. At first sight, this seems redundant to the capabilities provided at the reader level. It is however to support filtering and aggregation a both levels. On the one hand, there might be readers that do not support the optional filter and aggregation functionality due to limited computing resources. On the other hand, the middleware is capable of aggregating information captured by a number of different readers. It would be desirable though that the Reader Protocol and Application Level Events Specification would use the same nomenclature and implement common functionality in the same way.

5 Related Work

The need for an RFID infrastructure and the application requirements towards such an RFID infrastructure have been discussed in a number of publications [3, 9, 15]. Initially, the concept of a distributed networking infrastructure for RFID was proposed by the Auto-ID Center, an industry-sponsored research program to foster RFID adoption [16], which coined the term EPC Network.

Our work is closely related to a component in an now outdated architecture of the EPC Network, which was called Savant [11]. While the Savant software addressed some of the application requirements presented in Section 2, e.g. it featured functionality for coping with the idiosyncrasies of different kinds of readers and for cleaning the data, there was only limited built-in functionality that specifically addresses data dissemination, filtering and aggregation. In the Savant implementation, there was also no predefined subscription language. Instead, the Savant concept allowed applications to register event filters programmed in the Java programming language, which could operate on a combination of notifications. This approach increases the expressiveness of the subscription language at the expense of performance and scalability [4]. We believe that a predefined subscription language as provided by [7, 8] is expressive enough to support RFID filtering and aggregation. There is no need for the additional flexibility provided by user-defined operators programmed in the Java programming language.

There are a number of other commercial and non-commercial RFID middleware products available, among others [3, 10, 12, 13]. All of them address to some extent the application requirements for device and data management and data interpretation listed in Section 2. To our knowledge, none of them incorporate the standardized interfaces of the EPCglobal community to the extent shown in this paper.

The filtering & collection middleware functionality of our Accada platform differs from the one commonly found in publish-subscribe systems. In traditional publish-subscribe systems there is no feedback path from the messaging service to the producers. Due to the restricted bandwidth available to the RFID readers on the air interface and the potential large amount of data they produce [9], the subscriptions of the applications are fed back to the event producing reader instances. The result is that reader instances only disseminate data to the messaging service which are of interest to the applications. The event router Elvin [17] uses a similar kind of feedback mechanism called quenching. Different to most other publish/subscribe messaging systems such as [4, 17], the subscription process used in our middleware consists of two steps. In an initial step applications define a subscription, where they specify notification latency, aggregates and filters. In a second step, applications can then simply subscribe to a previously defined subscription.

6 Conclusion

In novel RFID application domains, such as supply chain management and logistics, there are many RFID readers distributed across factories, warehouses, and distribution centers capturing RFID data that need to be disseminated to a variety of applications. This introduces the need for an RFID infrastructure that hides proprietary reader device interfaces, provides configuration and system management of the reader devices and filters and aggregates the captured RFID data.

In this paper, we discussed these application requirements in detail and presented Accada, an open source RFID

prototyping platform that aims to address these application requirements. The paper showed that the current Accada implementation, which is based on a set of specification developed by the EPCglobal community, addresses the majority of the application requirements, but still lacks capabilities such as fault and configuration management. Our work also showed that there is room for improvement within the EPCNetwork standards, when it comes to a common nomenclature, support for writing to a tag, and the configuration of reader air interface parameters. The paper also showed that a publish-subscribe approach that provides an additional feedback path to the event producers is appropriate for RFID middleware design.

References

[1] K. Alexander, T. Gilliam, K. Gramling, M. Kindy, D. Moogimane, M. Schultz, and M. Woods. Focus on the Supply Chain: Applying Auto-ID within the Distribution Center. Technical Report IBM-AUTOID-BC-002, Auto-ID Center, 2002.

[2] Architecture Review Committee. The EPCglobal Architecture Framework. EPCglobal, July 2005.

[3] C. Bornhövd, T. Lin, S. Haller, and J. Schaper. Integrating Automatic Data Acquisition with Business Processes - Experiences with SAP's Auto-ID Infrastructure. In *Proceedings of the 30st international conference on very large data bases (VLDB)*, pages 1182–1188, Toronto, Canada, 2004. VLDB Endowment.

[4] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 219–227, Portland, Oregon, July 2000.

[5] S. Chawathe, V. Krishnamurthyy, S. Ramachandrany, and S. Sarma. Managing RFID Data. In *Proceedings of the 30st international conference on very large data bases (VLDB)*, pages 1189–1195, Toronto, Canada, 2004. VLDB Endowment.

[6] Economist. The Best Thing since the Bar-Code, Feb 2003.

[7] EPCglobal. The Application Level Events (ALE) Specification, Version 1.0, Sep 2005.

[8] EPCglobal. Reader Protocol Standard, Version 1.1, Jun 2006.

[9] Christian Floerkemeier and Matthias Lampe. RFID middleware design – addressing application requirements and RFID constraints. In *Proceedings of SOC'2005 (Smart Objects Conference)*, pages 219–224, Grenoble, France, October 2005.

[10] Sun Microsystems. Java System RFID Software 3.0 Developer Guide. www.sun.com, Feb 2006.

[11] Oat Systems and MIT Auto-ID Center. The Savant Version 0.1. Technical Report MIT-AUTOID-TM-003, Auto-ID Center, 2002.

[12] OATsystems. OAT C4 Architecture. www.oatsystems.com, 2006.

[13] B. S. Prabhu, Xiaoyong Su, Harish Ramamurthy, Chi-Cheng Chu, and Rajit Gadh. WinRFID – A Middleware for the enablement of Radio Frequency Identification (RFID) based Applications. In Rajeev Shorey and Chan Mun Choon, editors, *Mobile, Wireless and Sensor Networks: Technology, Applications and Future Directions*. Wiley, 2005.

[14] Kay Römer, Thomas Schoch, Friedemann Mattern, and Thomas Dübendorfer. Smart Identification Frameworks for Ubiquitous Computing Applications. *Wireless Networks*, 10(6):689–700, December 2004.

[15] Sanjay Sarma. Integrating RFID. *ACM Queue*, 2(7):50–57, 2004.

[16] Sanjay Sarma, David L. Brock, and Kevin Ashton. The Networked Physical World – Proposals for Engineering The Next Generation of Computing, Commerce & Automatic Identification. Technical Report MIT-AUTOID-WH-001, MIT Auto-ID Center, 2000.

[17] Bill Segall, David Arnold, Julian Boot, Michael Henderson, and Ted Phelps. Content Based Routing with Elvin4. In *Proceedings AUUG2K*, Canberra, Australia, June 2000.