

UCLA

UCLA Electronic Theses and Dissertations

Title

RFID Asset Management Solution with Cloud Computation Service

Permalink

<https://escholarship.org/uc/item/1bd1s73m>

Author

Chattopadhyay, Arunabh

Publication Date

2012

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

RFID Asset Management Solution with Cloud Computation Service

A thesis submitted in partial satisfaction of the requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

by

Arunabh Chattopadhyay

2012

© Copyright by

Arunabh Chattopadhyay

2012

ABSTRACT OF THE DISSERTATION

RFID Asset Management Solution with Cloud Computation Service

by

Arunabh Chattopadhyay

Doctor of Philosophy in Mechanical Engineering

University of California, Los Angeles, 2012

Professor Rajit Gadh, Chair

This work proposes the deployment of a web-based architecture of an RFID tracking solution. An extensive literature survey has been conducted to find existing RFID web based architectures and the current requirements of supply chain based organizations. These findings were used to design a web based RFID tracking system with a goal of reducing set up time and infrastructure for the client. The architecture attempts to depart from the typical localized approach of most RFID middlewares. The middleware has been replaced by a web service residing in a cloud. The traditional roles of the middleware such as tag data filtering, sorting, storage and event processing have been redistributed amongst an RFID reader-based application and the web service. This architecture provides more flexibility and control to large scale RFID-based supply chain operations which are global in nature and unsuitable to the traditional localized

middleware based architecture of RFID tracking solutions. The feasibility of the web based architecture is further augmented by validating performance parameters such as tag processing and primitive and complex event processing delay. Different kinds of RFID readers are also tested to ensure the suitability of the architecture. Different sub-architectures are proposed based on performance data for different scales of operations. Event processing capability on a reader device is demonstrated and tested, where occurrence of certain specified sequence of events results in automated notifications sent to the clients by means of e-mail, text messaging etc. It is concluded that for small number of tag operations (<400) or small scale businesses, performing complex event processing at the reader level is faster and more economical. For large scale tag operations (>400), performing only partial event processing at the reader level leads to lowest event trigger delays.

Finally, the event processing algorithm is demonstrated as a proof of concept for an RFID enabled parking lot to control and schedule charging and discharging of EV's and inform their users of the charge status. The RFID triggered events are used to automatically set charge/discharge preferences of the EV owner for the duration for which the EV is parked. This was supplemented by designing a charge/discharge algorithm to optimize the consumption of energy by parking lots.

The thesis of Arunabh Chattopadhyay is approved.

Eric P. Y. Chiou

Mario Gerla

Daniel Yang

Rajit Gadh, Committee Chair

University of California, Los Angeles

2012

iv

Table of Contents

1)Introduction.....	1
2) Motivation and Research Outline.....	6
2.1) Current Issues.....	9
2.2) Overall Research Objective.....	10
2.3) Research Proposal.....	12
2.4) Stages of RFID tag transport.....	16
2.5) Test-Case Study.....	17
2.6) Proof-of-Concept.....	17
3) Literature Review.....	20
3.1) Supply Chain based aspects addressed by the EPC.....	20
3.2) Database Research in RFID.....	21
3.3) Architecture Research in RFID.....	27
4) System Architecture.....	34
4.1) Reader Resident Application (RRA)	35
4.1.1) Role of RRA.....	36
4.1.2) Data structure.....	40
4.1.3) Sequence of operation.....	41
4.1.4) Message structure and transmission.....	43
4.2) Cloud Service.....	45
4.2.1) Database	45

4.2.2) Event Notification Service.....	47
4.2.3) Reader Web Service (RWS)	47
4.2.4) Sequence of Operation.....	47
4.3) Event Processing.....	49
4.3.1) RRA.....	50
4.3.2) RWS.....	50
5) Web based User Interface.....	54
5.1) Main Page.....	56
5.2) Reader Administration Page.....	60
5.3) Events Administration Page.....	62
5.4) User Administration Page.....	67
5.5) History Search Page.....	71
6) Load Testing.....	72
6.1) Boxusage consumption by RWS.....	74
6.2) Tag processing by RWS.....	75
6.3) Primitive Event Processing.....	79
6.4) Complex Event Processing.....	86
7) Discussion.....	92
7.1) Data Security and Privacy.....	92
7.2) Network/Connectivity Failure.....	93
7.3) Large data traffic from multiple readers.....	94
7.4) Customization and Potential Applications	95

8) Implementation of an RFID enabled Electric Vehicle Parking Lot.....	98
8.1) Literature Review.....	99
8.1.1) Radio-Frequency Identification (RFID)	99
8.1.2) Charge Scheduling.....	100
8.1.3) Vehicle-to-Grid (V2G)	102
8.2) System Architecture.....	104
8.2.1) Overview.....	104
8.3) Aggregated Charge Scheduler.....	106
8.3.1) Overview.....	106
8.3.2) Algorithm Description.....	107
8.4) Results.....	108
8.4.1) Overview.....	108
8.4.2) Simulation Setup.....	108
8.4.3) Charge Scheduling.....	110
8.4.4) V2G Profit.....	111
8.5) Conclusion.....	115
9) Conclusion.....	116
References.....	117

Table of Figures

Fig. 1.1: RFID Reader.....	2
Fig. 1.2: RFID tags.....	2
Fig. 1.3: RFID System.....	2
Fig. 1.4: Predicted Investments to be made in RFID in the coming years.....	5
Fig. 2.1: Desired Web Service Architecture.....	11
Fig. 2.2: Proposed Web Service System Architecture.....	16
Fig. 2.3: Initial Setup for proof of concept.....	18
Fig. 2.4: Testing scalability of the proposed web service architecture.....	19
Fig. 3.1: RFID Proprietary Web Service Architecture.....	26
Fig. 4.1: Symbolic system architecture of the proposed RFID.....	34
Fig. 4.2: Reader Resident Application workflow.....	35
Fig. 4.3: Reader Resident Application (RRA)	37
Fig. 4.4: Message Compilation time by RRA as a function of number of tags.....	38
Fig. 4.5: Data model of the RFID solution.....	40
Fig. 4.6: Sequence diagram of the proposed RFID system.....	42
Fig. 4.7: Body of SOAP message.....	43
Fig. 4.8: Reader Web Service (RWS) flowchart.....	48
Fig. 5.1: Web based interface for user.....	55
Fig. 5.2: User login page.....	55
Fig. 5.3: Creating an event on the user interface.....	59
Fig. 5.4: Using the Reader Administration page of the user interface.....	61
Fig. 5.5: Event Table page.....	63
Fig. 5.6: Creating complex rules from simple rule on the event.....	65

Fig. 5.7: User administration part of the user interface.....	67
Fig. 5.8: Modification of user privileges by the administrator.....	68
Fig. 5.9: Using the history search page of the user interface.....	70
Fig. 6.1: Delay performance comparison of localized server and cloud based web server.....	73
Fig. 6.2: Bar plot of boxusage of various reader web service.....	74
Fig. 6.3: Cloud database boxusage as a function of number of tags.....	76
Fig. 6.4: Message processing duration as a function of number of tags.....	77
Fig. 6.5: Symbolic system architecture of the RFID web service with a web based server.....	78
Fig. 6.6: Processing delay of RRA as a function of number of tags and number of events.....	80
Fig. 6.7: Processing delay of the RWS as a function of number of events and number of events.....	81
Fig. 6.8: Box usage of the RWS processing events as a function of number of events.....	82
Fig. 6.9: Combined processing delay of RRA & RWS as number of tags and events vary.....	83
Fig. 6.10: Difference in combined processing delay as number of tags and events vary.....	84
Fig. 6.11: Processing delay for evaluating primitive events.....	85
Fig. 6.12: Flowchart for Complex Event Processing by the RRA complex event	87
Fig. 6.13: Comparison of different Complex Event Processing Techniques.....	89
Fig. 6.14: Comparison of Complex Event Processing delay.....	91
Fig. 8.1: Parking garage access gate.....	103
Fig. 8.2: Charging station equipment and activities.....	104
Fig. 8.3: Event sequence for EV arrival at parking garage.....	105
Fig. 8.4: Electricity price [76]	106
Fig. 8.5: Actual Load [76]	108
Fig. 8.6: Charging schedule for 30 cars, 10 chargers.....	109
Fig. 8.7: V2G schedule for 30 cars, 10 chargers.....	109
Fig. 8.8: V2G profit per car as a function of the number of intervals per hour.....	111

Fig. 8.9: V2G profit (ϕ) contour plot for variable scenario.....112

Fig. 8.10: V2G profit (ϕ) contour plot for enterprise commuter scenario cars.....113

Table of Tables

Table 1.1: Types of Tags.....	3
Table 1.2: Frequency of Operation.....	3
Table 1.3: Standards in RFID.....	4
Table 3.1: RFID infrastructure-software vendor data architectures.....	25
Table 5.1: User privileges table.....	69
Table 6.1: Comparison of algorithms for Complex Event Processing for 1 event.....	89
Table 6.2: Comparison of algorithms for Complex Event Processing for 1001 events.....	90

Vita

Arunabh Chattopadhyay

Academic Qualification:

- **PhD candidate, Mechanical Engineering, University of California Los Angeles** Oct 2007 – June 2012
Major: Manufacturing Engineering, Minor: Systems & Controls

PhD thesis: RFID Asset Management Solution with Cloud Computation Service
- **M Tech, Electrical Engineering, Indian Institute of Technology, Kanpur, India**
June 2005 – June 2007

Major: Microwave & Photonics

M Tech thesis: RFID Indoor location tracking based on passive UHF RFID
- **B Tech, Electronics & Telecommunication Engineering, Jamia Millia Islamia, New Delhi, India**
June 2001 – June 2005

Work Experience:

University of California Los Angeles - Graduate Student Researcher October 2007 - present

- Implemented Windows Mobile based application on RFID reader to read RFID tags and process and transmit collected data to the local/cloud based web service through SOAP messages.
- Implemented ASP .NET web service deployed on IIS and Amazon EC2 to process, store RFID tag data and perform rule matching for client event notification.
 - Data storage/update was performed on MS SQL and Amazon SimpleDB.
- Using high volume simulated RFID data, extensive stress tests were performed on the web service
- Amazon Simple Notification Service implemented to notify clients of subscribed RFID based event occurrences.
- Web based monitoring & control application implemented to allow client to view data in real time and remotely control RFID readers in the field.
- Advanced Encryption Standard (AES) used to secure communication between RFID readers and the web server and HTTPS implemented in monitoring & control application to allow for commercial adoption of the architecture.

- Lecture graduate level course in RFID and undergraduate level course in CAD design.
- Design assignments, programming assignments, lab session presentations and videos and exam question papers, grading assignments and exams, holding additional discussion sessions.
- Prepared a CAD course manual to construct a fully functional bicycle and perform stress analysis for a senior year CAD course.

Publications:

- Arunabh Chattopadhyay, Jeremy Stentz, Rajit Gadh, “Modeling an RFID Asset Management Solution hosted on a Cloud Service and its Characterization”, ASME Journal of Computing and Information Science in Engineering (Submitted).
- Arunabh Chattopadhyay, B. S. Prabhu and Rajit Gadh, “Web based RFID Asset Management Solution established on Cloud Services”, IEEE International Conference on RFID-Technology and Applications, September 2011.
- Katina Michael, George Roussos, George Q. Huang, Arunabh Chattopadhyay, Rajit Gadh, B. S. Prabhu and Peter Chu, “Planetary-Scale RFID Services in an Age of Uberveillance”, Proceedings of the IEEE.
- S. Mal, A. Chattopadhyay, A. Yang, R. Gadh, “Electric vehicle smart charging and vehicle-to-grid operation”, International Journal of Parallel, Emergent and Distributed Systems, vol. 27, no. 3. March 2012.
- Riccardo Mogre, Rajit Gadh and Arunabh Chattopadhyay, “Using survey data to design a RFID centric system for hospitals”, Service Science, Vol. 1, No. 3.
- Arunabh Chattopadhyay, “Feasibility study through design and implementation of a Web Services based RFID Solution”, Third Annual Google Ph.D. Forum on Pervasive Computing and Communications, March 2010.
- Arunabh Chattopadhyay and Ayyangar R. Harish, “Analysis of UHF passive RFID tag behavior and study of their applications in Low Range Indoor Location Tracking” Antennas and Propagation Society International Symposium 2007, IEEE.
- Arunabh Chattopadhyay and Ayyangar R. Harish, “Analysis of UHF Low Range Indoor Location Tracking Techniques using Passive UHF RFID Tags” IEEE Radio and Wireless Week 2008.

Chapter 1

INTRODUCTION

In this section, the first sub-section gives a brief overview of the technologies involved in this work. The second sub-section explains the motivation for the work and how this work is different from the previous approaches of RFID middlewares and the benefits of such an approach. We briefly go over the challenges faced for such architecture. Finally, we give a use case example where such architecture could be useful. A. Terminology overview

RFID (Radio Frequency IDentification) for the past decade has been playing an important role in various kinds of supply chain based industrial applications such as warehousing, pharmaceutical tracking, retailing etc. An RFID system consists of a single or multiple readers which communicate with multiple tags by inquiring their identification (ID) as shown in Fig. 1.1, 1.2 and 1.3 [1], [2]. This technology has been widely adopted in supply chain systems to streamline the flow of information regarding identification, arrival/departure, status etc of objects in the system. The tags are periodically queried by their respective readers, and in turn notify a software service/middleware of their presence or absence so as to obtain some information which can be further processed to deliver useful services. Memory in tags can be read and also written into by the reader for storage to be retrieved later. Based on energy utilization, tags may be categorized into three: Passive, Active and Semi- Passive depending upon their power consumptions [3].



Fig. 1.1: RFID Reader



Fig. 1.2: RFID tags

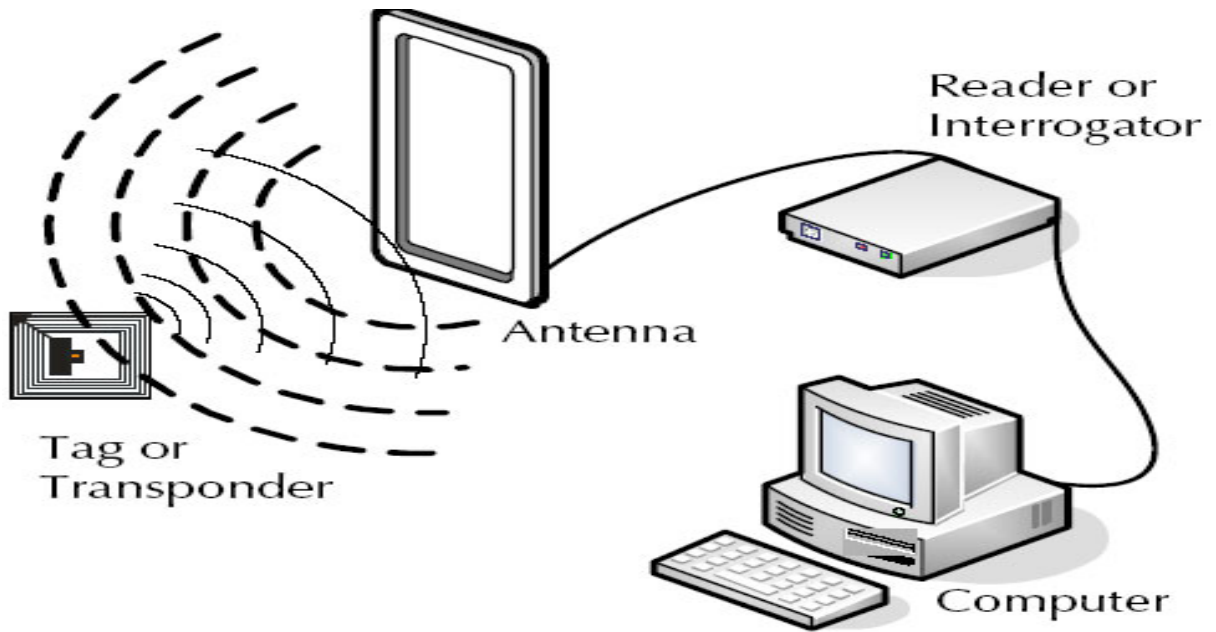


Fig. 1.3: RFID System

Table 1.1 shows the variation of behavior of RFID tags as their behavior changes from Passive to Active [4]

Tag type	Passive	Semi-Passive	Active
On board battery	No	→ Yes	
Cost	Cheap	→ Expensive	
Range	Low	→ High	
Read time	Large	→ Small	
Tag Life	Long	→ Short	
Cost	Cheap	→ Expensive	

Table 1.1: Types of Tags

Frequency Range	LF 125•134 KHz	HF 13.56 MHz	UHF 860•960 MHz	Microwave 2.45 & 5.8 GHz
Data Rate/Speed	Slower	↔		Faster
Ability to read near metal or liquids	Better	↔		Worst
Antenna Coil	Longer	↔		Shorter

Table 1.2: Frequency of Operation

RFID tags are also categorized based on the frequency at which they operate such as Low Frequency (LF), High Frequency (HF), Ultra High Frequency (UHF) and Microwave depending on the application [5]. The different properties of the different frequency spectrums are listed on

Table 1.2 [6]. Many different RFID standards were created, keeping in mind the different applications of this technology. The major three standards have been listed in Table 1.3 [7]. However, with the adoption of EPC Global Standards, RFID in the UHF band is widely used by organizations in supply chain management [8]. Wal-mart had a big role to play in the growth of RFID in the supply chain system as it mandated the incorporation of pallet level tracking of its inventory throughout the supply chain [9]. This was followed by the Department of Defense mandate on tagging items that are supplied by Defense contractors [10]. The state of California has also issued e-pedigree compliance requirements for pharmaceutical companies to authenticate the origin (pedigree) of drugs supplied by them, in order to prevent counterfeit drugs [11].

Standards	EPC	ISO
Frequency	UHF only	LF-Microwave
Access	Open	Not open
Applications	Dominates the retail market domain	All domains other than UHF such as animal tagging, smart cards etc
Latest UHF standard	EPC Class 1 Gen 2	ISO 18000-6
Interoperability issues solved?	Recently	

Table 1.3: Standards in RFID

A web service is any service that is available over the internet, is not tied to any programming language or operating system and uses a standardized XML (Extensible Markup Language) messaging system [12]. One example of XML based messaging system is SOAP (Simple Object Access Protocol) [13]. A web service based solution provides a standard means of inter-operation between different software applications, running on a variety of platforms and/or frameworks. Cloud computing can be defined as applications delivered in the form of services over the internet including the hardware and systems software situated in the data centers that perform those services [14]. The services offered are termed as Software as a Service (SaaS). The data-center with its hardware and software is called the Cloud. Fig. 1.4 shows the predicted investments to be made different aspects of the RFID technology [15].

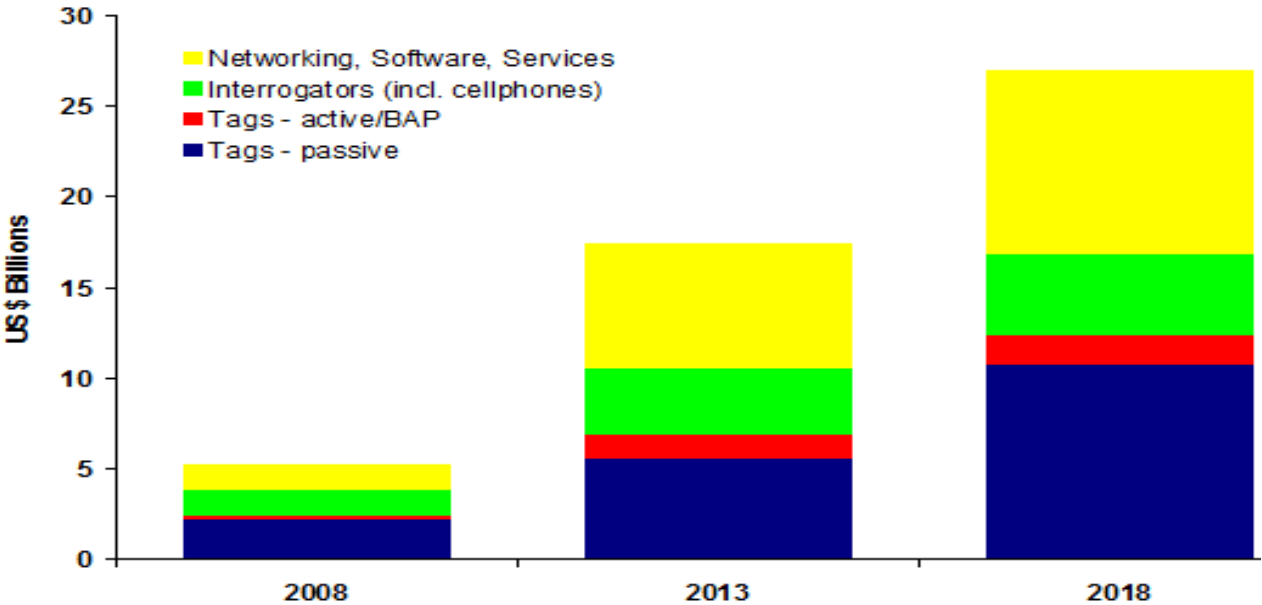


Fig. 1.4: Predicted Investments to be made in RFID in the coming years

Chapter 2

Motivation and Research Outline

The role of RFID in supply chain systems got a big thrust when many big players in the supply chain industry released guidelines and mandates to regulate the use of RFID. UCLA-WINMEC has been researching development of RFID based middleware such that the overhead investment made by a supply chain client interested in adopting the technology is minimal [16a]. Some of those organizations that pushed the adoption of RFID in an extensive way have been listed below:

- Supermarkets:- Walmart gave a big push to the RFID technology (June 2003) by mandating incorporation of RFID technology for pallet level tracking in supply chain [16].
- Defense:- Department of Defense (DoD) outlined a policy to adapt passive and active RFID technology in specific supply chain scenarios (In Transit Visibility, ITV) [17].
- Aerospace:- Boeing has laid out requirements to tag RFID tags for its various needs from the high value assembled spare parts to the tools used by the personnel. Different types of tags and frequencies are being simultaneously utilized to enable the multiple functionalities they want from an RFID system such as track the movement of high value spare parts. Airbus is also working with Boeing to be able to arrive to some standards suitable for the needs of the aerospace industry [18].
- E-pedigree:- The term e-pedigree is defined as an auditable electronic record taken by a retail package of prescription drugs as it moves from the factory to the final point of sale.

This chain of custody record helps to verify the authenticity and the integrity of the drug supply [19].

E-pedigree is meant to replace the current paper based chain of custody documentation which is error prone and subject to falsification. The State of Florida has enacted its own electronic pedigree law in the year 2006 and while it was a vast improvement over the paper based system, it showed two weaknesses:

1) Electronic data which duplicated the paper based records were not necessarily uniform across all suppliers

2) Data exchange rates in practice were found to be slow.

The State of California has asked all drug manufacturers to assign a unique serial number to each sealable unit within a manufacturing run. This serial number will act as the key item of data to record in each step of the chain of custody in the supply chain. The California drug pedigree program is to be implemented in beginning of 2009 and given the size and economy of the state of California, it will represent the pedigree default standards until the FDA comes up with its own rules.

The challenges faced by the introduction of the proposed system will be

- Creation of a serial numbering scheme which is suitable for the massive numbers of individual drug items sold by a company. These numbers should be unique for a time-span of several years.
- Development of ways to capture the serial number data at each step in the supply chain where physical and legal custody of the drugs change hands.

- Data handling modifications to the existing corporate IT infrastructure to handle the increased load of serialized transaction data.

The EPCglobal organization has developed the solutions for the challenges faced by the pharmaceutical industry mentioned above:

1. A scheme to provide for massive and unique item-level serialization in the form of a Global Trade Identification Number (GTIN) mapped into a 96-bit encodable EPC-ID number.
2. The establishment of a global standard for XML-based item transactions, the EPCIS (EPCglobal Information Services Standard).
3. The establishment of a specific pedigree standard to work within the EPC ID and EPCIS framework. This e-pedigree standard is the GS1 EPCglobal Electronic Pedigree Standard.

However the data capture technique for each transaction has been left as an open question. Although EPC widely uses UHF RFID standards, other methods include

- Standard linear barcodes
- 2-d barcodes
- HF RFID transactions
- Manually-entered data

These developments indicate a very rapid growth in the area of RFID related technologies. As RFID will be implemented in supply chains throughout the world, new issues regarding their improvement and streamlining will emerge

2.1 Current Issues

Currently, an organization which plans to use RFID needs to have supporting staff qualified in handling hardware, software issues, deployment and related aspects. Setting up an RFID infrastructure also needs extensive installations of enterprise management systems, middleware, databases etc which generally act as an obstacle for interested organizations. One can envision that in the future, an RFID service could be like a mobile phone service, where a client can simply buy the hardware as a plug and play device and instantaneously commence RFID operation. The web service can be similar to the switching station. The client would need to create an online account with the RFID service provider which automatically allocates server space, data processing and event notification services specific to the client on the cloud. Creation of such an account-based service would automatically enable client data partitioning/privacy and account customization capabilities just like most present day online account-based services such as email offer. A web-based architecture is also more suitable to RFID-based supply chains than localized middleware based architecture as most supply chain operations occur over multiple stages at different geographic locations and providing the client with a universal web-based interface would allow the client to exercise greater granular control and hence allow more flexibility to the operations. As an example a web based user interface enables the client to remotely monitor and control the reader operations regardless of the physical location of the reader. In this approach, we exploit the increased processing powers of the present generation industrial RFID readers (compared to the previous generation readers) and the database storage and event notification services provided by the cloud services to perform the functionalities which were performed by the localized middleware previously. The different services available within an RFID solution can be created and maintained by different stakeholders. As an example,

the initiation service of the readers could be provided by the manufacturers, data processing and filtration web service is offered by one stakeholder, database services are provided by another stakeholder and so on. This ensures that each component of the RFID solution is individually managed and optimized by different stakeholders who have expertise in that component. This allows us to inject federation into the architecture. The service provider can guarantee minimum quality of service benchmarks to the clients regarding data privacy, notification delay etc.

So the primary implementation aspects that need to be addressed are

- The present architectures involve extensive hardware and installation at the reader locations such as warehouses etc
- Installation of servers and other extensive hardware at the site makes it an expensive and cumbersome service The need to regularly schedule maintenance and checks on the proprietary hardware and software makes it difficult for the retailers, distributors and manufacturers to adopt it
- Can we have a simple plug and play system where we only connect the reader to the internet and the rest is taken care of by the service?

2.2 Overall Research Objective

A universal internet based remotely accessed and controlled RFID service which manages your RFID based supply chain (or other similar applications) by simply connecting your reader(s) to the web as shown in Fig. 2.1. The web service architecture would also provide enough flexibility to the clients to customize the service sufficiently to suit their business needs. This document will identify the major outstanding issues in this architectural transition such as raw data

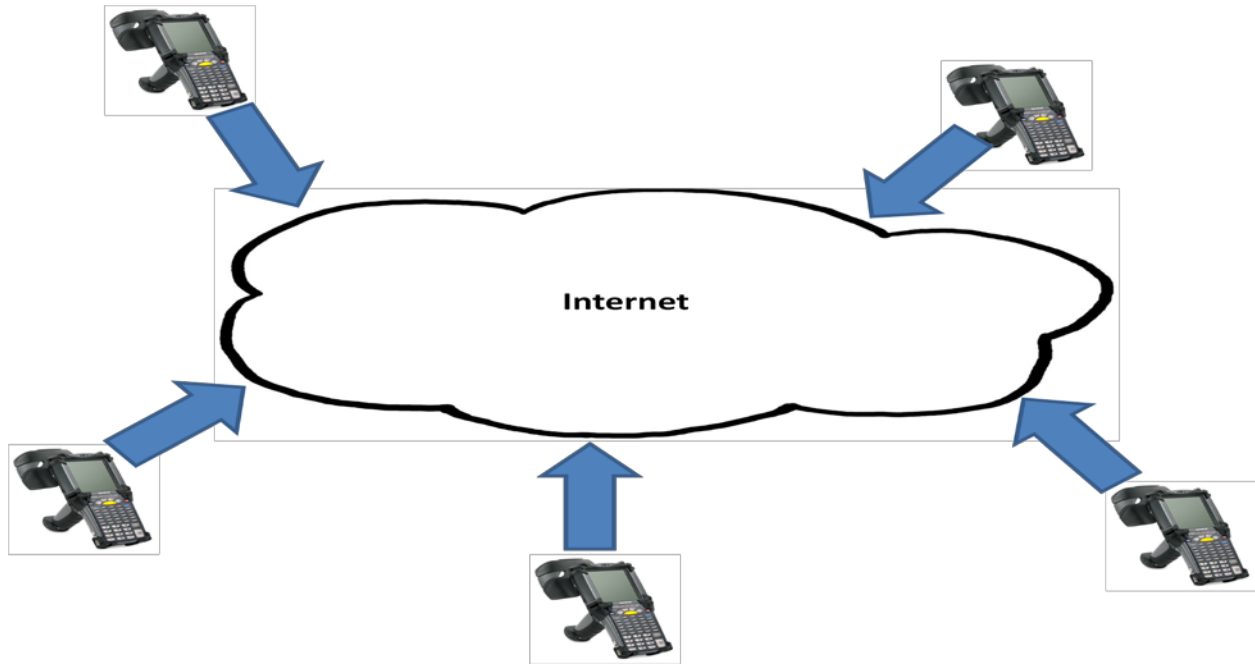


Fig. 2.1: Desired Web Service Architecture

compilation, transport and organization, remote reader control and event processing and address them. Further, the thesis would showcase some measured data obtained by performing stress tests on the web service and discuss its suitability to real world applications. The reader communicates with the web service primarily on a client-server model. The web service acts like a central hub which communicates with the readers, updates the database, performs event notification and routes the commands/settings issued by the client back to the reader. An example of a use case where this kind of architecture could be used would be a supply chain operation where manufactured goods are shipped from the manufacturer to the warehouse. If the manufacturing facility is large, the personnel can monitor all the readers by logging on to the web-based user interface to track the manufactured goods. Further, the personnel could be sitting on a centralized remote location monitoring the readers at multiple manufacturing facilities on personal/mobile devices as the web-based user interface removes the need to be physically

proximate to the readers at the manufacturing facility. From this remote location, the personnel can give commands to individual readers to interrogate certain tags or change the reader read range to look for more tags etc. Subsequently, when the goods are shipped to the warehouse, the personnel could create a trigger based events where the detection of some or all of the tags on the goods by the readers stationed at the warehouse automatically causes the warehouse gate to open and commence unloading operations.

2.3 Research Proposal

- Investigate services oriented architecture
- Attempt to implement the architecture
- Test in the context of Pharmaceuticals, warehouses, defense supply line etc
- Determine which services are implementable
- Implement and test the services

The intended web service will have the following application requirements:

I) RFID Data Dissemination: The information recorded by the reader needs to be sent to many different types of applications that might be interested in it. This information can be retrieved in two ways. One way is the query response model where the application queries the reader and receives a batch of data collected in the interval after the previous request and before the present request. Legacy applications that are not designed to handle streaming data might need to receive batched updates on a daily schedule. The other way is the asynchronous messaging. Here the applications are updated about a tag as soon as the information about the tag is received by the readers. As an example pick and pack applications are notified asynchronously whenever a new tag arrives. These applications require a short notification latency.

II) RFID Data aggregation: The large amount of tag data generated in an RFID based event needs to be aggregated. There are two types of aggregation:

(i) Type One is aggregating in time domain where the details of entry and exit events are stored.

(ii) Type Two is aggregating in the space domain, e.g. by combining data from different readers and different antennas that observe the same location. The RFID events are aggregated and the original reader ID is replaced by a logical ID which represents the dock door location.

Some applications are also interested in finding the number of items of a particular type arrived rather than their individual tag IDs so there is also a need for aggregate counting.

III) RFID Data filtering: This is a common RFID feature where the applications only want to receive filtered RFID events rather than all the RFID data captured. Different applications need different subsets of the total data pool based on the readers, reader antennas and the tags involved. For example an application might only be interested in tags attached to certain products.

IV) Writing to a Tag: Many tags these days have memory not only for the tag ID but also additional data. Hence the applications should have the flexibility and capability to read or write this additional data. This additional memory can be used to store additional inventory data such as expiry dates etc. They are also useful in situations where there is no network access and additional data about the tagged item cannot be retrieved.

V) Trigger RFID Readers by External Sensors: In many applications, one does not need to operate readers continuously as they consume bandwidth. In order to enable or disable their operation, external sensors such as motion sensors can be attached to them so that, they

automatically switch on when items enter their read range. This can be employed in areas such as dock doors etc.

VI) Fault and Configuration Management: RFID readers represent nodes in an RFID network that need to be monitored and configured periodically for fault and configuration management. While on-site maintenance is often performed, remote maintenance requires fewer resources and is potentially more scalable. Performing this function remotely also enables the integration of RFID into IT service management like other computing hardware.

VII) RFID Data Interpretation: RFID data received by the reader from the tag is usually low level data such as detection of a particular tag ID. This has to be converted into event based information. For example, the detection of a certain number of tags by a reader at the dock door will be translated into a “shipment complete” event. Previously RFID database architecture was designed for ID based data, but in the future the size of data transmitted by RFID tags will increase. Is the present database architecture suitable for this future scenario?

VIII) Lookup and Directory Service: Since an RFID tag passes through the readers of many different parts of a supply chain, its events and other related data are stored by these parts in their own information systems. In order to look up these various databases containing data on a particular item, a lookup service is needed. Present database architectures call for a local level database to enable the warehouse manager to fix local problems like false negative reads. Can a (local) server-less architecture take care of this problem, if not then what is the loss in information and accuracy in the system?

IX) Tag Identifier Management: The tag is usually identified by the unique identifier stored in its memory. Some prominent types of tag identification schemes are the EPC code, ISO code etc.

The EPC code is also available in different representations which are suitable for storage in tag memory as well as representations in uniform resource locators etc. This calls for the need of a tag identifier translation mechanism to convert between the different mechanisms.

X) Privacy: The ever increasing presence of RFID in many applications coupled with its small size has triggered a debate on its implications on human privacy. Security protocols are being strengthened to address these issues (for e.g. kill command for tags etc). Hence, support for such features also needed while designing a middleware.

XI) Cloud Service: Cloud based computing has been frequently cited as a favorable replacement for many types of software systems where the computation was previously localized. In this work, we are trying to create an RFID solution for supply chain systems with a web-service based architecture which pushes a lot of the traditional middleware based data processing to the more powerful readers and to the cloud infrastructure. The clients can subscribe to event based data notification service where the raw tag level data is translated into an event notification based on client preferences by the service provider and notified immediately to the client without the client having to worry about the specifics of data processing and assimilation. The goal of this paper is to migrate from a localized middleware based architecture specifically customized to an individual client to a more generic web based architecture which allows multiple clients to subscribe and use with lesser infrastructural obstacles.

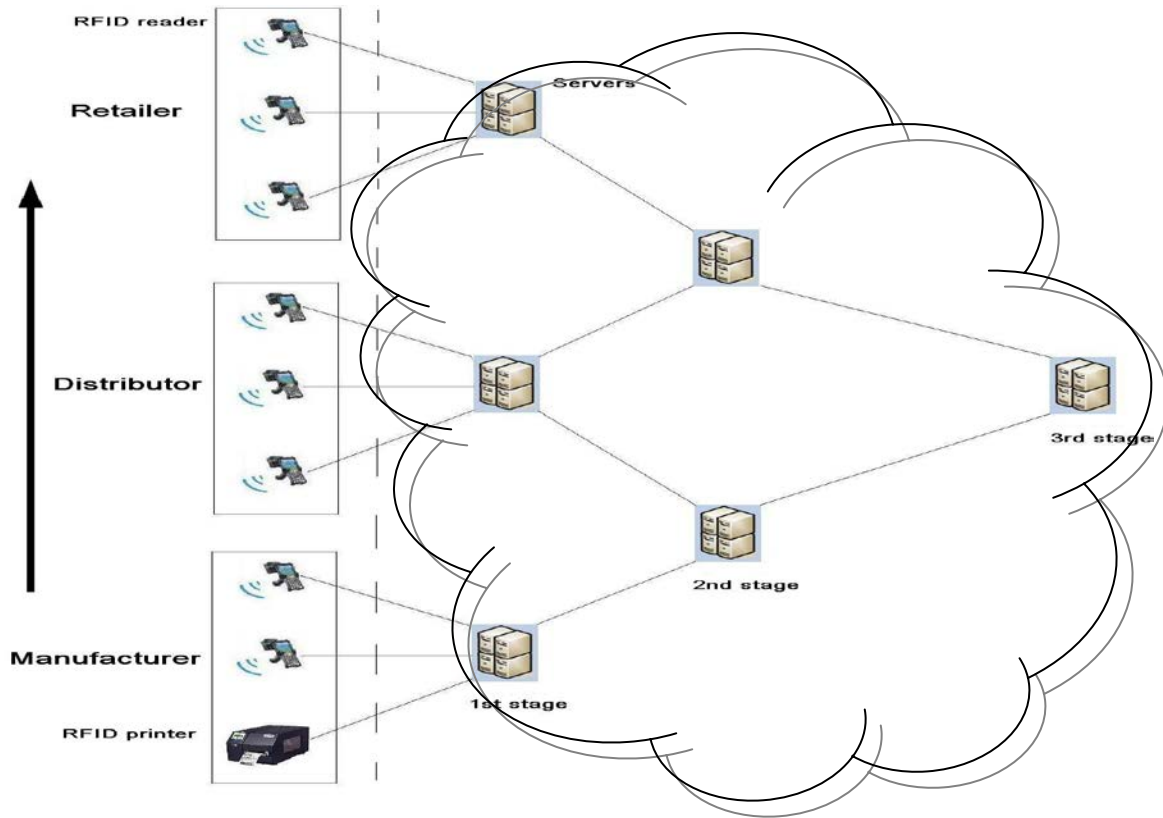


Fig. 2.2: Proposed Web Service System Architecture

2.4 Stages of RFID tag transport

In this section, the different stages of the supply chain are explained to signify how the RFID tracking solution automates the process of product distribution. A supply chain has been broken into three basic parts:- Manufacturer, Distributor and Retailer as shown in Fig. 2.2. An RFID tag is created by the RFID printer and attached to an item to be shipped. This data is sent to the first stage server. Then it is checked by an RFID reader and shipped to the Distributor. The Distributor receives the shipment and checks the tag using its RFID readers and verifies the details by the first stage server. Finally the Distributor forwards the shipment to the Retailer who again repeats the same process of verifying the tag details, its authenticity etc. The function of the first stage servers is to filter the data received by the RFID readers, to arrange them according to their origins and forward to the second stage. These servers receive requests from thousands of

readers in a given geographical location. This server needs to be scalable because of the potential for a flood of RFID data being transported through it. Second stage servers maintain the entire databases for organizations. The first stage servers refer to their database to reply to the tag enquiries by the readers. These databases are also used by the third stage for translation into business events. The third stage servers translate RFID based events into business events, for example, the arrival of tag A is translated into the arrival of item X to a particular destination. This information is sent to a subscriber of the service in real time who had requested this information. The third stage is also used to monitor and define the business process workflow by the authorized users.

2.5 Test-Case Study

Let us imagine a hypothetical situation that a drug company wants to ship 50,000 units of medicines to RiteAid. Every unit is attached with a 96 bit EPC tag. The entire shipment is scanned by an RFID reader mounted at the gate of the warehouse as the truck carrying the entire shipment enters the distributor/retailer warehouse. The truck takes approximately 3 seconds to cross the reader's field of view. Hence the reader reads and transmits $50,000 \times 96$ bits in 3 seconds (assuming ideally 0 data latency) which is approximately 196 Kbytes per second. If there are 1000 such readers in a given geographical area under the operation of a single server, then its maximum data handling capacity should be approximately 191 Mbytes per second.

2.6 Proof-of-Concept

First we try to create a basic setup consisting of three stages (i.e. Manufacturer, Distributor and Retailer) (Fig. 2.3). One server is setup and three readers are placed one each at the three stages. The readers are connected to the server to send and receive data. A printer creates a tag. This tag

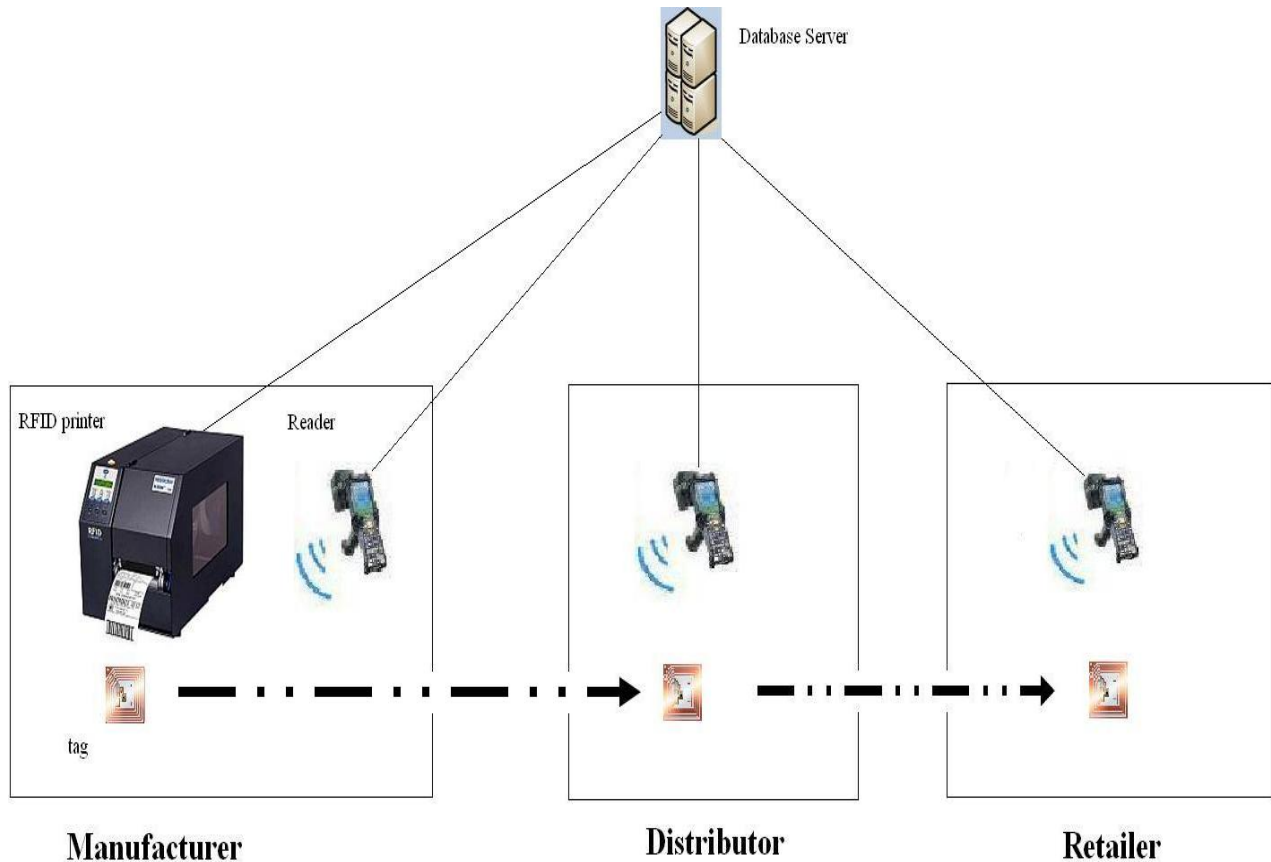


Fig. 2.3: Initial Setup for proof of concept

is read by the reader at the Manufacturer which registers the tag at a database in the server. Then the tag moves to the next stage (Distributor) imitating a supply chain. There the reader detects it and compares it with the server database. Finally the tag moves to the last stage (Retailer) where a reader again detects it and compares it with the database at the server. This process will be repeated with higher number of tags to study the choke points in the system (Fig. 2.4). The system will be tested by loading it with many more readers now simultaneously contacting the server. This will allow us to analyze a system at peak loads and make modifications to improve efficiency. Later multiple supply chains would be simulated to check the performance of the servers while handling multiple simultaneous requests from different supply chains.

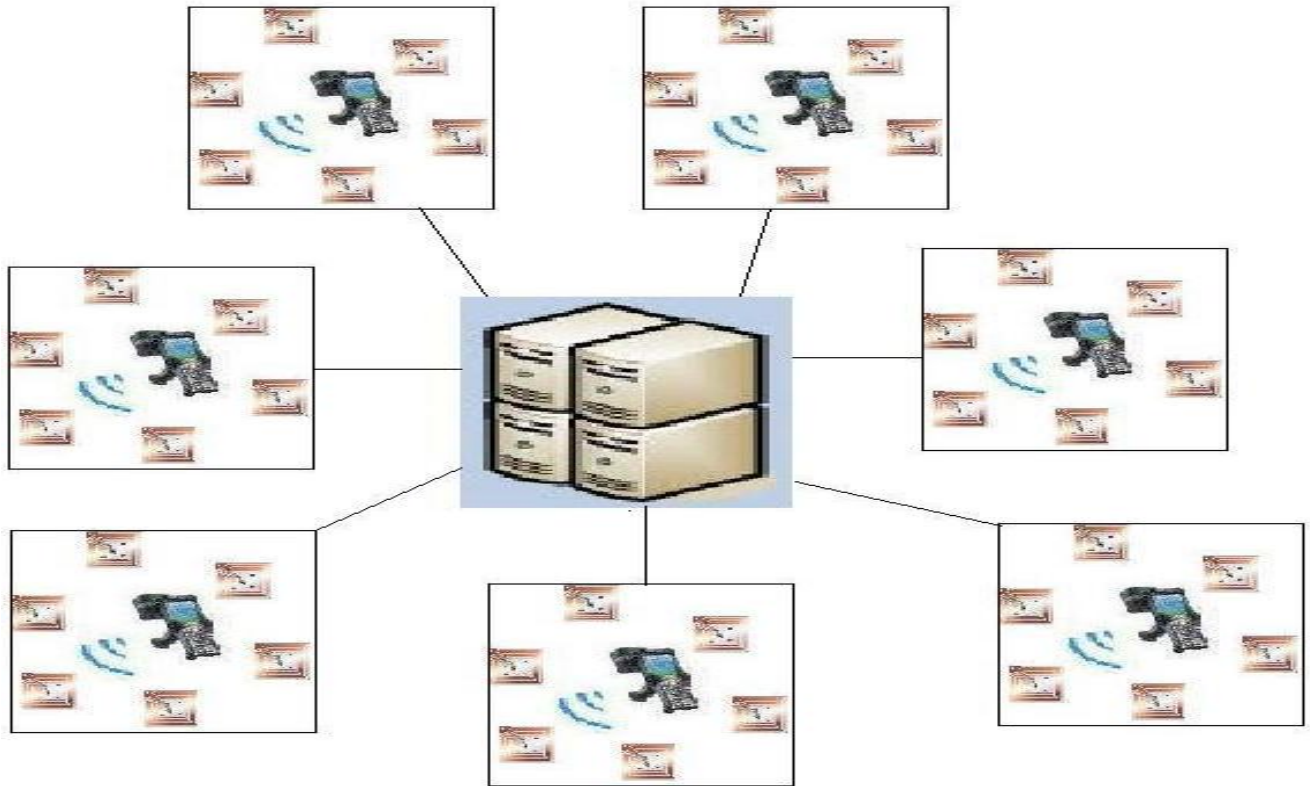


Fig. 2.4: Testing scalability of the proposed web service architecture

However to simulate a large number of readers, we use what is called a ‘virtual reader’. It was first implemented by our lab [20]. It is a software code which will generate large amounts of data at specified rates in the form of Tag IDs. We aim to improve it by making it capable of generating some erroneous Tag IDs at certain intervals to truly emulate a real reader. A desired amount of readers can be simulated in this way by running this code.

Chapter 3

LITERATURE REVIEW

Numerous studies have been done in analyzing and defining architectures for RFID based systems for various industries. These have covered various aspects of RFID architecture such as data filtering, database design, RFID event processing etc. Here, we look at some works which were relevant or inspired our research.

3.1 Supply Chain based aspects addressed by the EPC

This section explains how EPCglobal has delivered to the concerns of the supply chain operators [21]

Shipping and Receiving:- In today's supply chains sending and receiving products involves many complicated and time-consuming processes. The shipments often arrive with the wrong number and wrong type of products. This therefore requires an additional step of a thorough re-check of the products. This wastes a lot of time in investigating exceptions such as actual product loss or inventory accuracy. These issues lead to higher distribution costs and hence higher prices.

Product Theft:- Almost \$ 30 billion is lost each year due to theft, which is also known as "product shrinkage". Most of this loss takes place in the middle of the supply chain between the manufacturer's front door and the retailer's back door. As there is limited visibility in the supply chain, it is difficult to track down these losses.

Since the EPCGlobal system records the product entry and exit at each point, there is more visibility in the product's movements. So if any product goes missing, the record of the chain of

custody of that product can be checked to find out the point at which it went missing and corrective measures can be taken to rectify such occurrences.

Counterfeiting:- Product counterfeiting has become a serious problem throughout the world and the United States too. Drug counterfeiting has forced the US Food and Drugs Administration (FDA) to form the Counterfeit Drug Task Force. Ensuring the authenticity of drugs can be a difficult challenge due to the challenging nature of the modern supply chain. The EPC Global addresses this challenge by automatically scanning the drug shipments and authenticating them at each stage as they move from the manufacturer to the pharmacy. This electronic pedigree (e-pedigree) can also be used to determine the authenticity of the shipment by storing its pedigree and making it accessible to all parties.

Product Recall:- Product Recalls are also a significant source of loss for an organization. If a defect is found, the company needs to recall all the batches of the same product as it is not sure in which batch the defect occurred. This is extremely harmful to the profits of the organization as it leads to destruction of perfect goods (throwing/dismantling the good with the bad). With the introduction of RFID based inventory management, the manufacturers can exactly pinpoint the source of the defective goods which allows them to recall the goods only from that source, thus saving lot of cost and time.

3.2 Database Research in RFID

As discussed previously, many large enterprises such as WalMart, Department of Defense and Proctor & Gamble/Gillette have defined their RFID requirements [22]. The core data in RFID implementations is the unique identification of a product. This is usually accompanied by semantics, or metadata or data about data, describing where that product is in a business process

or its physical location. The core data typically never changes but the metadata changes frequently. Other characteristics that make RFID data management different are:

- 1) RFID reading is imperfect and it is likely to stay so. Thus when stacked on a pallet, a tag is readable from some angles and not from others. A sudden inability to detect the tags may mean that it has been lost, or that it cannot be read. Sophisticated filtering software can reduce the amount of these false negatives, but data management at the local level must monitor, alert, and handle false negatives in order to aid the manager of the warehouse, while cleaning the data as far as possible so that it can be used by enterprise data miners.
- 2) In an RFID based supply chain, each tag is constantly generating a new data as it moves through the production and distribution process. Even with the sophisticated filtering software, there may be thousands or tens of thousands of updates of RFID data (i.e., changes to where tens of thousands of products are in a business process or where they are physically) per minute at peak load. If we add the requirement that we track where each product has been, say, every half hour (a process trail), then we also generate an enormous amount of static data
- 3) Walmart and DoD have ruled that each supplier not only tag its products but also pass the RFID data for a product from supplier to customer, all the way down the supply chain. Thus, Walmart can query its supplier for the status of an item, or Dell can track a shipment from customer order to delivery and beyond. Every RFID data store is potentially accessible from outside the organization and every RFID data management system potentially queries across organizational boundaries.

Because of these reasons databases must be re-tuned or re-architected to handle the new needs of RFID transactions as it happened with data warehousing, content management and web data in recent past. According to Kernochan, real world RFID based database architecture implementations are tending towards a three level overall architecture: At the lowest level a buffer receives a data from RFID readers and printers, filters it for duplicates, cleans it if possible to eliminate false negatives, and adds semantics to the data for use at the next level up. As of yet, no database is attached to this level, but an embedded database can be used. The middle level is the local level. This typically involves one physical location such as a warehouse, and the aim here is to allow the manager (for e.g. warehouse manager) at the location, some ability to monitor, fix problems and do analysis. Often the local level allows the user to define workflow within the physical location. A local database supports these tasks. The database may also add semantics to the core data, both for local workflow and to support global analysis at the top level. The top level is the organization or enterprise level. At this level, the user may monitor across locations (and across organizations), define workflow across the entire business process, and perform business analysis on the combined data from multiple local databases, either separate from or combined with other enterprise data. Also at the organizational level, a user may reach across not only to access another organization's enterprise RFID database, but also to probe its local-level databases. Other RFID related database research is listed below:

- Chalasani et al. proposed RFID data models based on transactions generated by the participants of a supply chain which are the manufacturer, distributor and the retailer [23]. They have many similarities to the work presented in this thesis. Their model emphasizes on a decentralized architecture as it leads to lower costs and architecture

extension. The cloud-based approach demonstrated in this paper also highlights the benefits of federating the traditional localized middleware based RFID architecture. They discussed storage requirements of the RFID transactional data and strategies to minimize the storage requirements. They introduce the concept of reader ID. They use the Internet protocol (IP) address of the reader for this ID which could create problems if the readers are in a local area network (LAN) as two readers from different LANs could have the same ID. We have addressed this issue in our work. They discuss common supply chain issues such as detecting missing items on shelf. Such kinds of event detection are possible in our architecture using event creation. They have augmented their architecture with shelf replenishment costs and duration predictions. However, very little information has been given on real-life implementation.

- Mylyy stated the need to digest the data close to the source for filtering and preventing the flooding of higher layers of the system by low-level data which is an important component of our design too [24].
- Chawathe et al. have discussed in good detail, the issues regarding data management and warehousing in an RFID system [25]. Their discussion previews the various strengths and limitations of using RFID systems and thereby gives cues on an efficient middleware design. They have demonstrated that reader identification, tag identification and identification time-stamp are the three primary variables for numerous types of event-based business applications.
- Gonzalez et al. proposed the RFID cuboid for storing aggregated data based on the premise that items usually move in larger groups at initial stages of a distribution

system and that the RFID data generated at the lower levels is only useful and desirable at higher abstraction levels and so can be compressed and grouped at lower levels [26].

- Table 3.1 highlights the offerings of the present RFID infrastructure software suppliers in accordance with the discussion done above [22]:

Lots of research is being conducted in developing RFID based middleware which address issues such as tag data collection by readers, defining rules for business events etc [20], [27]. Some organizations (Skyetek, imobile systems) claim to have a complete web based RFID inventory

Supplier	Buffer level	Local level	Business level	Connectivity
Checkpoint Systems	No database	In-memory database (handles moderate amounts of data)	Supports Oracle with Oracle OLAP Option or IBM, provides own database and analytics	Third-party partners, adapters
ConnecTerra	No database	User chooses database and implements transactional code	User chooses database and implements transactional and replication code	Third-party partners
GlobeRanger	No database	User chooses database and implements transactional code	User chooses database and implements transactional and replication code	Third-party partners

Table 3.1: RFID infrastructure-software vendor data architectures

management system [28], [29] providing web services with proprietary hardware and software support to enable manufacturers to track their inventory in real time. IBM based WebSphere RFID Information Center [30] promises scalable, secure, standards based repository for RFID data capture and event query. Fig. 3.1 shows the architecture of the Skyetek RFID deployment [28]. Other organizations such as At&t [31] are coming together with RFID hardware manufacturers like Alien to offer RFID based supply chain services. The focus in these specialized middlewares is on translating tag data capture into business event. No information given on hardware compatibility (ie compatibility with RFID readers from various manufacturers etc) platform independent? Only specific hardware (tags and readers) and Local Servers need to be installed at the site.

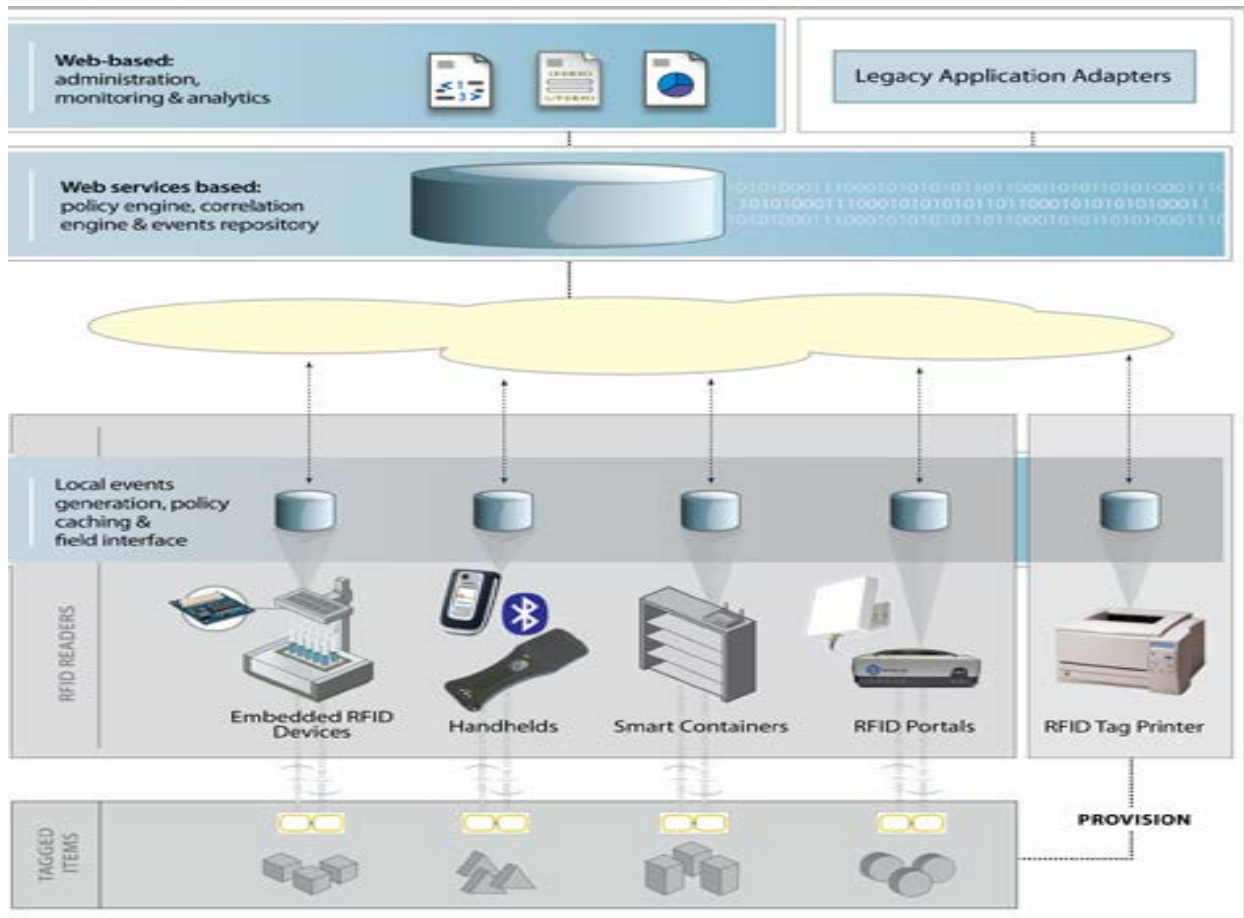


Fig. 3.1: RFID Proprietary Web Service Architecture

3.3 Architecture Research in RFID

In this section, we discuss the different kinds of middleware architecture designs that have been recommended by different research groups and how they are relevant to the work done in this thesis and what are their shortcomings that open a window of opportunity to address in this work.

- Bade et al. have proposed a three layered RFID middleware architecture in which event processing is performed by software agents [32]. These software agents allow frequent architectural changes in dynamic and distributed environments. However, their work is only descriptive in nature with no quantitative data or real life test cases to support the effectiveness of their architecture or use as a reference for comparison.
- Ngai et al. have also proposed a web-based RFID-prototype system for a container depot [33]. They discussed short-messaging service (SMS) connectivity as a back-up to deal with unreliable web connectivity. Unreliable web connectivity is an open issue for our architecture too and has been mitigated. However, their paper also mostly discusses qualitative issues with little quantitative data.
- Jeffery et al. propose an RFID middleware architecture whose novelty is a layer of metaphysical data independence [34]. One of the supporting arguments for the metaphysical layer is to minimize the access of client applications to the devices while maximizing the access to past and present data to prevent the unreliability and mismatch of date with application needs in time and space.
- Schwieren et al. discuss a three level architecture where each level is customizable and higher level layers are optional depending on the requirements of the application designer

[35]. The applications primarily involve events occurring on detection of certain tags like information screens opening when tags of certain museum showpieces are detected which can be termed as primitive event processing. Function triggering by tag detection has been made robust keeping in mind both the availability/non-availability of internet connection to mobile device. However, it is more well suited for low volume activities where interaction, with nearby tag triggers certain events, unlike the supply chain where hundreds or thousands of tags would be simultaneously read by longer range readers, so large scale tag read and processing issues have not been discussed.

- Guinard et al. have discussed the fusion of cloud infrastructure with EPC based RFID infrastructure as a fix to some common problems that are faced by users of RFID based supply chain solutions [36]. The first issue that they have identified that adds to an inertia to adoption of RFID systems in to supply chain systems is the added infrastructure such as installation of servers, databases etc which they have addressed using the adoption of cloud based infrastructure as solution to this issue. The second issue they addressed was the barrier of software development for RFID operations which can now be eliminated using Representational State Transfer (REST) Web services given their developer friendly characteristics, thereby reducing development efforts. Further, they proposed web based adapters which follow a push-pull mechanism to allow RFID readers to push raw data read by them into web based user interfaces. Here, our architecture differs as we first push the data from the RFID reader to a distributed storage server via a web server. This stored data then appears on the web based interface. The delay time involved with a single such process is less than 1 second as will be discussed later and will be unnoticeable in many applications. Further, having recent data stored in a database also

allows the user to access the history of the object in real time for multiple possibilities and also evaluate complex events which can be true based on the occurrence of some recent events, which necessitates the knowledge of recent data. Finally, the third contribution claimed by them is the use of the Web Mashup technology for its ease of creating cross platform "mashup modules" which accept data from multiple platforms and deliver an output as a combination of those inputs as defined by the user. The authors also advocated extension of the previously created EPC standards for RFID based web services which we have also observed in our implementation. Overall, the authors have outlined some common issues and demonstrated a proof of concept of web service implementation, however, we have attempted a more detailed and complete implementation of an RFID web service hosted on a cloud and also characterized its performance.

- Zappia et al. proposed a Complex Event Processing Engine based on a layered architecture where they have claimed to separate the event processing logic from low-level thread management [37]. Their design philosophy advocates reduction of dependency on external software components and a lightweight implementation. They form a standard SQL like event defining language which is parsed by the event processing engine.
- Wang et al. proposed the algorithm for complex event processing from primitive RFID events [38]. Primitive events are typically detection of tags, whilst complex events are the various logical and temporal combinations of primitive events to achieve more meaningful activity notifications. However, non-causal events such as notification based

on non-activity of certain prescribed events are more complicated and difficult and call for creation of pseudoevents etc.

- Wang et al. demonstrated the Siemens RFID middleware [39]. They showed that the four fundamental entities for an RFID middleware for most applications would include the object of interest which is tagged, the sensor or the reader, the location of the object event and the transaction involved.
- Bai et al. had demonstrated a sliding window approach to solve the problem where tag reads were considered legitimate (and not false readings) only after they repeatedly occurred above a minimum threshold [40]. This however, leads to the loss of order of tag reads and further modifications were proposed which lead to increase in complexity and detection latency issues.
- A six-layered framework for an RFID based web service has been proposed by Sundaram et al. [41]. The web service communicates individually with RFID tags that pass through the coupling field of readers which are controlled by the web service and the latter shares this data with other systems that require the data. Their framework addresses the problem of integrating heterogeneous and loosely coupled components to form a service chain. The unique aspect of their approach is the ability to integrate heterogeneous and loosely coupled components to form a meaningful value chain. In their conceptual approach, they have modeled the RFID data stream as an object which can be further linked up and joined with a choice of services like joining blocks. Our web based event creation interface also allows a user to type the URL of a service and set its subscription to receive messages on successful completion of an event. The messages could be either the tag data

stream or a user pre-defined message. The choice to send tag data or message is important as in an inter-organizational set-up, organizations might prefer to not send raw data streams to other organizations.

- Welbourne et al. have showcased a web based RFID tracking solution primarily for personnel in an indoor environment [42]. The main theme of their work is to attach RFID tags to certain objects which would be read by a network of reader antennas and their movements and location patterns would be recorded. These patterns can be used to record and observe higher level events such as the detection of a certain tag by a certain antenna signifies a certain type of high level event and gets logged in a table. This work has many similarities to our work such as a complete web based access, tag metadata management by the individual user and tag data translation to higher level events. However, their work was primarily used for personnel tracking and the low level tag movement to high level event translation was passive where the event was only stored and could be viewed by the user. In an industrial supply chain based environment the low level tag movement to high level event translation should be active, i.e., it should be capable of triggering another action such as informing a supervisor or starting another operation (detection of tags leading to automatic opening of gate doors) etc. Our solution has addressed these situations. Also another area where our solution stands out is web based control of readers. Welbourne et al. had installed 44 readers in a huge building with proprietary software installations and in house servers with complete connections. We have demonstrated that in such scenarios, one can greatly reduce the setup costs and infrastructural requirements by using cloud computing based resources instead of in house servers and web services to communicate with readers instead of localized

middlewares. Also in an industrial scenario allowing reader level control to an authorized remote user greatly improves the scope of flexibility of a system.

- Siorpaes et al. defined a universal client design and architecture which acts as a mediator by linking physical objects and the relevant services through a mobile device [43]. However, this was limited to simple mobile device data exchange. Applications such as numerous tag reads in a typical RFID based supply chain scenario were not considered.
- Broll et al. [44] demonstrate an application to run web services on mobile devices which are more ubiquitous in nature. They enlist certain mobile based NFC (Near Field Communications) applications; however, their requirements are different from the typical supply chain application of an RFID based system.
- Romer et al. have created two frameworks based on Jini (i.e., distributed Java objects) and web services to support the development of ubiquitous computing applications that make use of smart identification technology such as RFID [45]. These are based on the set of basic functions and services, and application model for RFID type of services.
- Su et al. have presented a layered architecture for an edge server which coordinates amongst various types of identification devices such as RFID reader, barcode scanner etc [46]. Features of their middleware include hardware abstraction and rule based device configuration.
- Ahmed et al. have discussed the concept of representing a group of readers using virtual abstractions/ representations which combine, filter and present the information provided by lower level readers to the hierarchy [47]. There is also the concept of virtual path where paths are autonomously created amongst the virtual abstraction for a supply chain

system which assists in informing supply paths, investigating missing tags etc. The author claims that such a combinatorial approach improves or adds upon the otherwise individual and error prone performance of a reader. The event queries are passed on from one virtual abstraction to the neighboring; however, the author has not explained how this scheme is better than searching an event plainly from a database formed by aggregating all event information.

- Lopez et al. have proposed the idea of an EPC based web service for a combined RFID and wireless sensor network (WSN) infrastructure [48]. The architecture proposed is truly ubiquitous in nature and aims to serve a large variety of tasks which can be served by sensor nodes and RFID tags. The architecture has been well explained using graphs and charts; however it involves an intricate network of databases and middlewares. Also not much information has been given regarding the implementation aspects.
- Liu et al. proposed a data framework for numerous RFID applications [49]. Their work had a flavor of Supply Chain applications however the focus was mostly on data modeling.

Chapter 4

SYSTEM ARCHITECTURE

The graphical representation of the system architecture of the RFID web service has been shown in Fig. 4.1. The system architecture has four main components. The four components are 1) Reader resident application (RRA), 2) Cloud based Reader Web Service (RWS), 3) Cloud based database service and a subscription based notification service and the 4) Cloud based Web based User Interface (UI). As shown in Fig. 4.1, when the reader scans tags, the RRA (resident in the reader) collects and filters RFID tag events and transmits it to the RWS by means of a SOAP message. The RWS updates the data (such as timestamps, event occurrences etc) in the tables corresponding to that reader in the cloud database which is the third component of the system architecture and checks if there is any remote command stored for the reader. Remote commands are commands issued to an RFID reader remotely by a user using the Web based UI.

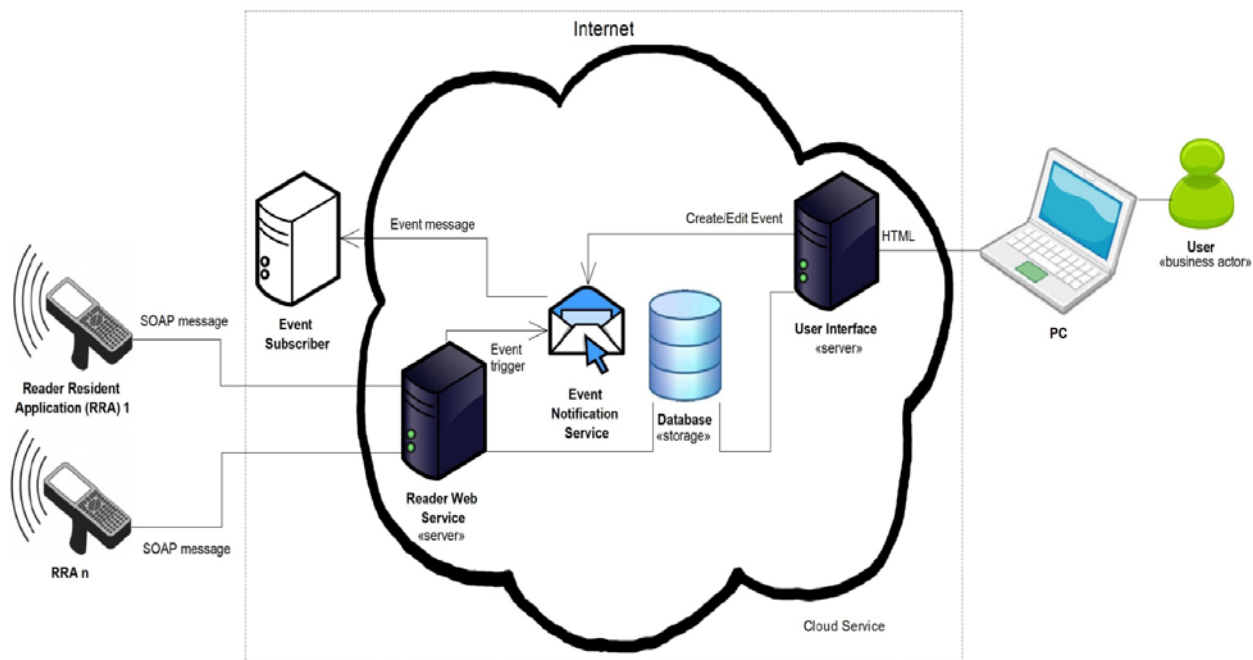


Fig. 4.1: Symbolic system architecture of the proposed RFID

Examples of such a command could be ‘Stop read’ or ‘Increment read range’ etc. If a command exists, then the RWS transmits the command back to the reader in the acknowledgment message. The UI which displays the operational data (such as tags read and their read times, readers in operation, events etc) from the cloud database to the client, allows the client to modify this data and collects remote commands for the reader from the client. The following sub-sections give a brief description of the sub components of the architecture. However, a more detailed description of each component has been discussed in the previous paper written on this work [50]. The detailed functioning of each component would be discussed in detail in the sections below.

4.1 Reader Resident Application (RRA)

Reader Resident Application (RRA) is a standalone software module that facilitates communication with the RWS. Usually, an RFID reader simply forwards the raw tag data to the next upper layer for data filtering; however, using the RRA, we explore the possibility of reassigning some of the tasks of the upper layer to the reader itself by exploiting the RFID reader’s processing capabilities. This type of application can be downloaded for any reader which can connect and browse over the internet.

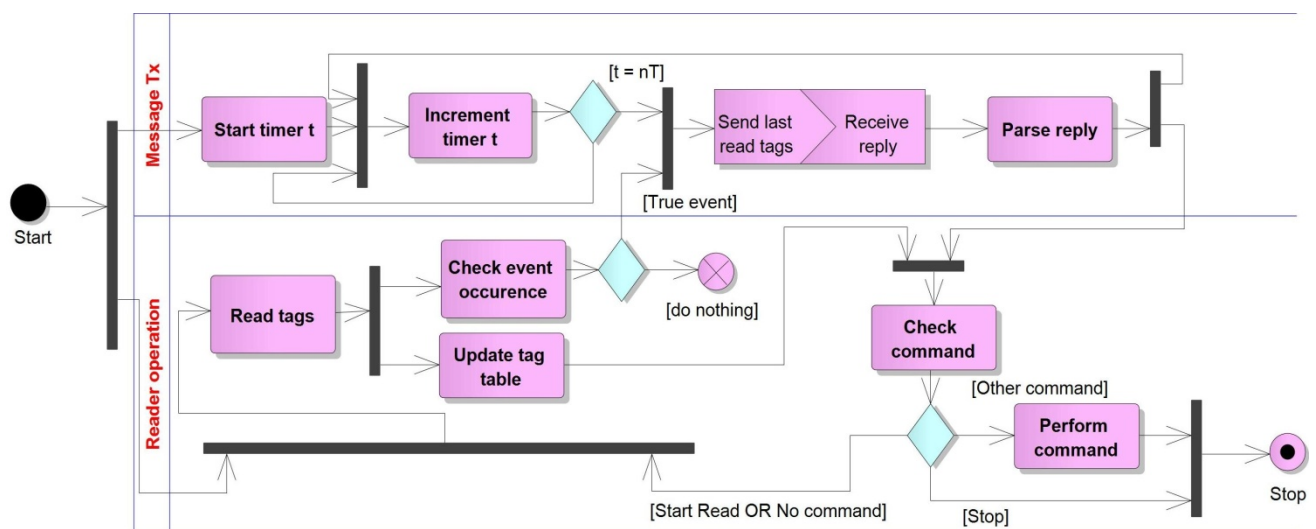


Fig. 4.2: Reader Resident Application workflow

There are many readers which require the installation of middleware on a host computer, after which, the device can be accessed. In such situations, the RRA stands out as it enables control and communication of such readers through a web based interface on a physically unconnected remote machine without installing any application on the machine. There are many OS based industrial readers by other vendors which have similar designs and capabilities and so the application can be offered for readers from different manufacturers. The possibility of having readers from different manufacturers run the same application will give clients more flexibility in using the solution which is a big impediment with most of the present day middlewares, as they work with only specific readers. Computational/networking capabilities such as serial port based readers can also be made to behave in a similar manner by running a version of the RRA on the host computer controlling the reader. In that case, it would function as a generic middleware on the reader host machines, where it would command such readers to perform the same actions (otherwise performed by the RRA inside the reader). Additional data processing and transmission to the web service would be performed by the host computer. As implementing the RRA (which is capable of being run by the RFID reader's mobile processors) on a host machine is not significant from a research standpoint, we have not implemented this part. Fig. 4.2 shows the workflow of the RRA. Fig. 4.3 shows the graphical user interface (GUI) for the RRA showing the reader's data table.

4.1.1) Role of RRA: The tasks of the RRA are to collect, compile and transmit tag information to the RWS and to execute commands received from it. One common requirement for collecting raw reader delivered tag data is the data filtration and compilation. The raw tag IDs collected by most readers contain many duplicate reads. This can be seen in the tag window of the RRA in Fig. 4.3 where one tag is read multiple times. Before the tag information is transmitted to the

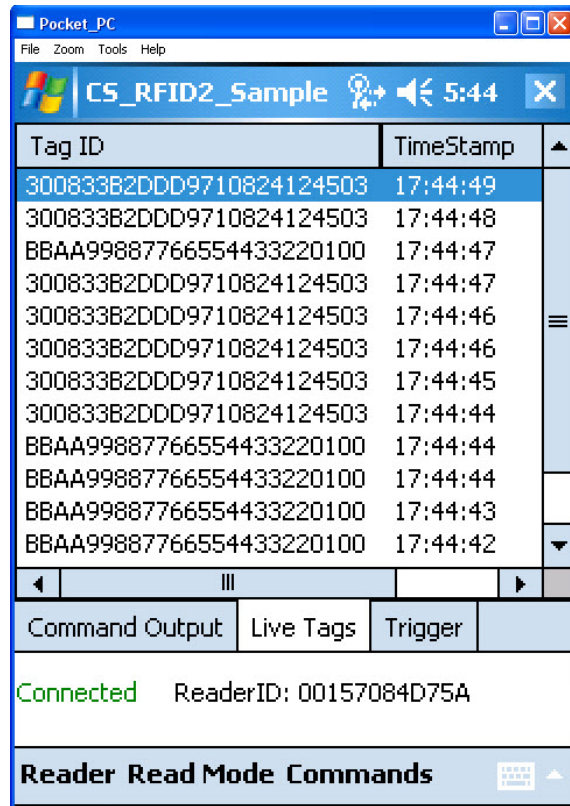


Fig. 4.3: Reader Resident Application (RRA)

next higher layer, it needs to be compiled and filtered to avoid redundant data transmission (e.g., duplicates elimination). Traditionally, this is done by the middleware; however, this functionality is tested in the RRA itself. In order for the tags to be read and compiled by the RRA, they need to be read for a certain period of time (buffering) before they can be compiled. This will require a buffering period before the RRA compiles and transmits the message containing all the previously read tags to the RWS which would be called the reader report interval. Ideally, the reader report interval should be minimal to prevent data transmission delay and preserve data fidelity. However, if the buffering period is too small, in an industrial environment, the arrival of shipments of large quantities of tagged items would cause a surge of new tag events at a very high rate. If each new tag detected by multiple readers is sent individually

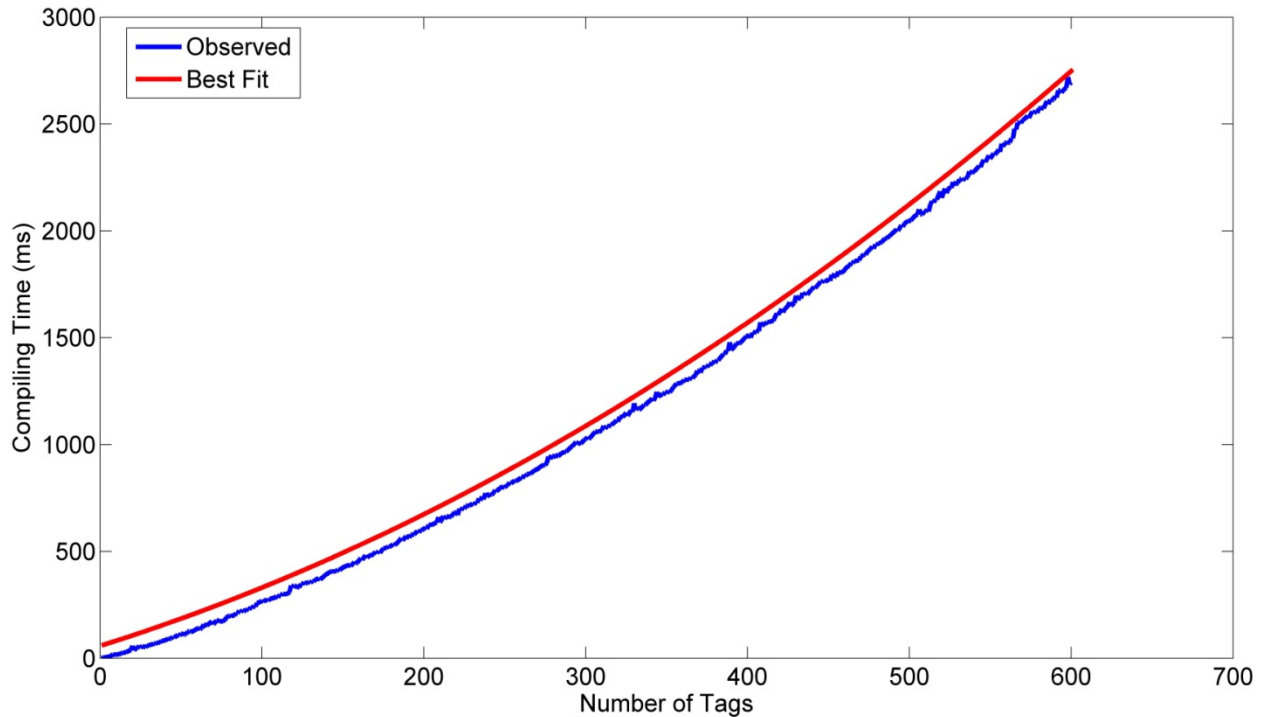


Fig. 4.4: Message Compilation time by RRA as a function of number of tags

to the RWS at a high rate, there is a high likelihood of the server getting overwhelmed and numerous messages being lost or being missed. This was explained in greater detail by Chawathe et al. [21]. So sending the entire situational information as a single consolidated message increases the chances of data fidelity and occupies lesser bandwidth as the tag detection events are aggregated and hence sent less frequently than individual tag data packets. The compromise in this technique would be a finite latency in transmitting new tag events. However, for most practical supply chain operations, a finite latency would be tolerated. Also the RRA would have its own processing delay in compiling the messages. In order to test the RRA processing delay, a simulated run was performed on the RRA residing in the RFID reader to measure the processing delay in compiling a given number of tags into a single message. In this experiment, the number of tags was to be varied from 1 to a high number (600). However, performing this test in flowchart a lab setup is challenging due to various physical factors such as nearby objects,

distance from the tag, orientation of the reader etc which make it hard to precisely control the reader to consistently read a desired number of tags. In order to overcome this challenge, the RFID driver of the reader was disabled and replaced with a simulated RFID driver which provided the specified number of simulated tags to the RRA. Then, a test was carried out with the simulated driver simulating the specified number of tags for the RRA to compile. Fig. 4.4 shows the graph of the average processing time to compile a given number of tags versus the number of tags. The average processing delay values were obtained after taking a mean of 4 similar runs. The processing delay for compiling a message consisting of one newly read tag was 0.75 milliseconds (shown by the blue curve); while the processing delay to process 600 tags was 2695 milliseconds. The best upper-bound curve for the RRA performance has been shown in the red curve. This curve was obtained by using a Matlab based curve fitting tool. The second order norm for the two curves was 1671.6 seconds. This upper-bound curve equation is used as a safe limit by the RRA to autonomously determine the optimal reader reporting interval depending on the size of present batch of active tags it last read (if the client has not specified their own reader report interval). However, the frequency of transmitting last read tags by the RRA to the RWS might depend on the kind of application. Some application would require a low latency while some applications can tolerate higher latencies and would prefer a lower load on the RWS. Hence, the client can specify the reader report interval in the Web based UI which is transmitted to the RRA in the next message cycle and the RRA would use that number as the reader report interval regardless of the number of tags. There also might be situations, where, despite affording a finite latency in tag reporting by the RRA, the client would like an immediate response on the occurrence of certain events. Examples could be to open the gate on detection of tag A, or switch off reader B on detection of tag C etc. The RRA has a separate functionality for processing such

events, where the client can create a rule, where the RRA should report to the web service immediately on detection of certain combination of tags at certain time spans which would lead to a certain user defined trigger. This would be discussed in greater detail in the event processing section.

4.1.2) Data structure: Fig. 4.5 shows the data structure of the RRA. It maintains two data tables: A tag table and an events table. The tag table stores the list of tags it detected and the time-stamp corresponding to the last time of detection. In the tag table, there are three columns, i.e. the tag ID, the time of tag read which stores the last time the tag was read and the tag update column.

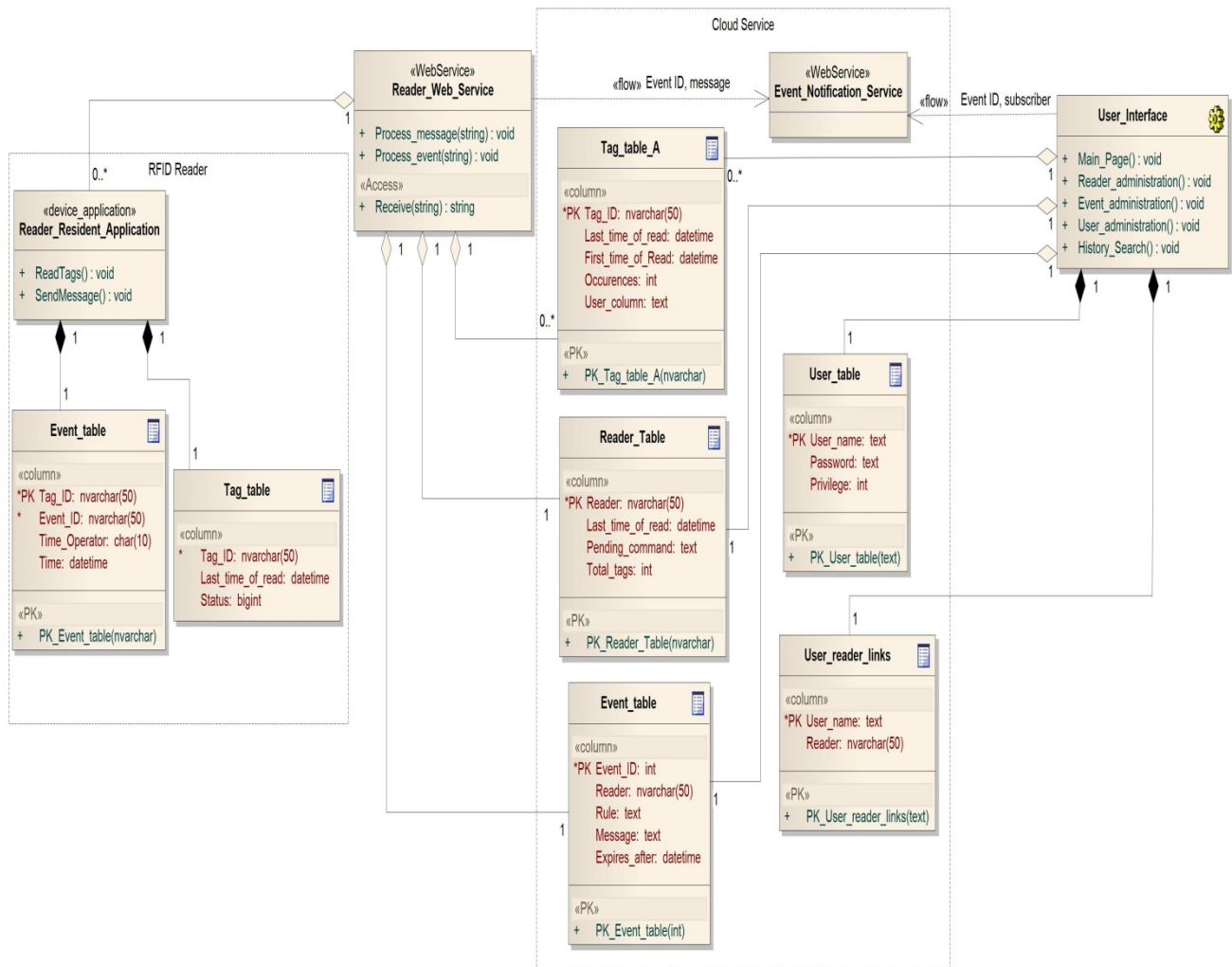


Fig. 4.5: Data model of the RFID solution

The tag update column (which is a boolean) is used to indicate if the corresponding tag has been re-read by the RRA since the last time it was transmitted to the RWS using a '1' or a '0'. If the value is '0', then it means the tag has not been read by the reader since the last transmission. This ensures that only newly read tags (with tag update column set to '1') are transmitted to the RWS. The events table is a data table which contains event based information that is to be used by the reader to detect the occurrence of an event and transmit a message to the RWS. The events table consists of four columns, i.e. the event tag (the tag to be detected), the decision time (the time before/after which the tag should get detected for successful occurrence of the event), the time logical operator (if the tag is to be detected before or after the specified time) and the event ID. An alternate format for the events table would also be discussed later, where the decision time column and the time logical operator column do not exist.

4.1.3) Sequence of operation: The sequence diagram shown in Fig. 4.6 explains the cycle of operation of the RRA. When the RRA is in operation, it reads tags by default unless commanded to do otherwise. A timer runs in a parallel thread whose task is to send the batch of last read tags to the reader every T_1 seconds (reader report interval) by means of a SOAP message. Whenever the RRA reads a tag, it is listed in a pending stack to be sent to the RWS in the next message cycle. The stack stores the tag ID and its last time of read. Once the last read tags are successfully transmitted to the RWS and receive a positive confirmation, the stack is cleared and the cycle repeats. If the reader reads a tag multiple times within the time span of one message transmission cycle, then the last time of read of that particular tag is simply updated in the stack table with the time of read. This time of read is updated whenever the tag is re-read and the tag update column is set to '1' which indicates that this tag read needs to be a part of the next

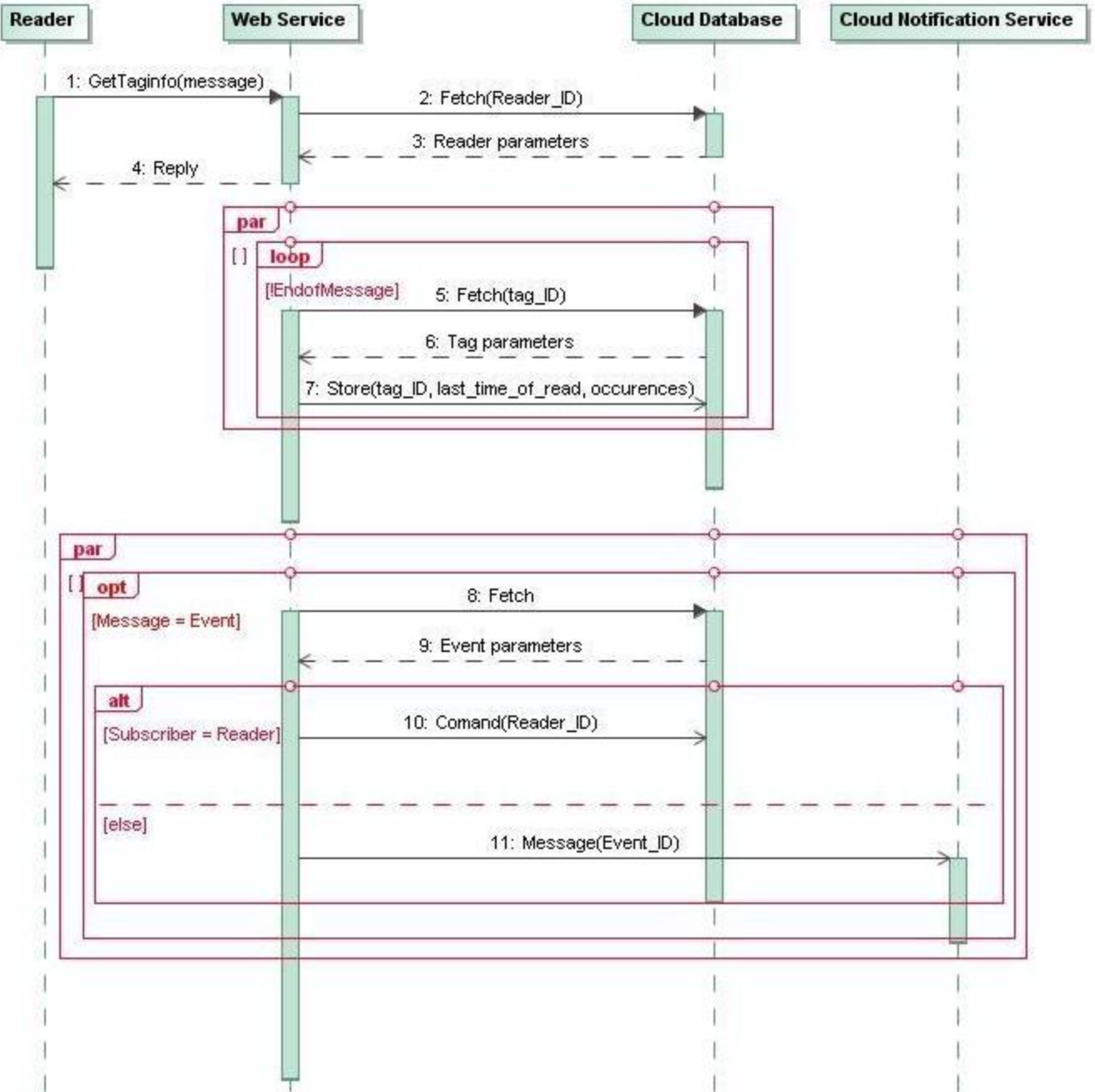


Fig. 4.6: Sequence diagram of the proposed RFID system

message transmitted to the web service. When the RRA compiles the message for the next transmission, it selects only those tags from the table which have a tag update column set to '1'. Then, it compiles the last read tags since the previous read into a batch along with their corresponding time of reads and transmits it as a message. After the message transmission, the

tag update column of these tags is set to '0'. So this enables the RRA to transmit only those tags from the tag table which have been read after the previous transmission and hence filter out older tags, thereby eliminating data redundancy during transmission.

4.1.4) Message structure and transmission: While transmitting data to the RWS, the reader needs to identify itself correctly, so that the RWS can differentiate the packets sent by multiple readers and store the data in tables corresponding to each reader. One obvious choice for identifying each reader was the IP address of each reader. However, this option will not work if multiple readers within the same intranet transmit messages to a server outside the network as the server will fail to differentiate their addresses. The readers will send their identifiers in the message packet to overcome this problem. This identifier helps the RWS differentiate between different readers within the same network. We chose the Media Access Control (MAC) address of the device (reader) as the identifier as it is unique for every device. The MAC address as the identifier of a reader might cause concern for the security of the RFID reader and its data; however, it can be easily replaced by some other form of identifier such as random number generator etc. A similar concept was discussed by Floerkemeier et al. where they proposed that readers existing in the same network can be aggregated and represented as a single entity ('virtual reader') to the higher layers [51]. However, if the higher layer would at a later stage need to differentiate between the two readers, such as for location purposes, there would be no way to do so and so our application solves that problem. The message format has been displayed in Eq. 4.1.

```
<soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <GetTagInfo xmlns="http://164.67.192.236/WebSite/chatserver/Service.asmx">
    <message>DE3EA259FF3FF8C37ECFA1DEACB7A3B1512DB6F12010-06-12T18:04:13</message>
  </GetTagInfo>
</soap:Body>
```

Fig. 4.7: Body of SOAP message

$$SOAPMessage_k = \{R_k ID_1 t_1 ID_2 t_2 \dots ID_n t_n\} \quad (4.1)$$

The packet begins with the reader identifier shown as R_k . This is followed by the contents of the data table of the reader i.e. the tag ID, ID_n , followed by the time-stamp of the read operation. The date and time are specified in the sortable date time pattern, “yyyymm-ddTHH:mm:ss”, shown as t_n . This is followed by the next tag ID and so on. If the reader-tags system were sensing and recording additional parameters such as temperature etc, then those parameters could be appended after the datetime for each individual tag. An example of a transmitted message packet is shown in Fig. 4.7. The message has only one tag ID which is 28 letters long (‘F8C37ECFA1DEACB7A3B1512DB6F1’) and its time-stamp (‘2010-06-12T18:04:13’). The reader MAC address is 12 hexadecimal letters (‘DE3EA259FF3F’). The sequence diagram of the message transfer from the RRA to the RWS has been shown in Fig. 4.6. The RWS processes the message and retrieves any pending commands issued for that particular reader from the reader table in the cloud database and sends it in the acknowledgment message to the reader. The reply received by the RRA typically contains two types of information, i.e. the reader report interval T_1 and a reader command. It might be possible that a reader command has not been issued for a reader and in that case the reader will only contain the former. The reader report interval which defines the time gap between successive reader transmissions to the web service will always be present in the reply. On receiving the reply, the RRA parses it and performs the command if any, else, it continues with the read cycle. In case, the reader has not read any new tags since the last read, the RRA simply transmits an empty message and receives the reply containing any commands issued by a user through the web based interface or a triggered event. In a large scale scenario if the RWS is receiving tag information from multiple readers, then having each reader periodically send message packets of tag information would consume less bandwidth and

processing power than maintaining a continuous communication channel with every individual reader and hence a reader initiated periodic message is justified.

4.2 Cloud Service

In this section, the cloud service is explained which offers the database and event notification services for the RFID solution. Such a model enables the supply chain manager to distribute the system tasks to other organizations willing to handle them as a separate service and reduce his setup time and costs. A client organization interested in setting up an RFID network only needs to create an integrated account by the cloud service provider. This will automatically set up the RWS, cloud database and event notification services for processing the data emanating from the RFID readers. The Web based UI would also be automatically set up by creating this account. This creates a simple model for outsourcing part of the RFID system operations to the service provider. The operational usage based charging of the cloud service makes it suitable for both small and large scale users as they only pay for the amount of resources they have consumed.

4.2.1) Database: A cloud based distributed database (Amazon SimpleDB) has been used to host our event data [52]. Conventionally, this type of functionality is performed by a clustered relational database that requires an upfront investment for setup and also necessitates a database administrator to maintain and administer. The database usage is charged by measuring three quantities, viz. the amount of structured data storage, the amount of data transfer and the machine utilization for the data operations also known as boxusage. We would be analyzing the boxusage patterns for different operations in the following sections. In the Amazon database, tables are called domains, every row in a table corresponds to an item and every column in a row is called the attribute of the item. The tables have been represented in Fig. 4.5. The database has primarily five kinds of tables: Tag tables, Reader table, Events table, User's table and User

reader links table. The tag table corresponds to the table created for each participating reader listing their corresponding tags. The tags detected by the reader are represented as items. Since in this study we are modeling the RFID solution on web services, for the sake of generality, we will limit ourselves to three attributes with respect to each tag, i.e. the last time of detection, first time of detection and the number of occurrences/reads. The table can also have additional user-defined attributes pertaining to each tag (object name, temperature etc) which can be added on the web based interface to be discussed later. Many previous works such as [25], [26], [38] and [39] have shown that the reader ID, tag ID and time-stamp of reader-tag interaction are necessary for higher level event processing. This is needed by many typical supply chain tracking solutions, where the primary goal is to track movement of shipments by processing times of entry, exit, presence, absence etc. Other modifications can be made to the table which would be discussed in the web based interface section. The reader table is a table listing all the readers that have communicated with the web service. The primary function of this table is to store operational information corresponding to each individual reader. The reader IDs of each reader are represented as items. The different attributes corresponding to each reader stored are total number of tags, reader report interval, pending reader command and last time of read. The reader table can also have additional user defined attributes. One common attribute would be reader location as many operations used the location of fixed readers as a reference to identify positions of tags. The event table is a table listing all the events created by authorized users. The unique event IDs are the items of the table. Each event has some attributes such as the associated reader, the event rule, the event message which is to be delivered on successful occurrence of the event, the event expire condition and the last event occurrence time. Finally, there are two tables, user table and user reader links which are used for user administration. These tables are only used by

the web based user interface and are not accessed by the RWS. User table stores the username as the item and the corresponding password and the privilege level of the user as its attributes. User reader links is a table which stores the list of associations of the users and the tag table's names (readers) that they are allowed to access. The user name is the item while the tag table name is the corresponding attribute.

4.2.2) Event Notification Service: The second component of the cloud service is the cloud notification service. This is used as the event notification service for the RFID system. In this service, an event can be created. This event can be subscribed by subscribers. The subscribers could be an external URL, web service or an email address. Any message posted on the event by the RWS is forwarded to the subscribers. A message is posted on the event notification service by the RWS whenever the RWS validates a successful trigger of an event. This has been shown in Fig. 4.6. The event notification service usage is also charged by measuring three quantities, viz. the number of requests made to the service, the number of notifications issued by the service and the data transfer carried out by the service.

4.2.3) Reader Web Service (RWS): The RWS contains methods which perform three primary functions: 1) It accept messages from the readers to process them and store their contents to various tables in the cloud database, 2) It executes messaging to subscribers based on successful event triggers by posting on the cloud notification service and 3) It retrieves and sends commands issued by clients on the web based user interface or through successful event triggers to the readers. These three methods are asynchronous and happen in parallel to reduce processing bottlenecks. The performance aspects of the RWS would be discussed in greater detail in Chapter 6.

4.2.4) Sequence of Operation: The RWS is sitting on the cloud. A typical SOAP message has

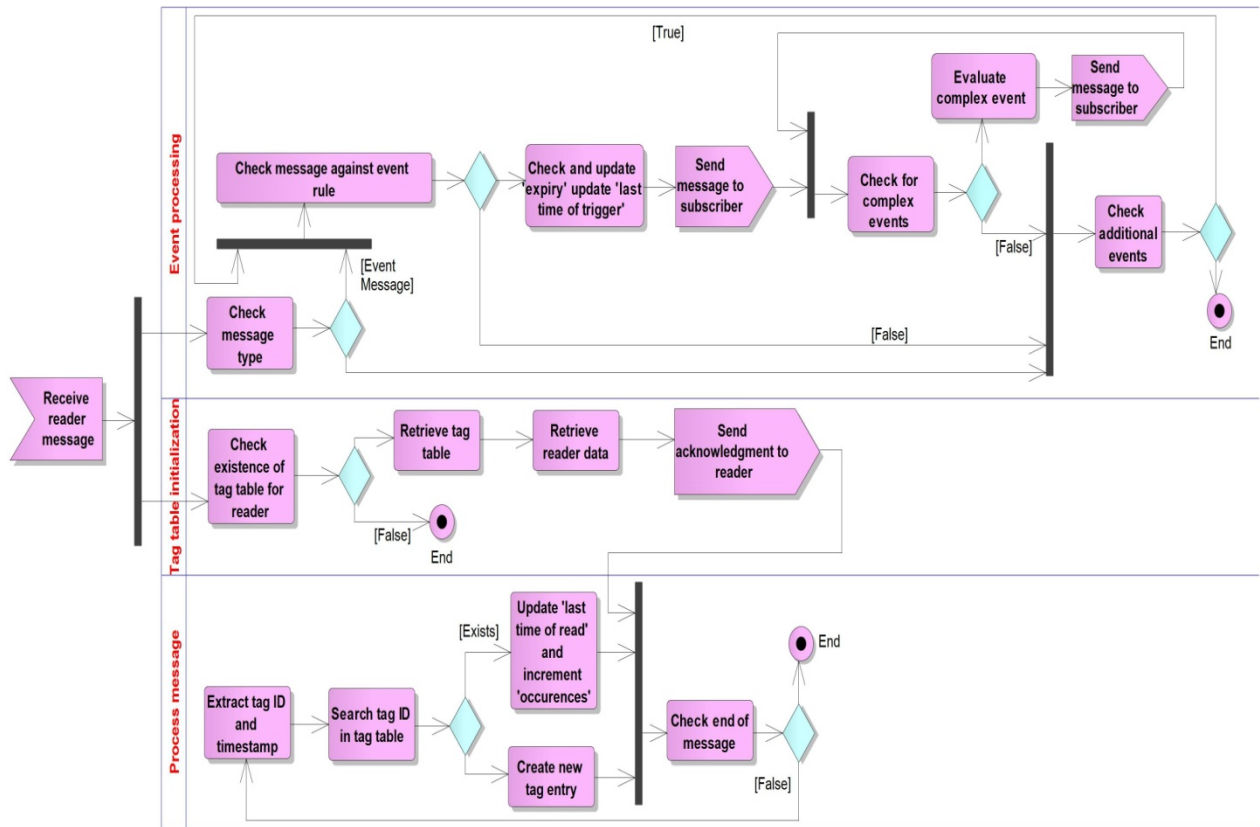


Fig. 4.8: Reader Web Service (RWS) flowchart

been shown in Fig. 4.7. As discussed in the above section, the RRA sends the last read tags to the RWS as a SOAP message. The SOAP message when received by the RWS is processed in the following way (as shown in Fig. 4.8): The service first extracts the reader MAC address from the message. If the MAC address matches to a previously registered reader, then the service retrieves two operational parameters corresponding to the reader from the Reader table i.e. 1) The reader report interval and 2) the ‘pending reader command’ issued by an authorized remote user through the UI or an earlier event (if any). These two pieces, if available, are compiled as a single message and sent back to the reader as a reply. In most cases, if there is no remotely issued command, the RWS method only sends the reader report interval as the reply to the reader. After sending the reply, the ‘pending reader command’ column is cleared. Since the remote command corresponding to a particular reader will be executed only when the RWS receives a message

from that reader, the minimum latency of the execution of a remote command would be the same as T_1 . It should be observed that, since, retrieving the reader parameters from the reader table is a parallel operation, the RWS always takes the same duration to reply back to a reader irrespective of the size of the message. This prevents the reader to keep waiting for the RWS to process the message and then reply back and hence, prevents a bottleneck. If the message is not empty and contains tag IDs and their timestamps then the RWS updates the tag table corresponding to that reader in the cloud database through a parallel thread. The name of a tag table corresponding to a particular reader is the same as the MAC address of that reader. After the table has been retrieved, the service extracts each tag ID and its corresponding time-stamp from the message and inserts/updates it to the table. If the tag ID already exists in the table, then the service updates the ‘last time of read’ column of the table with the latest time-stamp of the tag ID, otherwise the service adds the new tag ID to the table and enters data for ‘first time of read’, ‘last time of read’ and ‘occurrences’. The web service also increments the number in the ‘occurrences’ column to keep a track of the number of times the tag was tracked by the reader. The number of occurrences of a tag along with a log of the reader report interval can be used to predict the time estimates of each read of the particular tag which can be useful for historical data based tracking applications, although that functionality has not been implemented yet. The ‘last time of read’ of the row corresponding to the reader in the reader table is also updated. Then, the service extracts the next tag ID from the message and repeats the process until the end of the message. This is shown in Eq. 4.2.

$$C_{max} = T_1 \tag{4.2}$$

4.3 Event Processing

This section explains how an event is processed at different stages of the architecture.

4.3.1) RRA: In the previous sections, it was discussed that the RRA periodically sends a batch message consisting of last read tags to the RWS. However, there might be some situations where a finite delay might not be tolerated and an instantaneous reaction to an event would be desired. The RRA has a provision to detect the occurrence of such user specified events. An example of an event could be the detection of tag x after/before time t. Each time the reader reads a tag, it looks up the events table to check if that particular tag read occurred in the specified time range for a corresponding event ID. If a match exists, then the reader will instantaneously send an event message containing the event ID and the last read tags (since the last transmission) and their times of read. The RRA compares the tag data against the event parameters stored in an event table. Such events could act as trigger for initiating an imminent operation in the supply chain by the RWS. The instantaneous transmission of an event message removes the inherent delay associated with the normal periodically transmitted message, thereby, maintaining the fidelity of the event. Since many practical trigger based events would require instantaneous triggering of an action on detection of certain events, the above mechanism would make it possible, while otherwise transmitting non-trigger based tag data with a permissible delay. Some examples of instantaneous tag based events could be triggering to open the gates of the warehouse if certain groups of tags are detected simultaneously etc. Event triggered messages have an additional keyword “EVENT” and the event ID prefixed to the message which enables the RWS to redirect the message to the event processing method for processing. If multiple events are reported in a single message, then all the event IDs included in the message are separated by ‘,’.

4.3.2) RWS: The RWS contains the event processing method which runs in a parallel thread as shown in Fig. 4.8. The RWS processes the event triggered messages sent by the reader. An

event-ID number enables the RWS to retrieve the event processing logic of the corresponding event from the event table and perform a logical event condition check. If the event condition is not satisfied, then the method ignores it. If the event condition is satisfied, then the web service method sends the preset message (specified in the 'Message' column) to the preset event subscriber by posting on the event topic on the cloud notification service, as shown in Fig. 4.8. The cloud notification service manages the list of subscribers subscribed to an event with a particular event ID and simply forwards the message posted by the RWS to the event ID which reaches all subscribers. The event subscriber could be a web server address or an email address or a phone number. The possibility of messaging web addresses allows the linking up of the RFID supply chain with external web services, offering additional services. After the successful trigger of the event, the RWS logs the trigger time which could be used for validating a complex event rule. The RWS will also search if the event is a part of a complex event in the database by searching for event rules. A complex rule is a logical and temporal combination of simple/primitive rules. An example of a complex rule could be if Event 1 AND Event 2 occur AFTER Time t, then perform certain action. If true, then the RWS will evaluate the occurrence of the complex event too. An event can also have an expiry condition which is stored in the 'Expires after' column. The value in the 'Expires after' column could be either a date-time or an integer. In the former case, the RWS deletes the event from the event table after the specified date time. If the value is an integer, then the event would be deleted (expired) after those many successful occurrences as the value specifies. In case an event has expired and the reader sends an event message corresponding to an expired event rule, the RWS method will check if the rule exists in the event table and if it does not exist anymore or has expired, then it will issue a command for the corresponding reader(s) to delete the corresponding event trigger parameters

from their internal tables. If the message contains multiple event IDs, then the RWS will cycle through the above explained process until all events have been processed. An event can also be created with a reader as a subscriber. This kind of an event can be used for creating automated triggers to command other readers on occurrence of certain events. An example could be, after the arrival of certain tags, all the readers in the warehouse are commanded to stop reading. If the user has created a rule with an event success message destined for a reader, then the client interface does not create message topic on the cloud notification service as the readers do not have a unique address. In this case, the reader ID of the message receiver reader is simply appended to the event ID. This variation of the event ID allows the RWS to differentiate a rule addressed to a reader. If the event is successful, the RWS can simply extract the reader ID from the rule ID and issue the event message as a pending reader command on the reader table which would be sent to the reader in the next transmission cycle. This has been shown in Fig. 4.8. Each time a reader recognizes the occurrence of a single tag rule, it forwards the last read tag pop In this process for the actual rule. This technique ensures minimal processing burden on both the reader and the RWS while simultaneously maintaining the fidelity of the event triggering mechanism. Finally, after transmitting the single tag events to the readers, the client creates a message topic in the cloud notification service. This message topic serves the purpose of notifying all its subscribers of the successful occurrence of the event by the RWS by posting the preset message. After creating the message topic, all the subscribers who were listed as the message recipients in the event creation phase are sent subscriptions to the message topic. The sequence diagram shows the sequence of operation of the entire web service. The RRA sends last read tags as a batch message which is processed by the RWS and stored/updated on the cloud database. In case, the message is an event message, the RWS compares the tags and their

timestamps with the message condition and in case of a successful event, fetches the event message and transmits it to the notification service. This message sent by the RRA to the RWS will be further parsed and compared with the event condition to verify the occurrence of the event by the web server.

Chapter 5

Web based User Interface

This chapter explains the different parts of the Web based UI and their functionalities. The Web based UI is shown in Fig. 5.1. The web based interface acts as an interface between the remote user and the data stored in the cloud database (which is in turn updated/populated by the RWS). This serves as the primary user interface to monitor, control and create all kinds of RFID operations by the user. This is hosted on a cloud server by the RFID service provider (user name: Administrator, password: Administrator) [53]. By allowing the authorized users to access and view all events and controls in a web based format removes the need for local middleware/client installations on customer machines (except for an application installation on the reader which could come factory installed) and allows them to remotely monitor and control the RFID reader's behavior. The UI has five parts. Different parts have different access levels as determined by the administrator. The different accessibility levels help the administrator to prevent unauthorized access to critical RFID reader operations and behavior. Every user needs to login into the UI through a login page as shown in Fig. 5.2. The administrator can control reader access by allowing the user to access only some of the entire group of readers in operation. Further, the user access to individual operations (such as issuing commands to readers etc) can also be controlled according to their privilege level. All pages in the UI have an upper and a lower blue colored panel bar. The lower bar has a button which toggles from green colored "Active" mode to red colored "Inactive" mode. This button enables/disables the auto refresh activity of the page. Auto-refresh mode of the page continually refreshes the page after an interval specified by the "Reader Report Interval" which is set in the right panel of the Page. In the auto-refresh mode, the UI retrieves the data from the table being currently displayed on the UI from the databases to

ent user name : Administrator Logout Event table History search Reader administration User admin

Selected reader: F070266DF5C6 Page N^o: 1

refreshed at 3/19/2011 12:34:20 PM

Serial No	Tag ID	Last time of read	First time of read	Occurences	Object
1	0D2898441C1E3EDF310D5E95181B	2011-03-19T12:33:54	2011-03-19T12:25:57	6	
2	F25B5BAE690486B370C7530C4414	2011-03-19T12:33:54	2011-03-19T12:25:57	6	
3	9891A7F497A4CD4CAEF4EC032521	2011-03-19T12:33:47	2011-03-19T12:25:57	5	
4	DFB8BC3E74F89B02A962C7FF8D01	2011-03-19T12:33:36	2011-03-19T12:25:57	4	Red Shirt
5	E6F75FDA506F8BACBE3D8CC25D76	2011-03-19T12:33:30	2011-03-19T12:25:57	5	
6	2A7541F14F2057EB40513E98F928	2011-03-19T12:33:30	2011-03-19T12:25:57	8	Blue Shirt
7	AE9D5831A40B75CCFEC7D78178DA	2011-03-19T12:33:25	2011-03-19T12:25:57	6	Black pant
8	967CB5BBCC143731A05839D01D4D	2011-03-19T12:33:19	2011-03-19T12:25:57	7	
9	F8B5DC5B2166335E8C703D837EDD	2011-03-19T12:33:19	2011-03-19T12:25:57	6	
10	5BECDBBC513038386CDD883CCBF3	2011-03-19T12:33:13	2011-03-19T12:25:57	5	
11	C2D70CEDFE7D6765F41CCC14F50D	2011-03-19T12:32:51	2011-03-19T12:25:57	7	Green Shirt
12	E213050EFAED815C413C3DC3188C	2011-03-19T12:32:34	2011-03-19T12:25:57	2	
13	F9DEF72EF4095790F26A92AE49A6	2011-03-19T12:32:34	2011-03-19T12:25:57	4	
14	6D32DC682E31225C7FF5AF8294EC	2011-03-19T12:32:23	2011-03-19T12:25:57	3	
15	7190E00CECB34A9D01CE5B6A1152	2011-03-19T12:32:18	2011-03-19T12:25:57	3	
16	2E6BE819F46303E06C5A2D14BCEC	2011-03-19T12:32:18	2011-03-19T12:25:57	4	
17	FBFD42DCDA48B9DB7CC694CE09D3	2011-03-19T12:32:01	2011-03-19T12:25:57	5	
18	797DA8ADC85DBA0C52AF29991D9A	2011-03-19T12:31:16	2011-03-19T12:25:57	6	
19	6C828993440ABD5E8E713B53AAD5	2011-03-19T12:30:48	2011-03-19T12:25:57	2	
20	0312B0DD6DC6486689B6456F56F2	2011-03-19T12:28:56	2011-03-19T12:25:57	4	

Delete Selected Tags

ending reader command: Clear pending command

Edit Table entries :

Remote control ON

Select column: Object

Add column

Delete column

Add entry

Reader Controls :

10 Reader report interval (sec)

Send Command

Start read Stop read

Rule creation :

Create event notifier rule

Reader status: Active Auto refresh status: Active Total number of tags: 20

Fig. 5.1: Web based interface for user

Favorites Login

Login

Users Name: Administrator *

Password: ***** *

Login

Fig. 5.2: User login page

show the latest status of the data. This could be details of the tags being read or the readers or events. If the button is in “Inactive” mode, the page will not auto refresh. By default, the pages

are set to auto-refresh mode. Since, every retrieve operation from the database adds data usage charges to the users web service account, disabling the auto refresh mode helps in saving costs.

The upper panel shows the user name of the user logged into the UI. The log out button is placed besides the user name. On the right side of the top panel, the navigation buttons to view the other tables have been provided such as Events table, Reader table etc. Other than the Main page, all other pages only have a “Back” button to return to the Main Page. In the following pages, every page of the UI would be explained in greater detail.

5.1 Main Page

This is the main user interface where the client can do real time monitoring of tags, their time of reads and other details of the associated objects to which the tags are attached. Through this page, the client can also issue real-time commands to remote readers. Primitive events are also created on this page.

After signing into his account, the user is usually presented the default page which shows the tag table of the selected reader. The tag table has 4 primary attributes/columns:- the tag ID, the last time of read, the first time of read and the number of times it has been read by the reader (Occurrences). The tags are arranged according to the order in which they were read with the last read tag being on the top. This table is refreshed (re-fetched) after a certain interval equal to the read rate interval of the reader to reflect the latest tag reads of the reader. If the tags associated with the reader are more than 100, then they are shown in multiple pages. The page number can be chosen using a drop down page selector menu on the top right of the table. The tags are also color coded to show the age of the tags. The tags with a darker green shade are the most recently read while the tags with a more reddish shade are older. The user can select other readers (if available and authorized) from the drop down menu in the top left and view their corresponding

tags. There is a middle panel provided between the top panel and the table. This panel shows the data corresponding to the present state of the reader. It shows the last time the tag data was retrieved from the database. This helps the user ensure, whether the UI is periodically refreshing and retrieving data from the database. It also shows the total number of tags read by the reader since it was turned on and the current status of the reader. If the reader has not been communicating with the RWS, then its status is shown as “Inactive” in red. Otherwise, it is shown as “Active” in green. At the bottom of the table, three buttons are provided. They are “Delete Selected Tags” which allows the user to delete tags from the table which have been selected by clicking on the tick box, “Select all” which allows the user to select all the tags displayed in the table and “Deselect all” which allows the user to deselect the tags he previously selected. Furthermore, the user can make modifications to the table such as add additional columns to the table, edit values in the additional columns etc. One example of adding a column to the table could be assigning each tag to an associated object. So the user could create a new column called ‘Object’ and then enter the description of the object. This has been shown in Fig. 5.1. Many applications require the storage of long term histories of tags as an when they are detected by multiple readers. Such long term data is usually used to analyze the movement of the object associated with the tag. Such as the routes taken by tagged shipments as they are transported from one warehouse to the other etc. The button titled “Detailed tag history” shows the user, upto the last 250 instances of the tag.

On the right panel of the page, a Reader control panel has been provided. This panel can be used to send remote commands to the reader. Some common remote commands incorporated in the panel are ”Stop Read”, ”Start Read” and change the reader report interval. Any remote command or change in the reader report interval issued by the user for a particular reader is stored in the

'Pending reader command' and the 'Reader report interval' column of the reader table respectively to be transmitted by the RWS to the RRA in the next message cycle. Other custom commands such as "Kill tag", "Write tag" can also be issued by typing on the text box and pressing the "Send Command" button. A toggle button to enable/disable the ability to disable the remote control of the reader has also been included in the interface ("Remote Control On/Off") to avoid sending accidental commands to the reader. On pressing this button, the remote commands button such as "Stop Read" and "Start Read" are disabled/enabled based on their previous state. On the bottom left part of the page, a command issued by the user which is pending to be sent to the reader in the next read cycle is also displayed. This can act as a confirmation to the user, that the command is queued to be delivered to the RRA after it has been issued. In case, the user wants to withdraw the command previously issued by him, he can click the "Clear pending command" button which is placed next to the displayed pending reader command. However, this button is effective only before the next read cycle after which the pending reader command is already transmitted to the reader.

The default page also acts as an interface for setting up rules for event based triggering. On pressing the "Create event notifier rule", (Fig. 5.3) the user is presented with the event creation panel. In order to create a rule, the user specifies the reader for which he wants to create the event in the text box next to the 'Set Reader' button. If the user does not enter any reader name and simply presses the button, then the interface would select the reader corresponding to the tag table in the background. Then, the tags that should be part of the event are selected from the tag table list displayed in the background and entered. The user can also manually type the tag ID if it is not in the list on the rule text box. The relationships between the tags can be set by selecting

<input type="checkbox"/>	15	A420B4C44F3FBFEFA1F1E07E73A99	2011-03-21T00:32:43	2011-03-21T00:16:28	8
<input checked="" type="checkbox"/>	16	6DBB4C1A023748C2F5A2AA6F073A	2011-03-21T00:32:43	2011-03-21T00:16:28	13
<input type="checkbox"/>	17	F68DC9679E206B119C9			
<input type="checkbox"/>	18	A7210A44677B6F63589			
<input type="checkbox"/>	19	B963AF4B7845989C45C			
<input type="checkbox"/>	20	CE42F648D9AB26DBD			
<input type="checkbox"/>	21	22F037C5BB0CA97F6A			
<input type="checkbox"/>	22	C0A19F4A46095D4E13			
<input type="checkbox"/>	23	5C7403F81A5132F3C04			
<input type="checkbox"/>	24	3D0E2C58658B438568E			
<input type="checkbox"/>	25	78ED02405BF13D91D07			
<input type="checkbox"/>	26	E244C35118C05D58AF			
<input type="checkbox"/>	27	B5154EF6B7210EA1461			
<input type="checkbox"/>	28	EB93BD40ED5BC61468			
<input type="checkbox"/>	29	A89CDD41C45D878CB			
<input type="checkbox"/>	30	6FFD493EDA569C7A72			
<input type="checkbox"/>	31	1A4930C78315B6CDAD			
<input type="checkbox"/>	32	1A95A01102FB799CDB			
<input type="checkbox"/>	33	B0EE31DE7D80D14371			
<input type="checkbox"/>	34	89AEA77B4998B1C1094			
<input type="checkbox"/>	35	8FA4D4300BADAD2127			

Create event notification rule :

Select logical operator : OR

Tip the reader name :

Select tags in the list and include them :

Event occurrence time Rule expiry time

Event completion message: Select time operator : after

Select time : 17 : 19 : 19

Select the date :

Select receiver type and fill the address : E-Mail

Reader: FFFFFFFF Tag: 6DBB4C1A023748C2F5A2AA6F073A after 2010-12-26T17:19:19 To arunabh71@gmail.com Send 'Arrived' Expires after 1

Server status: Active Auto refresh status: Inactive Total number of tags: 35

Fig. 5.3: Creating an event on the user interface

logical operators (for example, Tag A AND Tag B or Tag A OR Tag B etc). After the tags have been elected, the time range within which the event should occur is entered (for example, between 2011-01-27T12:10:20 and 2011-01-27T19:12:20 or after 2011-01-27T12:11:12 etc). Once the event trigger condition has been set, the event action items need to be set i.e., on the successful trigger of the event, what kind of action needs to be taken. The action in this context involves sending a preset message to a pre-defined subscriber. The subscriber could be an email address or a web address (such as web address of a server or a web service etc) or another reader. Some example of messages could be to open a gate (or toggle some other type of switch of some device) on detection of certain tag ID or send a certain command to a reader (like stop read) on detection of certain tag ID etc. The message could also be forwarding the tag ID and corresponding timestamps to the subscriber. This allows the administrator to choose which data

can be accessed by external subscribers and hence maintain data privacy. Finally, after setting the preset message and the subscriber, the user needs to define the expiry condition of the rule. The user can either define a date-time after which the rule would expire or define an integer which gives the number of successful occurrences of the event after which it expires. If the user does not specify the expiry condition, then the rule is assumed to be permanent. After the complete rule has been created by the user, the client interface first stores the rule in the events table. The event table contains the columns of 'Event ID', 'Reader ID', 'Rule', 'Message' and 'Expires after'. The contents of these columns are self-explanatory from the name of the columns as was discussed previously. The client interface will parse the user created rule for the above listed details and fill the table columns. After registering the event rule in the events table, the client interface would extract the tag IDs and compile each one with the event ID and update the corresponding reader entry in the reader table by filling the 'Pending Reader Command' column to be sent to that particular reader for detecting events associated with that tag. In this way, the event rule has been broken down into smaller single tag rules for the lesser computationally capable readers to identify.

5.2 Reader Administration Page

The Reader table shows the Reader ID (which is the same as the MAC address), the total number of tags it has read since it was registered with the UI, the reader report interval which is the duration that the RRA waits before sending the last read tags to the RWS, the pending reader command which shows any pending commands issued by the user awaiting transmission to the reader in the next message cycle and the last time of read which shows the last time the reader communicated with the RWS (Fig. 5.4). This table also shows all the readers as a colored gradient where a greener shade represents a recently communicated reader, while an increasingly

2012 6:16:01 AM Total Number of Tags: 2009 Total Number of Readers: 4

Serial No	Reader	Total tags	Reader report interval	Pending reader command	Last time of read
	defba233bdeb	1	10		2012-04-16T00:33:43
	FFFFFFFFFFFF	2002	10		2012-04-13T16:26:50
	DE3EA259FF3F	4	10		2012-04-06T08:01:41
	00157084D75A	2	10		2012-03-21T08:28:09

Reader Controls :

Table Controls :

Registration ID:
Registration Key:

Fig. 5.4: Using the Reader Administration page of the user interface

red shade represents a less recently communicated reader. The mid panel between the table and the upper page selector panel contains three pieces of information. The first and leftmost piece shows the last time the table was refreshed. The mid information shows the total combined sum of all the tags that have been read by all the readers (accessible to the user) ever since each one of them registered with the RWS. The third and right most data shows the total number of readers visible/accessible to the user. This page also has the “Select all” and “Deselect all” buttons to be able to select or de-select all readers shown in the table. The right panel has 3 sub-sections. The topmost sub-section deals with modifying the data contained in the reader table. These functions are the same functions as explained in the Main page such as Add/Delete column, entering a value to one of the added columns. One example where this functionality is used is, to add location data to every reader. The user would first add a location column to the table using the “Add Column” button. After the location table has been created, the user would enter the location data of the reader one by one by typing the location data of the reader in the blank space, then select the corresponding reader in the checkbox, select the “Location” column from the drop down menu and press the “Add Entry” button. Beneath the table control panel is

the Reader Controls panel. This panel is similar to the Reader Controls panel in the main page and contains the same buttons as discussed previously. However, in this page, commands can be issued in bulk. The user can select readers from the table by checking the corresponding tick box from the table and then give commands like “Stop Read” etc. This will issue that command to all the readers selected in the table. The third panel in the panel bar is the “Table Controls” panel. This panel allows the authorized user to delete readers from the system or add new readers. In order to delete the readers, the user selects the readers from the table by ticking in their corresponding check box and then pressing the “Delete selected reader” button. The user can also change the reader report interval of multiple readers and the interval after which the reader administration page is refreshed using the “Table refresh interval” button through the same technique explained before. In order to register a new reader to the system, the user needs to enter the Registration ID and the Registration Key in the blank spaces provided in the panel. These two pieces of information is obtained from the reader as the RRA is initialized for the first time. This scheme enables the encryption of data exchange between the RRA and the RWS and prevents data spoofing by an unauthorized entity. When this information is entered, the UI creates the tag table corresponding to this reader and enters the corresponding reader data in the reader table. This enables the RWS of multiple readers using the “Table refresh interval”.

5.3 Events Administration Page

The next page in the client interface is the events administration page (Fig. 5.5). In this page an authorized user can see the list of events created in the event table for readers associated with his account. The events table in the page has the following columns:

Total Number of Event Rules: 4

Select Event	Serial No	Event ARN	Reader	Rule	Message	Expires after	Last time of trigger
<input type="checkbox"/>	1	20110405173504	FFFFFFFFFFFF	Tag:BBAA99887766554433220100 after 2010-12-26T17:19:19	OnDemand	197	2011-04-05T18:22:24
<input type="checkbox"/>	2	20110405173735	FFFFFFFFFFFF	Tag:BBAA99887766554433220200 after 2010-12-26T17:19:19	OnDemand	200	
<input type="checkbox"/>	3	20110405180353		Rule:20110405173504 AND Rule:20110405173735 after 2010-12-26T17:19:19	OnDemand	100	
<input type="checkbox"/>	4	20110405182134	FFFFFFFFFFFF	Tag:BBAA99887766554433220300 after 2010-12-26T17:19:19	OnDemand	2012-12-26T17:19:19	

Table controls :

Rule creation :

Fig. 5.5: Event Table page

- 1) Event ID, which displays the Event ID of the event. Every event created is given an Event ID, which helps both the RRAs on the readers and the RWS to exchange and identify event specific data
- 2) Reader ID, which shows the ID of the reader associated with the event. It could be possible that in the case of a complex event, there is more than one reader associated. In such a case, the Reader ID column corresponding to that reader would be empty.
- 3) Rule, which shows the rule-based logic which is used to evaluate the successful occurrence of an event.
- 4) Message, which shows the requested message which is to be sent to the subscriber, on successful occurrence of the corresponding event. An example could be sending the message “Shipment arrived” after Tag A is detected by Reader B.
- 5) Expires after, which shows the number of times the event can occur after which it expires and is deleted by the RWS. An example could be that a shipment shipped from the factory to the seller goes through x stages (factory warehouse, shipper’s warehouse etc) where every stage has

a portal reader to record the arrival of the shipment. After the package has been detected x times which includes the total number of stages from its source to its destination, its rule should be deleted as its transportation process has completed.

6) Last time of trigger, which shows the date-time when the event last occurred successfully. This allows the user to see the last time the event occurred successfully. This data is also used to evaluate the rule of a complex event, if the corresponding event in the row is part of the complex event rule.

The event rows are also color coded to show the recency of the event occurrence. The event rows with a darker green shade are the most recently read while the events with a more reddish shade are older.

If the number of events in the event table are more than 100, they are displayed in multiple pages, where the page can be selected using the drop down page selector menu. The middle panel between the table and the page selector panel shows the last time the table was refreshed and the total number of events contained in the event table. The “Select all” and the “Deselect all” button toggle the selection/de-selection of all the events presently displayed in the table for deletion. The right panel of the events table page contains two sub-panels. The upper sub panel contains two buttons: “Delete selected rule” which is used to delete rules selected by ticking the check box against the corresponding event in the event table and “Table refresh interval” which selects the interval after which the event table auto-updates to show the latest table content. Apart from visualizing and editing event rules, another rationale for creation of this page is the ability to create complex rules. As an example, some users might want a complex event rule to exist which is logical and temporal combination of two pre-existing primitive rules. This page allows the user to select multiple rules and associate those through logical and temporal

Select Event	Serial No	Event ARN	Reader	Rule	Mes
<input type="checkbox"/>	1	20110405173504	FFFFFFFFFFFF	Tag:BBAA99887766554433220100 after 2010-12-26T17:19:19	OnDe
<input checked="" type="checkbox"/>	2	20110405173735	FFFFFFFFFFFF	Tag:BBAA99887766554433220200 after 2010-12-26T17:19:19	OnDe
<input type="checkbox"/>	3	20110405180353		Rule:20110405173504 AND Rule:20110405173735 after 2010-12-26T17:19:19	OnDe
<input type="checkbox"/>	4	20110405182134	FFFFFFFFFFFF	Tag:BBAA99887766554433220300 after 2010-12-26T17:19:19	OnDe

Create complex rule :

Select logical operator : AND < December 2010 >

Select rules in the list and include them :

Event occurrence time Rule expiry time

Event completion message:

Select time operator : at 17 : 19 : 19

Select the date :

Select receiver type and fill the address : E-Mail

OnDemand

Rule:20110405173504 AND Rule:20110405173735 after 2010-12-26T17:19:19 To arunabh71@gmail.com Send 'OnDemand'

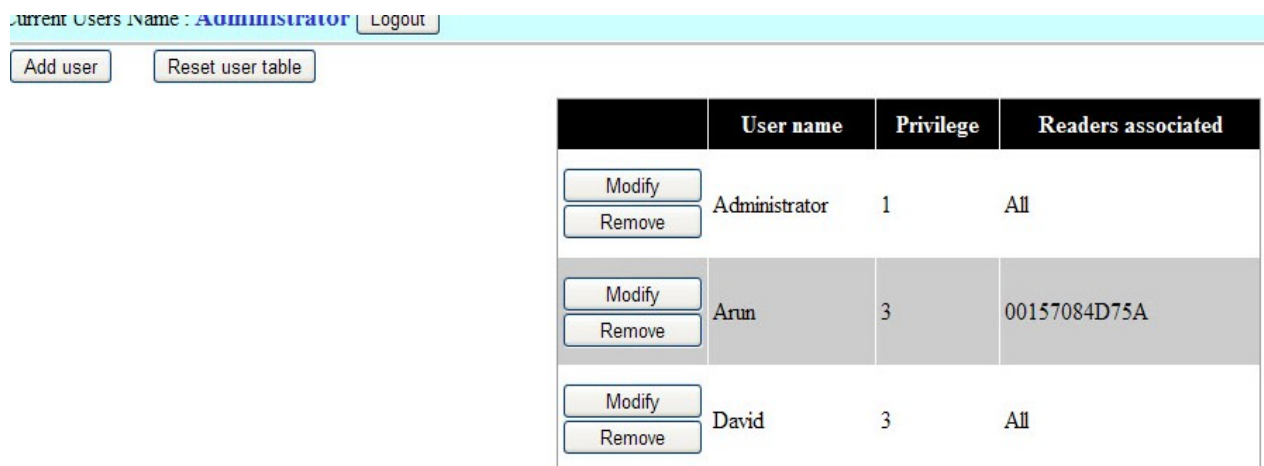
Fig. 5.6: Creating complex rules from simple rule on the event

relationships and then form a new complex event rule with a new event ID. The procedure for creating a complex rule is similar to the procedure for creating a simple rule as explained in the Main page section (Fig. 5.6). Instead of choosing readers and tags, the user selects events from the table. On pressing the “Create/edit rule”, the user is presented with the event creation/edit panel. In order to create a complex rule, the user specifies the rule with which he wants to create the event in the text box next to the ‘Set Rule’ button. The previous step is repeated to select more rules that will form a part of the complex rule. The user can also manually type the event ID if it is not in the list on the rule text box. The relationships between the events can be set by selecting logical operators (for example, Event A AND Event B or Event A OR Event B etc). After the tags have been selected, the time range within which the event should occur is entered (for example, between 2011-01-27T12:10:20 and 2011-01-27T19:12:20 or after 2011-01-27T12:11:12 etc). Once the event trigger condition has been set, the event action items need to be set i.e., on the successful trigger of the event, what kind of action needs to be taken. The action in this context involves sending a preset message to a pre-defined subscriber. The subscriber could be an email address or a web address (such as web address of a server or a web

service etc) or another reader. Some example of messages could be to notify the user (or toggle some other type of switch of some device) on occurrence of certain events in a certain sequence etc. The message could also be forwarding the tag ID and corresponding timestamps to the subscriber. This allows the administrator to choose which data can be accessed by external subscribers and hence maintain data privacy. Finally, after setting the preset message and the subscriber, the user needs to define the expiry condition of the rule. The user can either define a date-time after which the rule would expire or define an integer which gives the number of successful occurrences of the event after which it expires. If the user does not specify the expiry condition, then the rule is assumed to be permanent. After the complete rule has been created by the user, the client interface first stores the rule in the events table. The event table contains the columns of 'Event ID', 'Reader ID', 'Rule', 'Message' and 'Expires after'. The contents of these columns are self-explanatory from the name of the columns as was discussed previously. The client interface will parse the user created rule for the above listed details and fill the table columns. After registering the event rule in the events table, the client interface would extract the event IDs and compile each one with the event id and update the corresponding reader entry in the reader table by filling the 'Pending Reader Command' column to be sent to that particular reader for detecting events associated with that tag. In this way, the complex event rule has been broken down into smaller single events. An event can also be edited using the same panel, by clicking the "Create/Edit Rule" button. Once the create/edit rule panel appears, the user selects the rule to be edited from the event table by ticking the check box corresponding to that event in the table and then pressing the "Edit Rule" button on the menu. This causes the rule to appear on the bottom rule composition text box. Here, the user can type and edit the rule and finally press the "Create Rule" button to set the rule.

5.4 User Administration Page

In this page, the administrator can add or remove users who can log in to the UI. This page also allows the administrator to set which users can access which readers and how much control they have over the readers (such as the ability to give commands, ability to create event rules etc). This page utilizes the user table and the user reader links table as discussed previously. This page is accessible only to the administrator. (Fig. 5.7). The UI allows three kinds of privilege levels: Administrator, Super user and Simple user. The administrator can add or remove additional user accounts which can access the user interface as shown in Fig. 5.8. The administrator also decides what readers can be viewed by users, the amount of access the users have over the readers such as power to modify reader parameters etc. Other administrative functions include modifying username, password etc. The super user has a privilege level less than the administrator but greater than the simple user. A reset button has also been provided to delete all the users and retain only the administrator as the single user for the interface. The interface has to supply a way to reset the tables if a problem appears on the system. By default the system will have one user named “Administrator” with administrator privilege and “Administrator” as password. In addition, the user can access to all the readers. The reset button is available in the page and the



The screenshot shows the user administration interface. At the top, it displays the current user name as "ADMINISTRATOR" and a "Logout" button. Below this, there are two buttons: "Add user" and "Reset user table". The main part of the interface is a table with the following columns: "User name", "Privilege", and "Readers associated". Each row in the table has "Modify" and "Remove" buttons next to the user name.

	User name	Privilege	Readers associated
Modify Remove	Administrator	1	All
Modify Remove	Arun	3	00157084D75A
Modify Remove	David	3	All

Fig. 5.7: User administration part of the user interface

User modifications

User name :
Arun

Old password :
[]

Password :
[]

Confirm password :
[]

Privilege : Simple user (selected)
Simple user
Super user
Administrator
Cancer

OK []

Select readers in the list :
 00157084D75A
 F070266DF5C6
 1EDDF1681838
 729103263ED8
 16DBD945E6F5
 964DD18C85AE
 or all readers

Fig. 5.8: Modification of user privileges by the administrator

user is asked if he wants to reset the user tables (Deleting all the existing users and creating one user “Administrator”). The page contains additional buttons:

- Add user: Show a popup for adding an user
- Reset user table: Reset “user_table” and “user_reader_links” table
- Modify for each user: Allows modifying the user parameters

Remove for each user: Remove the user

When “Add user” is clicked, a panel asking the user parameters appears. It contains the following fields:

- User name to enter the user name
- Password to enter the password
- Confirm password to confirm the previously entered password
- Privilege level to set the privilege level of the user to “Simple User” or “Super User”

User Name		Administrator	Super User	Simple User
Privilege level		1	2	3
User administration	User administration access	Yes	Yes	No
	All user access in user administration	Yes	No	No
	Add user	Yes	Only Simple	No
	Edit any password without old password	Yes if user is not "Administrator"	No	No
	Edit own password with old password	Yes	Yes	Yes
	Edit own privilege level	No	No	No
	Edit other privilege level	Yes if user is not "Administrator"	No	No
	Remove own user account	Yes if user is not "Administrator"	Yes	Yes
	Remove other user account	Yes if user is not "Administrator"	No	No
	Edit reader access	Yes	Yes	No
	Modify own user name	Yes if user is not "Administrator"	Yes	Yes
Reader Administration	Reader administration access	Yes	Yes	No
Reset Tables		Yes	No	No

Table 5.1: User privileges table

Checkboxes for selecting the readers to select the readers accessible to the new user

Connections between users and readers

Every user has access to certain readers. Also, the system needs to know when a user logs in, which readers are accessible to him and which are hidden. The table contains three columns:

index column : when a new entry is created it takes the “last index” + 1 as index

user column

reader column

Each entry links a user with a reader. In the given example, David can use the readers “00157035DB94”, “BFEE5712D62A” and “3B666E305454”. The administrator can use all the readers. Remark: If the system were a normal database (e.g. MySQL), the architecture would have been different. Normally there would be a table for each user with the list of all the readers allowed. But this architecture implies a big number of tables because the more users exist, the more tables are created and yet the table number is very limited. It is more economical to create only one table listing all the user/reader links.

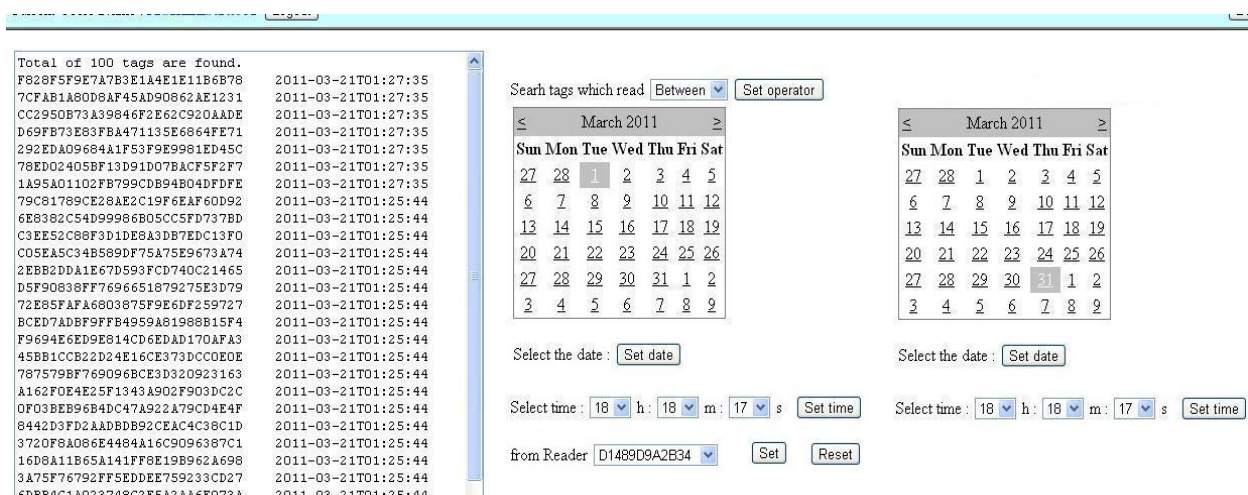


Fig. 5.9: Using the history search page of the user interface

5.5 History Search Page

This page is used to search previously read tags read by a particular reader in a given time span. The user can search tags before a given date time, after a given date time or in a slot between two given date times. The search operation retrieves all the tags in the asked time span and lists them and their last time of reads. The history search page has been shown in Fig. 5.9. The steps to display tags read by a particular reader within a certain timespan are as follows:

- 1) The user first selects the time operator using the drop down menu from choices of “Before”, “At”, “After” and “Between”.
 - 2) After selecting the time operator, the user selects the date from the calendar function and sets the date by pressing the “Set date” button.
 - 3) Further, the user selects the hour, minute and seconds from the respective drop down menus to set the time of the day for the event time condition
 - 4) User presses the “Set Time” button. Steps 2-4 process are repeated if the time operator “Before” is chosen in Step 1 to give the second time limit to define the time span.
 - 5) Next, the reader from the drop down reader menu is selected and the “Set” button is pressed.
- This action shows all the tags that satisfy the above condition and their date and time of detection in the upper text box. In order to carry out a new search, the user can press the “Reset” button under the text box to rest the search conditions.

Chapter 6

Load Testing

One of the primary concern that arises on migrating the RFID middleware and database functionality to a less powerful mobile device and a remote cloud location is the compromise in performance parameters especially processing delay due to longer processing times by the mobile device and the longer data transport duration to the remote cloud. In order to measure and compare the processing delays for the traditional architecture versus the cloud based architecture, we implemented the RWS in a local lab machine on an Intel 3.07 GHz Quad Core processor with 3.25 GB RAM running Windows XP. The database server running on this machine was Microsoft SQL Server 2008. The RWS logic on the local and the cloud server was identical. The application resides on a Motorola FX7400 UHF reader which runs Microsoft Windows CE 5.0 Operating System (OS) on an Intel PXA270 processor and on a Motorola MC9090 UHF reader which runs Microsoft Windows Mobile 5 Operating System (OS) on an Intel 624 MegaHertz (MHz) processor. The application was created using Microsoft Visual Studio and the Symbol Motorola Development Kit (SMDK). On the cloud side, the RWS was implemented on a Machine Image which uses a small amount of CPU resources and 613 Mb of memory [54]. The Machine Image processor is capable of increasing CPU capacity in short bursts when additional cycles are available. The Machine Operations can exploit up to 2 Compute Units (CUs) for short periodic bursts, where 1 CU provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. The Machine Image was running Microsoft Windows Server 2008 R2 Datacenter. The data measured in this test was averaged out over 10 identical runs. Fig. 6.1 shows the total delay it took for the data to be stored in the database tables after the message packets were transmitted by the RRA to the RWS as a function of number of tags. The

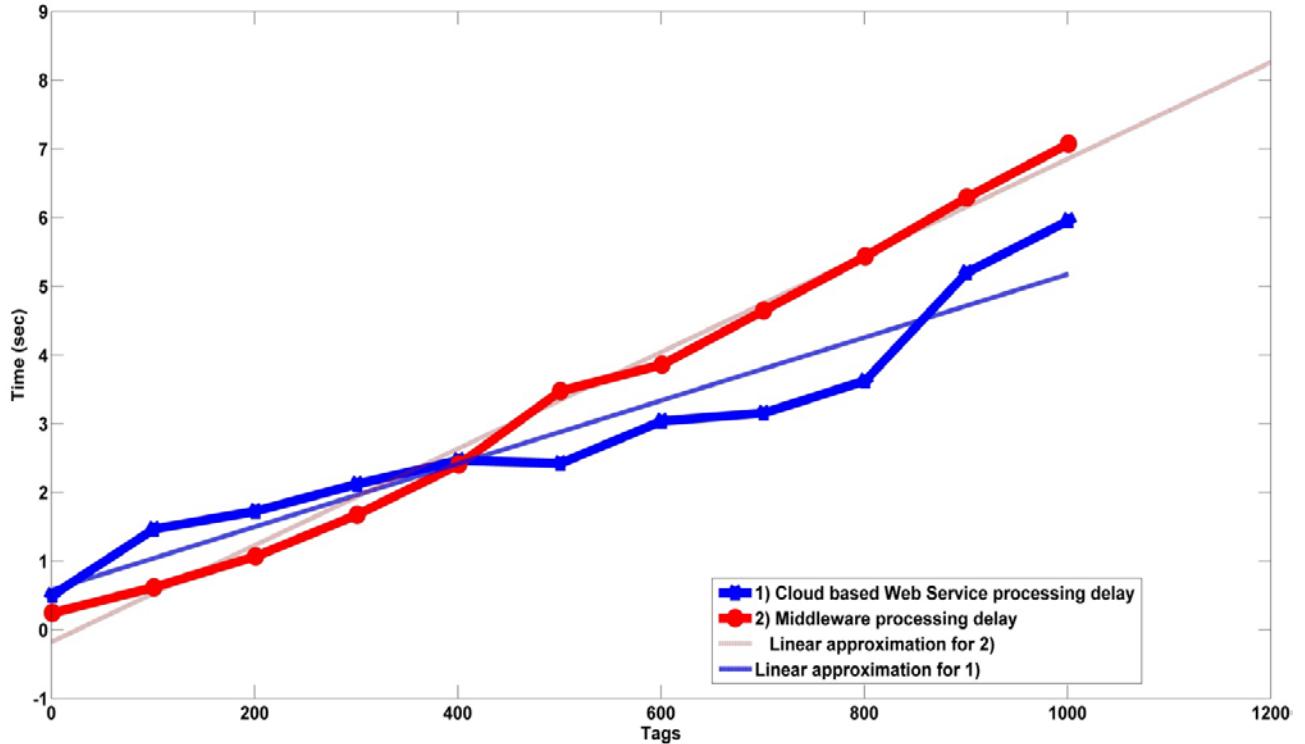


Fig. 6.1: Delay performance comparison of localized server and cloud based web server

total delay also includes the data transportation delay to the corresponding server. The average one way network delay from the RRA to the cloud based RWS was 0.2627 seconds. The same from the RRA to the local lab based RWS was 0.1066 seconds. As can be seen, the cloud architecture has an approximate average increase of 1.5233 seconds for every additional tag data in the message while the local architecture has an approximate average increase of 2.3311 seconds. Although, for small number of tags (<400), the local architecture has less delay, the cloud architecture demonstrates faster processing for large number of tags > 400). The smaller processing delay of the cloud based RWS helps it overcome the additional network delay disadvantage. This helps us conclude that a remote cloud based server can perform as well or even better than a localized architecture.

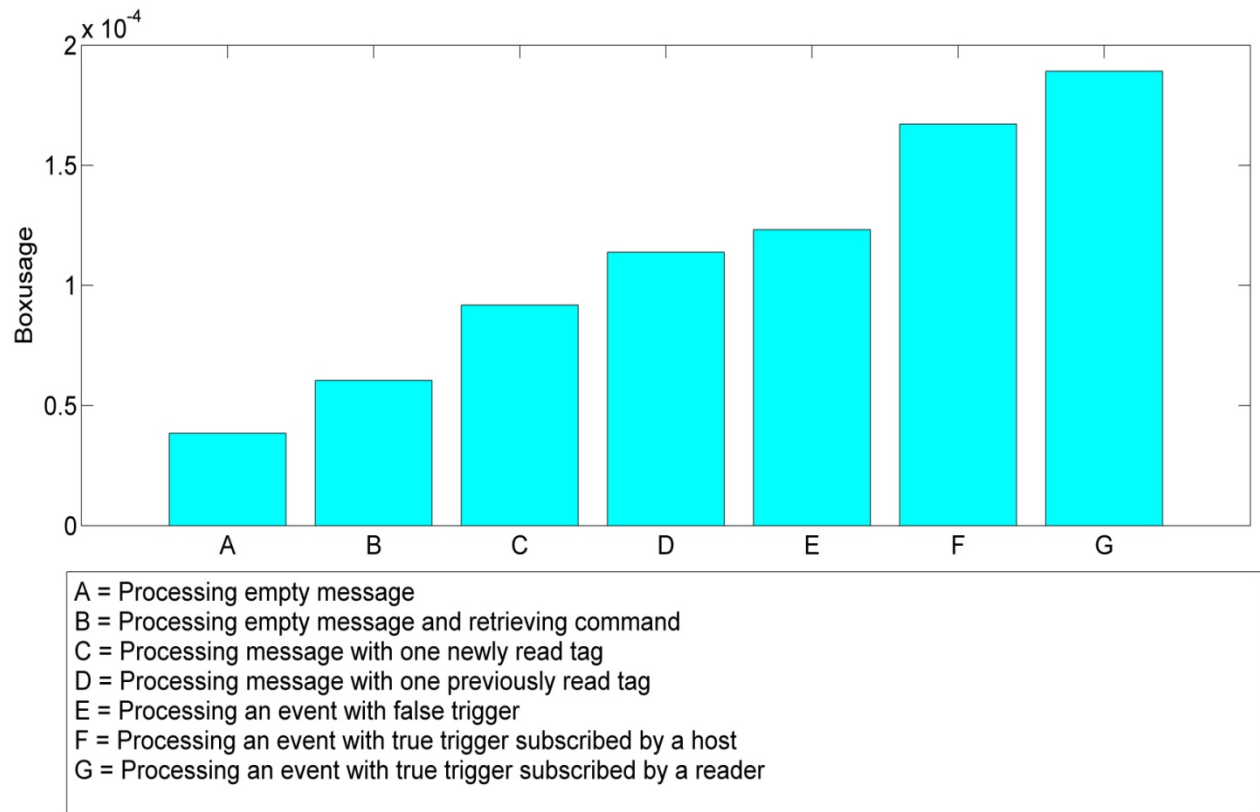


Fig. 6.2: Bar plot of boxusage of various reader web service

6.1 Boxusage consumption by RWS

Fig. 6.2 shows the cloud database boxusage of some typical operations performed by the RWS. Operation A shows the boxusage involved in processing an empty message sent by the reader. This message contains only the reader ID and no tag IDs. It has a boxusage of 3.85×10^{-5} . Since this message does not contain any tag IDs, the RWS only fetches the reader parameters from the reader table and sends a reply back to the reader. So the boxusage corresponds to the reader parameter fetch operation from the reader table. Operation B shows the boxusage for processing an empty message with a pending reader command. It has a value of 6.04959×10^{-5} . Operation B involves all steps of operation A and an additional step of clearing the pending reader command attribute of the corresponding reader in the reader table after sending the pending command in reply. Operation C which has a value of 9.18122×10^{-5} shows the boxusage of processing a reader

message containing a single newly read tag. It involves all the steps of operation A and two additional steps of fetching a list of all tag IDs stored in the corresponding tag table of the reader and then adding a new tag entry to the table. Operation D showing the boxusage for processing a reader message containing a single previously read tag has a value of 0.0001138159. The difference between operation D and operation C is that the former updates two columns ('last time of read' and 'occurrences') of a tag item in the tag table of the reader instead of adding a new tag item in the tag table as done in the former. Operation E shows the boxusage for processing an event with a false trigger i.e. the event trigger condition is false. It has a value of 0.0001231681. It involves all steps of operation D (assuming the event is based on a previously read tag and not a newly read tag) and an additional step of fetching the event data for event processing by the RWS. Operation F shows the boxusage for processing an event with a true trigger with a non-reader subscriber (i.e. subscriber has a URL or e-mail address). It has a value of 0.000145159 and involves all steps of operation D and two additional steps of updating the event expiration column in the event table and searching for any complex event rules in the database involving the successful event. Operation G showing the boxusage of processing a true trigger event message subscribed by a reader has the highest box usage. It has a higher boxusage than operation F as it involves all steps of operation F and an additional step of updating the 'Pending command' column of the subscriber reader in the reader table.

6.2 Tag processing by RWS

Although this architecture works smoothly with the readers and the web service server in our lab setup, we wanted to perform load tests on the service. For this purpose, we created a program in Matlab which can generate and send SOAP packets to simulate reader generated message traffic. In order to test the performance of the system, we generate SOAP messages with the number of

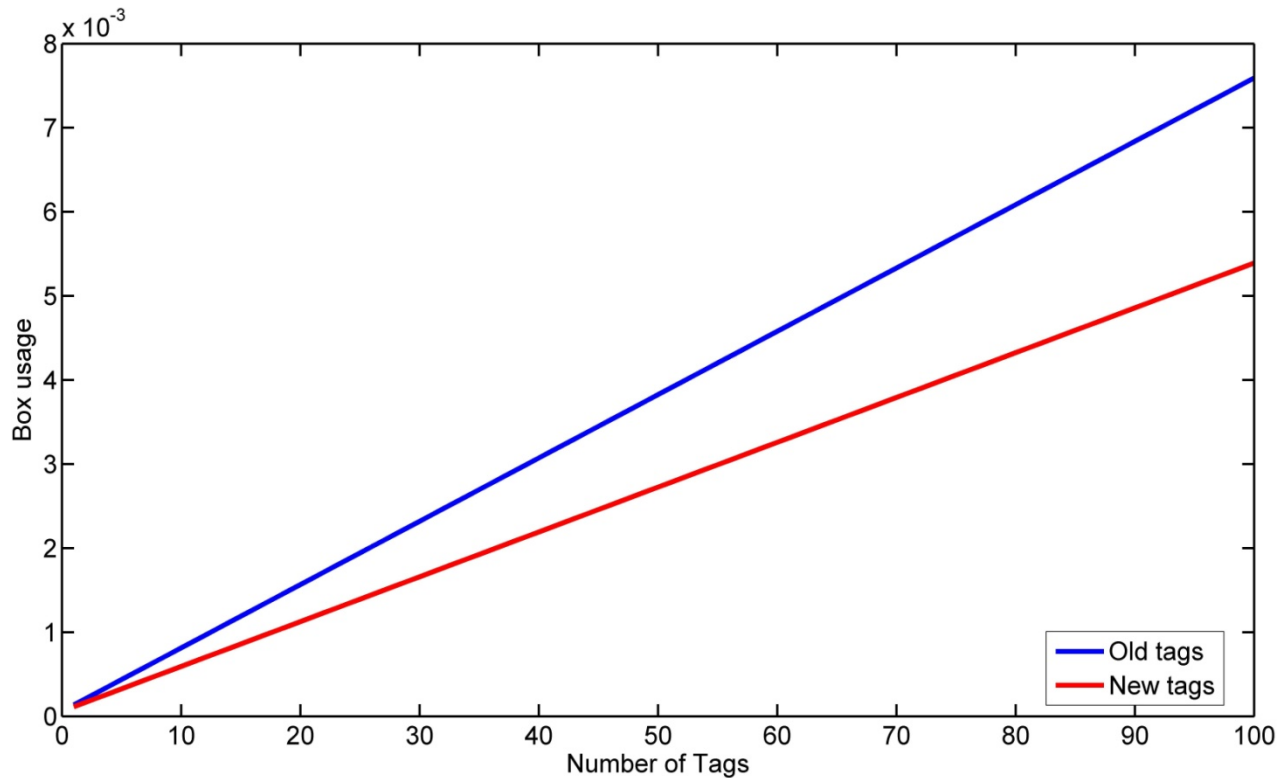


Fig. 6.3: Cloud database boxusage as a function of number of tags

tag IDs per message spanning from 1 to 100. So the smallest message has one tag ID and its corresponding time-stamp in the message, whilst the biggest message has 100 tag IDs and their corresponding time-stamps embedded in the message. The test was performed over a network with a speed of 54 Mbps. Fig. 6.3 shows the boxusage of the cloud database service as the number of tag IDs per message increase. The red line shows the boxusage when the message contains new tags i.e. tags that were read by the reader for the first time. The blue line shows the boxusage when the message contains previously read tags that were read by the reader again. The boxusage for updating previously read tags is high because for previously read tags, the RWS has to update the two columns of ‘last time of read’ and ‘occurrences’ (two update operations), however, for new tags, the RWS has to simply add a new entry (one store operation) on the tag table, the latter having smaller boxusage. The slope of the curve for the old tags is

7.5×10^{-5} which means that every additional previously read tag in the message costs an additional boxusage of 7.5×10^{-5} . Similarly, every additional new tag contained in the received message costs an additional boxusage of 5.3×10^{-5} . Hence, if a message sent by the reader to the RWS contained x new tags and y previously tags, the cost of processing that message in terms of boxusage would be given by B as shown in Eq. 6.1.

$$B = (5.3x + 7.5y) \times 10^{-5} \quad (6.1)$$

Fig. 6.4 shows the time to process the SOAP message from the simulated reader by the RWS in an observed run. The service took 1.078 seconds to process a message containing 1 newly read tag and took 45.13 seconds to process a message containing 100 newly read tags. On an average, RWS consumes 0.447 seconds for every newly read tag added to the message (using best fit).

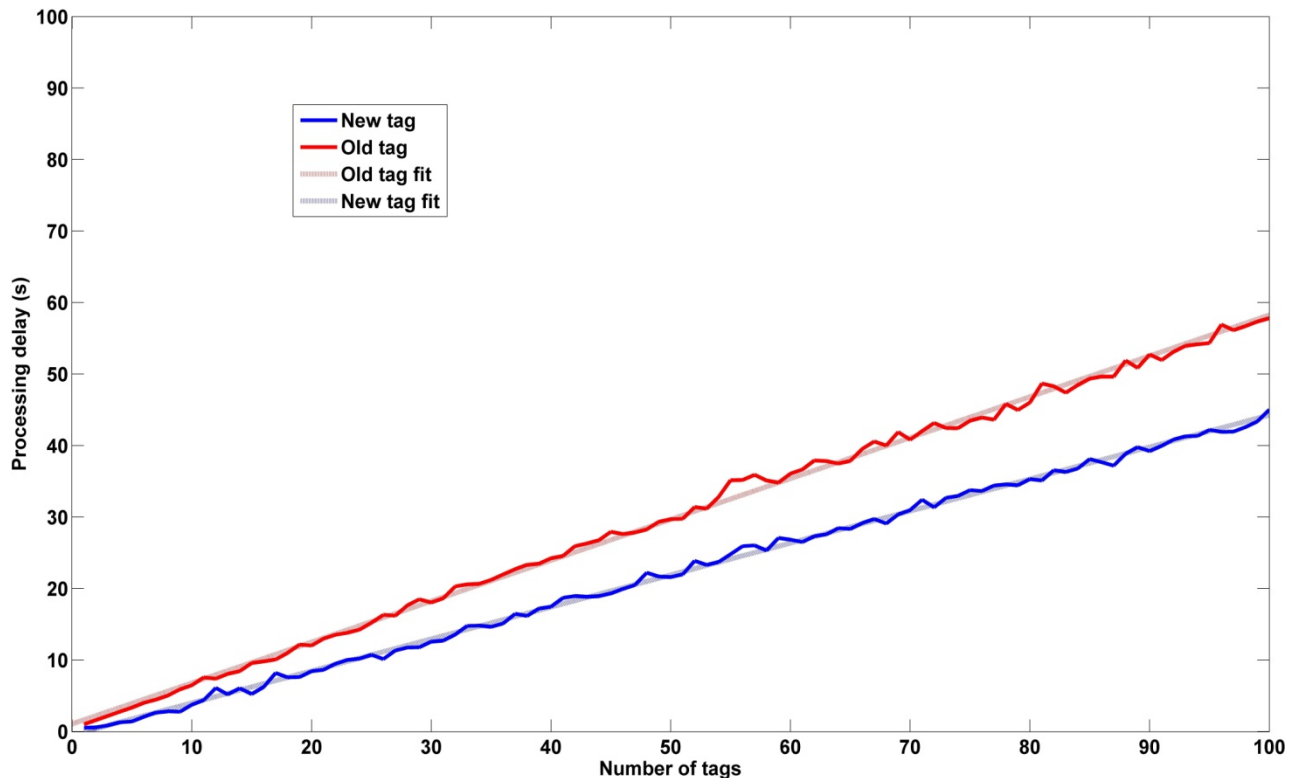


Fig. 6.4: Message processing duration as a function of number of tags

In the case of previously read tags, the corresponding durations were 1.172 seconds and 60.89 seconds respectively. On an average, RWS consumes 0.572 seconds for every previously read tag added to the message (using best fit). These time values represent the values obtained as a mean of 5 identical runs. They may show a small variance in another run. Evidently enough, the longest step involved in this procedure is the store routine on the database. The time measured to process these messages by the RWS is the duration it takes for the tag tables in the cloud database to be updated. However, it takes the RWS a mean of 0.43 seconds (with a variance of 0.0321, sampled over 200 random messages) to receive a message and reply back to a reader. This avoids a bottleneck at the reader and RWS interface, despite a longer message processing delay as the reader is not kept waiting for the RWS to reply back. This is possible due to asynchronous message processing. It must also be realized that the message processing delay of the RWS is also dependent on the internet connection speed as it involves frequent communication between the RWS and the cloud service. However, since the RWS and the cloud service would be provided by optimized service providers, they could provide a minimum guarantee of service quality.

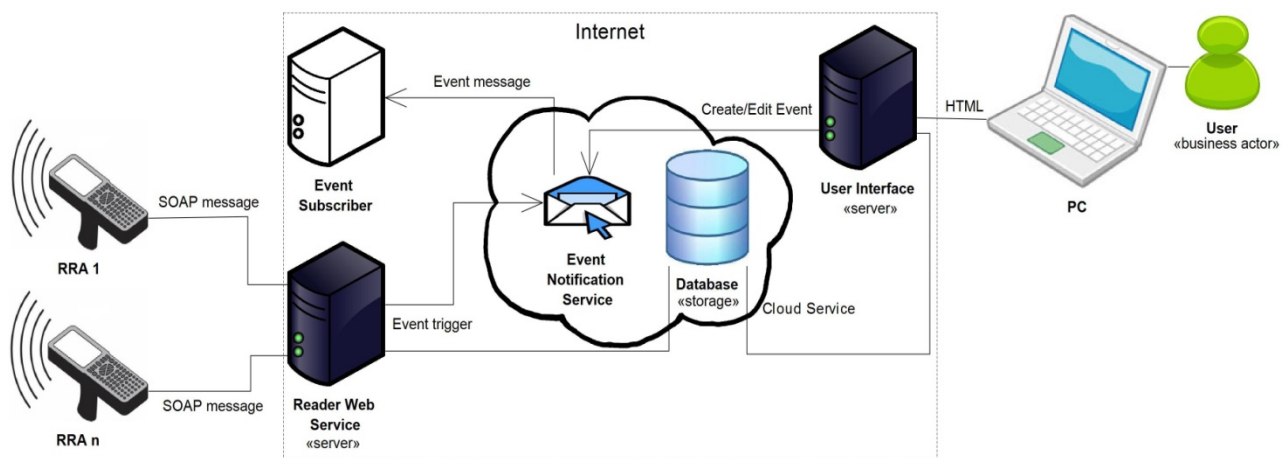


Fig. 6.5: Symbolic system architecture of the RFID web service with a web based server

6.3 Primitive Event Processing

In this part, the performance of the various components during primitive event processing would be analyzed. The application resides on a Motorola MC9090 UHF reader which runs Microsoft Windows Mobile 5 Operating System (OS) on an Intel 624 MegaHertz (MHz) processor. The RWS has been set up on our lab machine which is an Intel 3.07 GHz Quad Core processor with 3.25 GB RAM running Windows XP. Previously, it was discussed, that for processing a primitive event, the RRA compares the tag and its time stamp against the event condition and if true, forwards it to the RWS for further processing. An example is, “If Tag A is reader AFTER Time T then report”. When the reader is reading multiple tags, this event evaluation involves dual comparison i.e. 1) ID based comparison (for identifying the event associated tag ID from the remaining tag IDs) and 2) temporal comparison (comparing tag read timestamp of the associated tag ID with event time span). However, this type of individual event evaluation could slow down a RRA and create a bottleneck. If the RRA performs double comparison for the event, it would be sending the event ID of the successful event to the RWS to simply post the preset event message (corresponding to the event ID) to the preset destination (Complete Primitive Event Processing @ Reader). So, another alternate way of event evaluation is proposed in which, the RRA simply performs an ID based comparison and forwards the tag data packet to the RWS for performing the remaining temporal comparison (Partial Primitive Event Processing @ Reader). The latter technique would free up the RRA’s resources and reduce its processing time and transfer the processing load to the RWS server processor. However, it must be stated here that the reader device is the property of the client, while the cloud service is a service purchased from the service provider. The more the processing performed at the cloud, the greater the running cost is borne by the client so shifting event evaluation task to the RWS also leads to

greater processing done at the server and hence increased running costs. In order to get a detailed understanding of the combined performance of these two approaches, they are discussed in the next part.

First, a test was carried out to measure the processing delay experienced by the RRA as a function of tags and events. In order to test this, the previously explained simulated RFID driver was used to simulate tags from 1 to 1001 in intervals of 50. Similarly, 1 to 1001 events were sent to the RRA in intervals of 50 to report successful events. Fig. 6.6 shows two contour plots representing the processing delay of the RRA residing on the RFID reader as function of number of tags. The processing delay is in seconds. A point on the contour plot with coordinates (x,y) shows the event processing delay (given by the color) when the RRA has x distinct events to track in its events table and read y tags in a single read operation. Fig. 6.6b shows the processing delay for full event processing by RRA (ID based and temporal) while Fig. 6.6a shows the

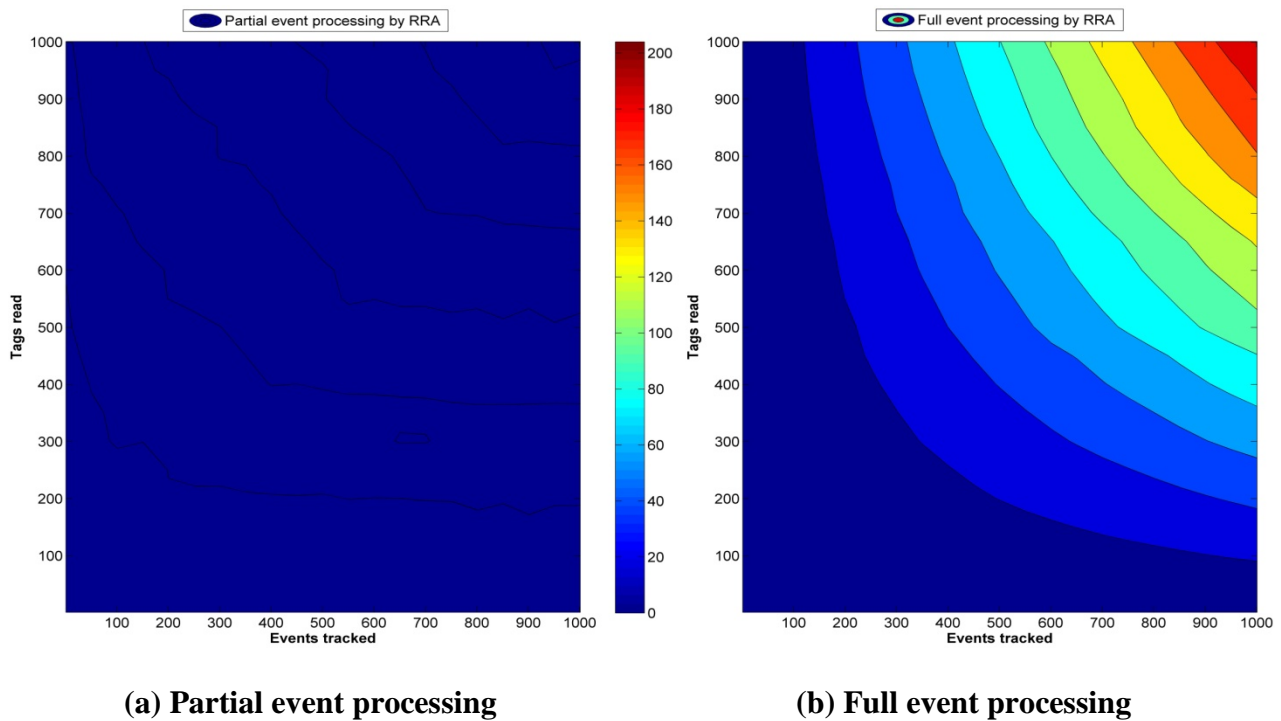


Fig. 6.6: Processing delay of RRA as a function of number of tags and number of events

processing delay for partial event processing (ID based only). In full event processing, the RRA took 1.335 ms to evaluate 1 successful event while reading 1 tag and having 1 event and it took 203.3367 s to evaluate 1001 successful events while reading 1001 tags and having 1001 events. In contrast, for partial event processing, the RRA took 0.667 ms to evaluate 1 successful event while reading 1 tag and having 1 event and it took 1.2629 s to evaluate 1001 successful events while reading 1001 tags and having 1001 events. The data for the contour plot was obtained as a mean of 4 similar tests.

In the next test, the behavior of the RWS was analyzed while evaluating events. In this test, reader generated event messages were sent to the RWS for processing. The RWS performance was then logged to measure the processing time for evaluating the event messages. The test was carried out from 1 event to 1001 events in increments of 50. Again, in this case, two cases were

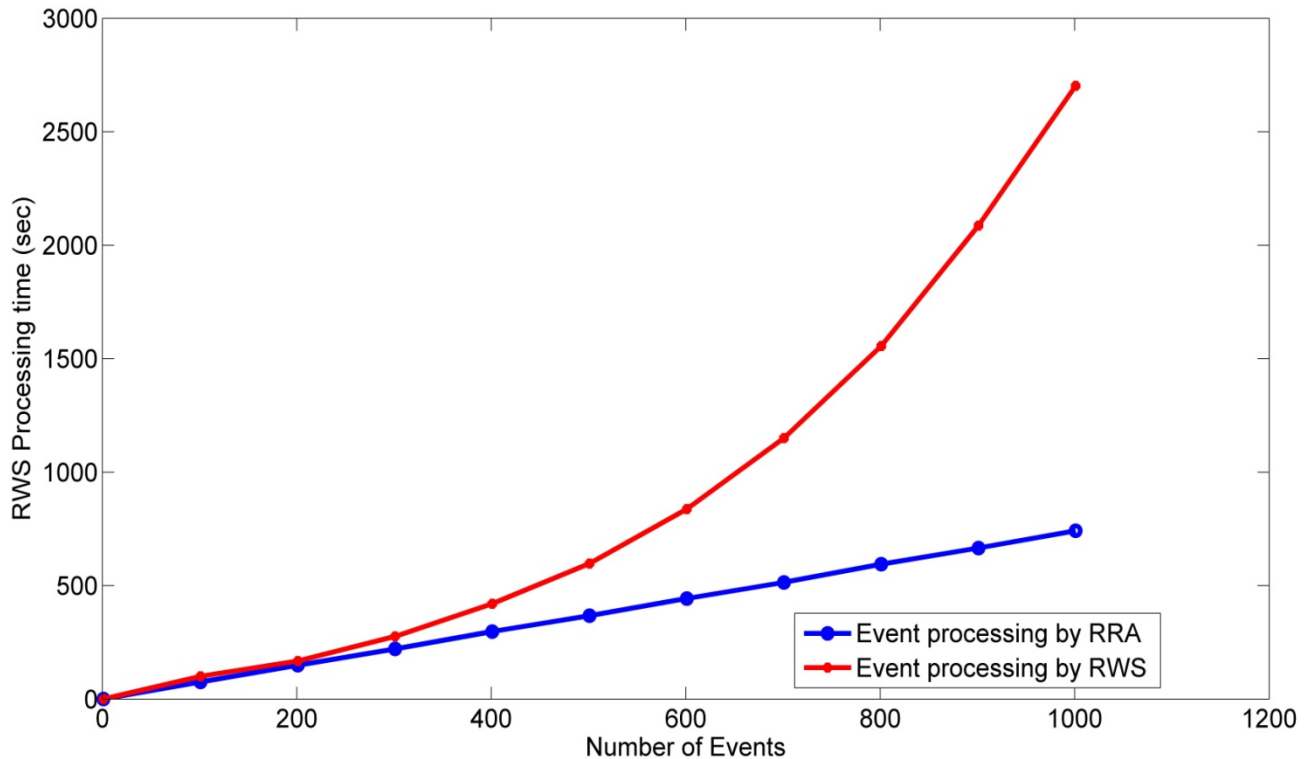


Fig. 6.7: Processing delay of the RWS as a function of number of events

studied. First case involved, full event processing being performed at the RRA. In this case, the RWS fetches the preset event message from the event table in the cloud database and posts it to the cloud notification service and then, updates the former. The average processing delay (averaged over 5 runs) is shown by the red curve in Fig. 6.7. The average processing delay for 1 event is 0.2844 seconds, which increases to 797 seconds for 1001 events.

The box usage for this operation is shown by the red curve in Fig. 6.8. The box usage for 1 event is 3.966×10^{-5} , while for 1001 events, it is 0.06711. The RWS shows an increase of boxusage of 6.7×10^{-5} for every additional event. In the second case, if the RRA performs only ID based comparison (partial event processing) and forwards the tag data to the RWS as an event message, then, the RWS needs to perform temporal comparison for the evaluation of the primitive event. The average processing delay for this operation by the RWS is shown by the blue curve in Fig. 6.7. The average processing delay for 1 event in this case is 2.228 seconds while for 1001 events,

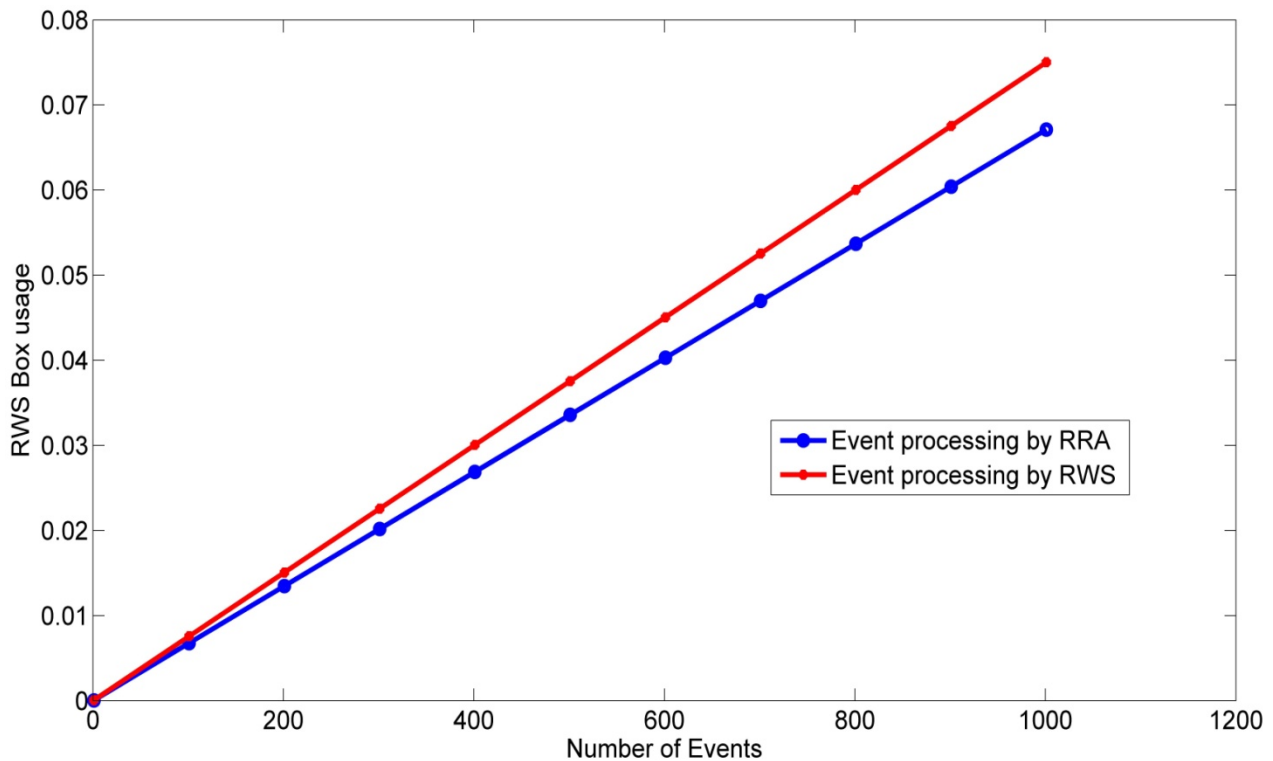


Fig. 6.8: Box usage of the RWS processing events as a function of number of events

it rises to 2740 seconds. The box usage is shown by the blue curve in Fig. 23. The box usage for 1 event is 7.494×10^{-5} , while for 1001 events, it rises to 0.07501. The RWS consumes 7.5×10^{-5} additional box usage for every additional event. Partial primitive event processing by the RRA is 1.12 times more expensive in box usage than full event processing by the RRA. The higher delay compared to the previous case is due to the additional step of fetching the event rule from the event table and performing the temporal comparison for the event.

Fig. 6.9 shows the combined processing delays. After adding, it is apparent from the first look that partial event processing by RRA has more delay than full event processing by RRA. For 1001 events tracked and 1001 tags read, partial event processing by RRA takes a combined duration of 2741.26 seconds, while for the same state, full event processing by RRA takes 1000.33 seconds.

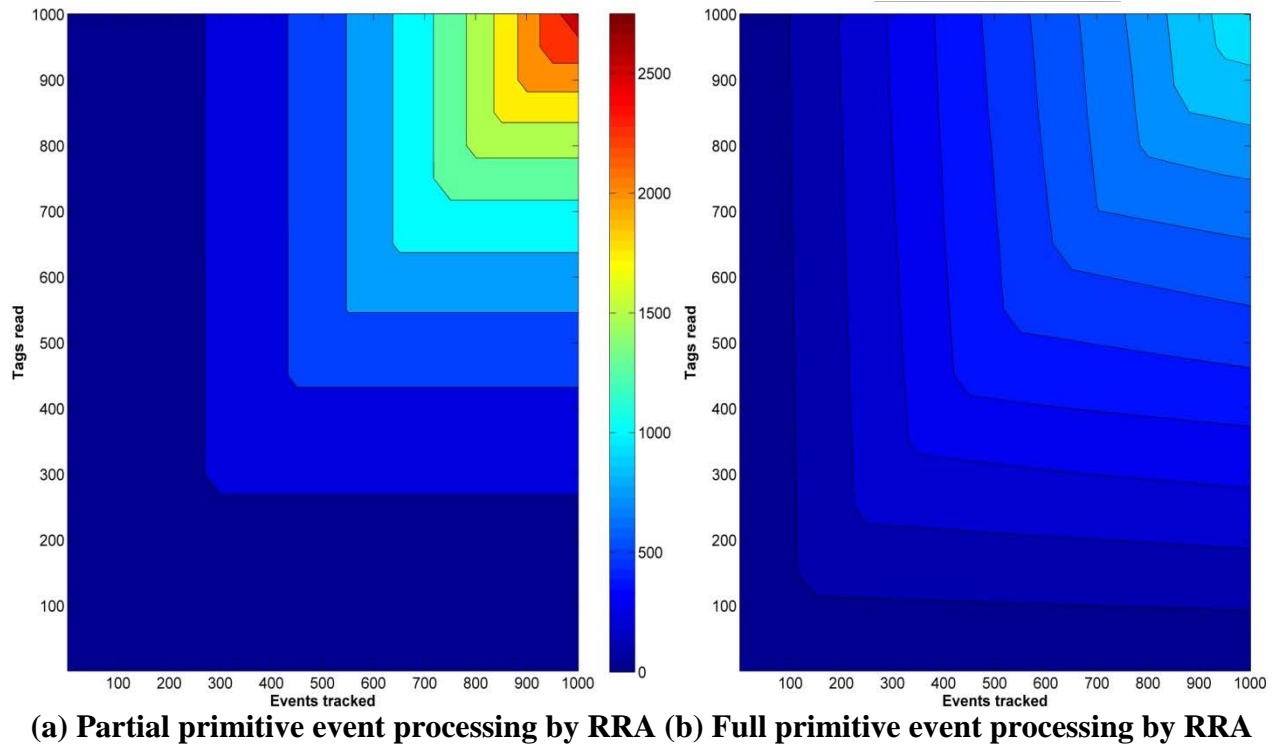


Fig. 6.9: Combined processing delay of RRA & RWS as number of tags and events vary

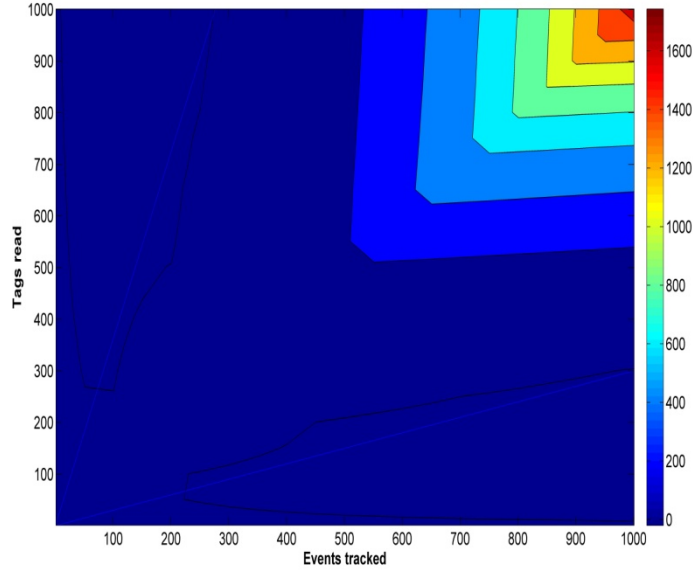


Fig. 6.10: Difference in combined processing delay as number of tags and events vary

However, in order to get a more accurate picture to compare the performance, we subtracted the combined processing delay of full event processing by RRA from the combined processing delay of partial event processing by RRA. This has been shown in Fig. 6.10. Processing delay of full event processing at RRA is almost always smaller than processing delay of partial event processing at RRA, except inside the two triangular shaped contours at the left and bottom of the contour plot. We approximated these zones by the two straight lines. Inside the quadrilateral shaped zone demarcated on either side by the two straight lines, combined processing delay of full event processing by RRA is mostly greater than the combined processing delay of partial event processing by RRA. Let us assume that N_T represents the number of tags read and N_E represents the number of events tracked. The equation of the two straight lines is given by the Eq. (6.2) and (6.3).

$$N_T = 0.3 * N_E \quad (6.2)$$

$$N_T = 3.62 * N_E \quad (6.3)$$

So, if the ratio of NT /NE has a value between (0.3, 3.62), then, full event processing by RRA is favorable. Partial event processing by RRA is favorable otherwise. Through these tests, we can conclude that if the average number of tags read by the reader per event tracked by the reader is within the range, (0.3,3.62), then full event processing at RRA yields shorter processing delays, otherwise partial event processing at RRA is preferred.

A test is carried out to measure the processing delay experienced by the entire system to evaluate primitive events. The delay is measured at two stages: At the RRA and at the RWS. As RRA would perform more computation for Complete Primitive Event Processing compared to Partial Primitive Event Processing, it would have a greater delay for the former; however, the inverse would be true for the RWS. In order to measure the event evaluation delay, the RRA is made to read 1000 tags and the number of events is increased from 1 to 1000 in steps of 100. The events are generated in such a way that on being evaluated, all of them will be true to enable the

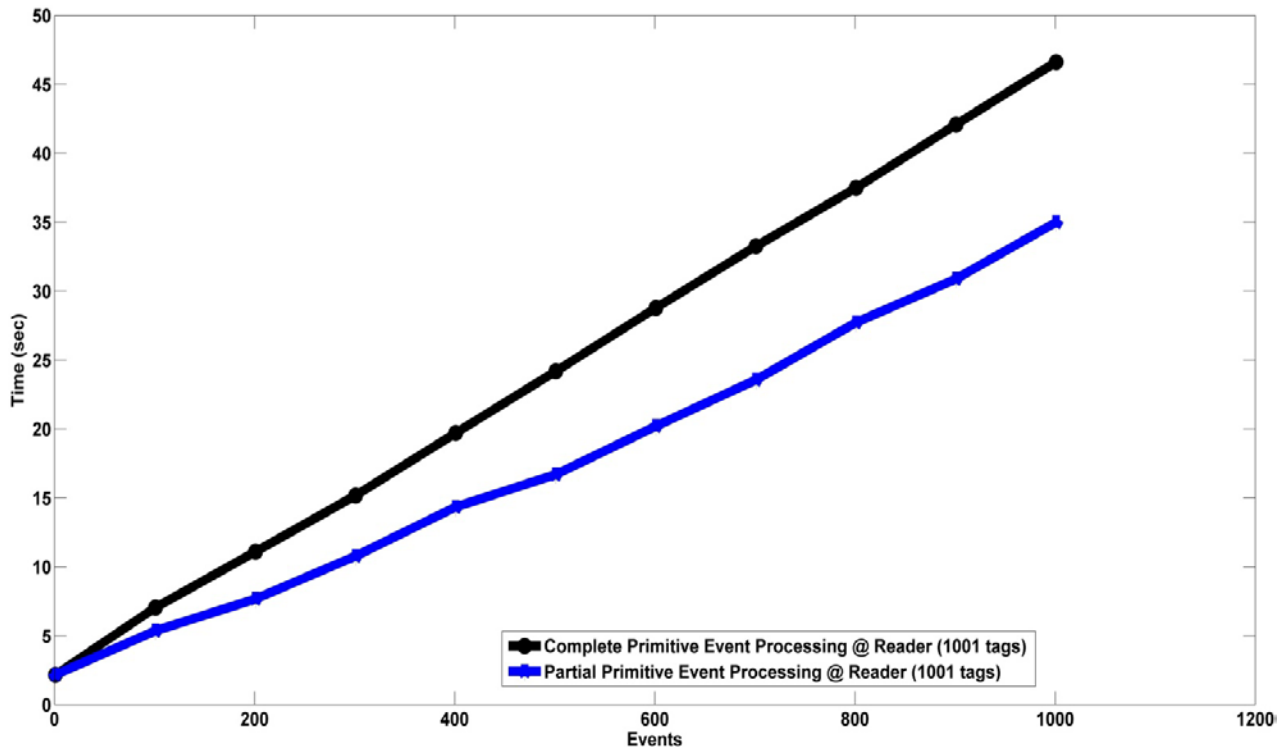


Fig. 6.11: Processing delay for evaluating primitive events

calculation of the complete processing delay only (and not false events which would have shorter processing delays). The results plotted were obtained after averaging the results from 10 identical runs. Fig. 6.11 shows the combined processing delay for the two primitive event processing techniques. The combined delay was obtained after adding the delay measured at the RRA and the RWS. This does not include the network delay as that would be same for both cases and has already been discussed before. In the tests, it was found that performing Partial Primitive Event Processing at the RRA gives a lower processing delay as the number of primitive events evaluated is increased from 1 to 1000. Complete Primitive Event Processing at RRA takes an average of 44.251 milliseconds (approximate from slope of the graph) for every additional event evaluated, while Partial Primitive Event Processing takes 33 additional milliseconds (approximate) for every additional event evaluated.

6.4 Complex Event Processing

In this section, the complex event processing techniques in this architecture are explained. The application resides on a Motorola FX7400 UHF reader which runs Microsoft Windows CE 5.0 Operating System (OS) on an Intel PXA270 processor. An example of complex event is the following: Event A and Event B have been created previously. Event A is associated with Reader R_A while Event B is associated with Reader R_B . A Complex Event AB is created which will be true if Event A AND Event B occur after Time T. A real life example which fits such a complex event could be an RFID tagged shipment is to be notified as arrived at its destination to its shipper only if it passes and gets detected through two separate reader portals sequentially. Although, complex events can involve a combination of several primitive events and time and logical operators, this example would be used as a model to show as proof of concept in this paper because of its relevance to the supply chain system. In the previous section, the two

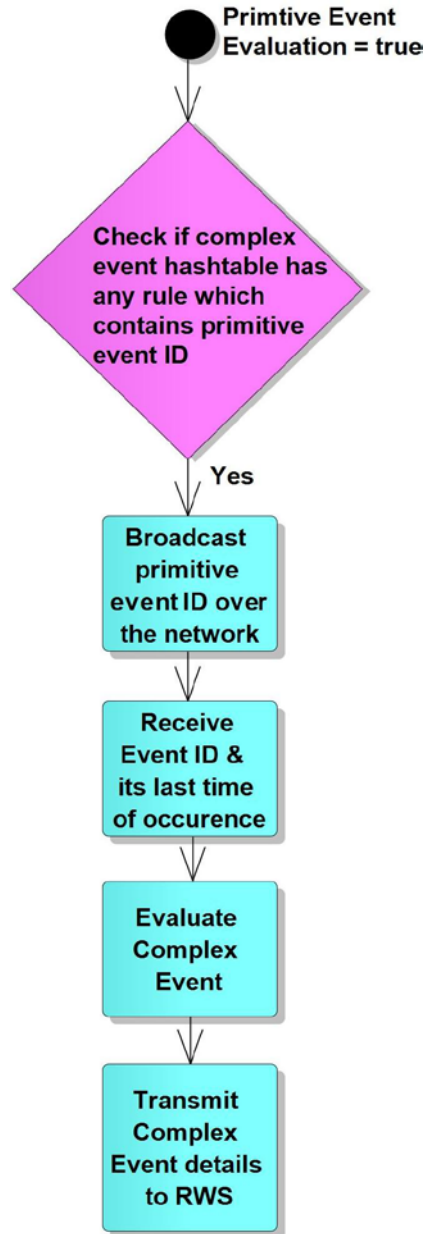


Fig. 6.12: Flowchart for Complex Event Processing by the RRA complex event.

different techniques to evaluate primitive events were discussed. After a primitive event has been evaluated and is found to be true, it can be checked to see if this primitive event is contained in the rule of another complex event.

The search is performed by scanning through a complex event table to search for the primitive event ID in the complex event rule, if found, the complex event rule is parsed and evaluated. In,

the given RFID architecture, three different complex event processing techniques have been identified: 1) The first technique involves partial primitive event processing occurring at the RRA, while the remaining primitive event processing and the consequent complex event processing occurring at the RWS (Scenario 1), 2) The second technique involves complete primitive event processing at the RRA, while complex event processing at the RWS (Scenario 2) and 3) the third technique involves both primitive and complex event processing occurring at the RRA level (Scenario 3). The first and second techniques are similar to the primitive event processing techniques discussed previously except that an additional complex event processing step occurs at the RWS where the RWS checks if a complex event has occurred based on the occurrence of the primitive event. Complex event processing for Scenario 3 takes place entirely at the reader level. This involves multiple readers within a network collaborating with each other to evaluate complex events. The diagram showing the steps of complex event processing have been shown in Fig. 6.12. In order to evaluate complex events, Reader R_A maintains two separate hash-tables which store primitive events and complex events respectively. Besides, the event IDs, these tables also store the event evaluation parameters. As explained previously, the RRA checks the primitive event table for a successful event after reading tags. If a tag read causes Event A's occurrence to be true, then Event A is then further checked through the complex event hash-table to check if any complex event exists which has a condition which contains the event A. If such a complex event is found (Event AB), then the other primitive event involved (Event B) in the complex event is broadcast (User Datagram Protocol) over the network to other readers by the reader to get its details (last time of occurrence). Reader R_B which recently evaluated a true occurrence of Event B replies to the broadcast by giving out the last time of occurrence of

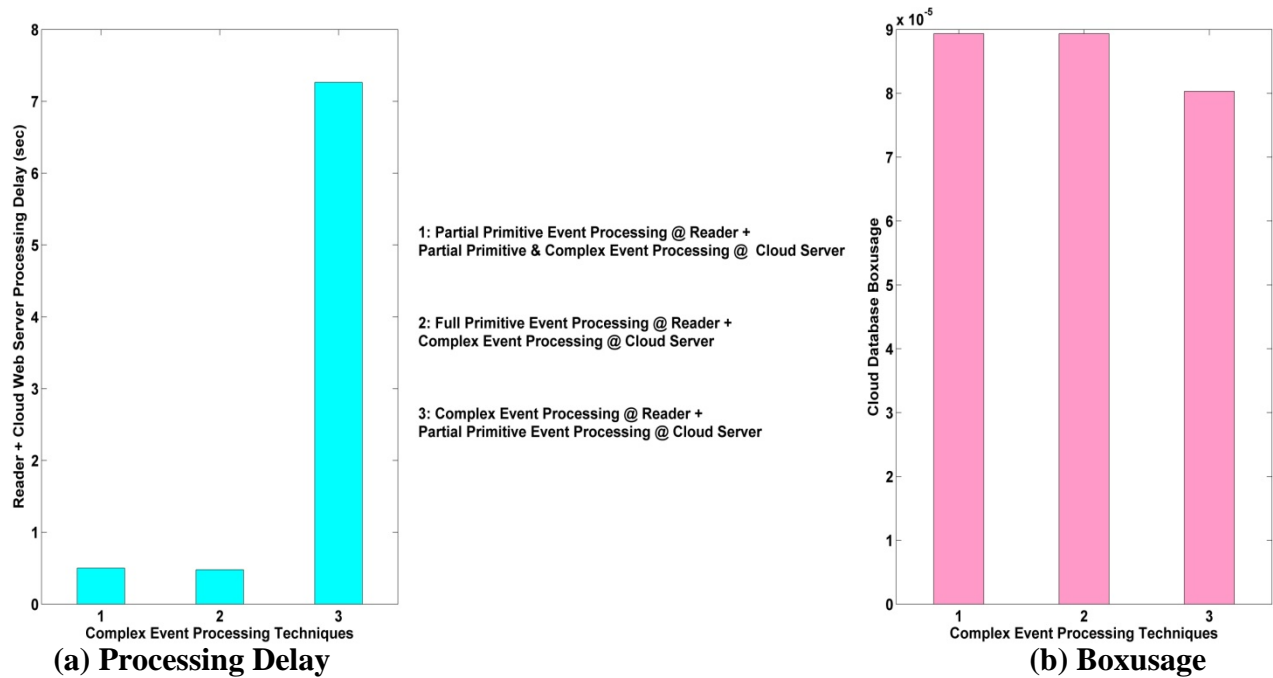


Fig. 6.13: Comparison of different Complex Event Processing Techniques

Scenario #	Avg. RRA Delay	Avg. RWS Delay	Avg. Total Delay	Avg. DB Boxusage
Scenario 1	2 ms	0.5005 s	0.5025 s	8.934×10^{-5}
Scenario 2	3.5 ms	0.4764 s	0.4799 s	8.934×10^{-5}
Scenario 3	14.3889 ms	0.3422 s	0.3565 s	8.0315×10^{-5}

Table 6.1: Comparison of algorithms for Complex Event Processing for 1 event

Event B. This information is again received by Reader R_A and used to evaluate the condition of Event AB. If true, then Reader R_A sends the occurrence report to the RWS.

In Table 6.1, the processing delay of the RRA, the RWS and the boxusage of the RWS (the processing cost incurred by the operation of the cloud database) have been evaluated for the three approaches for a single complex event evaluation. The values listed in the table are the

average of 10 identical runs. The events were generated in a way that both the primitive events and the complex event are evaluated as true to obtain the processing delay of the whole process. A single complex event evaluation occurs, the fastest for Scenario 3 at 0.3565 seconds. The box usage is also the least for Scenario 3 as it involves the least processing at the RWS stage. However, for 1000 events, Scenario 1 has the least processing delay at 45.6855 seconds (Table 6.2). In order to get a better understanding of the processing delays for Scenario 1 and 3, the complex events to be evaluated by the RRA are increased from 1 to 1000 in steps of 100 and combined processing delay by the two Scenarios are measured. Fig. 6.14 shows that Scenario 3 is the fastest for events numbering less than ~400. This result was compared with the work of Wei et al. who have evaluated the traditional middleware based approach to process complex events [55]. Their architecture also consisted of a Microsoft IIS Web Service, to accept data streams, Microsoft BizTalk middleware for messaging and orchestration and a Microsoft SQL Server for data storage. In their work, processing 1000 complex events took a total delay of 66 seconds, which is higher than the total processing delay it took for Scenario 1 and Scenario 2. Although, this cannot be used as a conclusive metric to claim the better performance of the architecture discussed in this paper, but it does demonstrate that a Cloud based architecture can perform tasks traditionally performed by middlewares based architectures with comparative

Scenario #	Avg. RRA Delay	Avg. RWS Delay	Avg. Total Delay	Avg. DB Boxusage
Scenario 1	23.195 s	22.4905 s	45.6855 s	0.1036
Scenario 2	37.57 s	14.0931 s	51.6631 s	0.1036
Scenario 3	73.873 s	11.5707 s	85.4437 s	0.0804

TABLE 6.2: Comparison of algorithms for Complex Event Processing for 1001 events

performance.

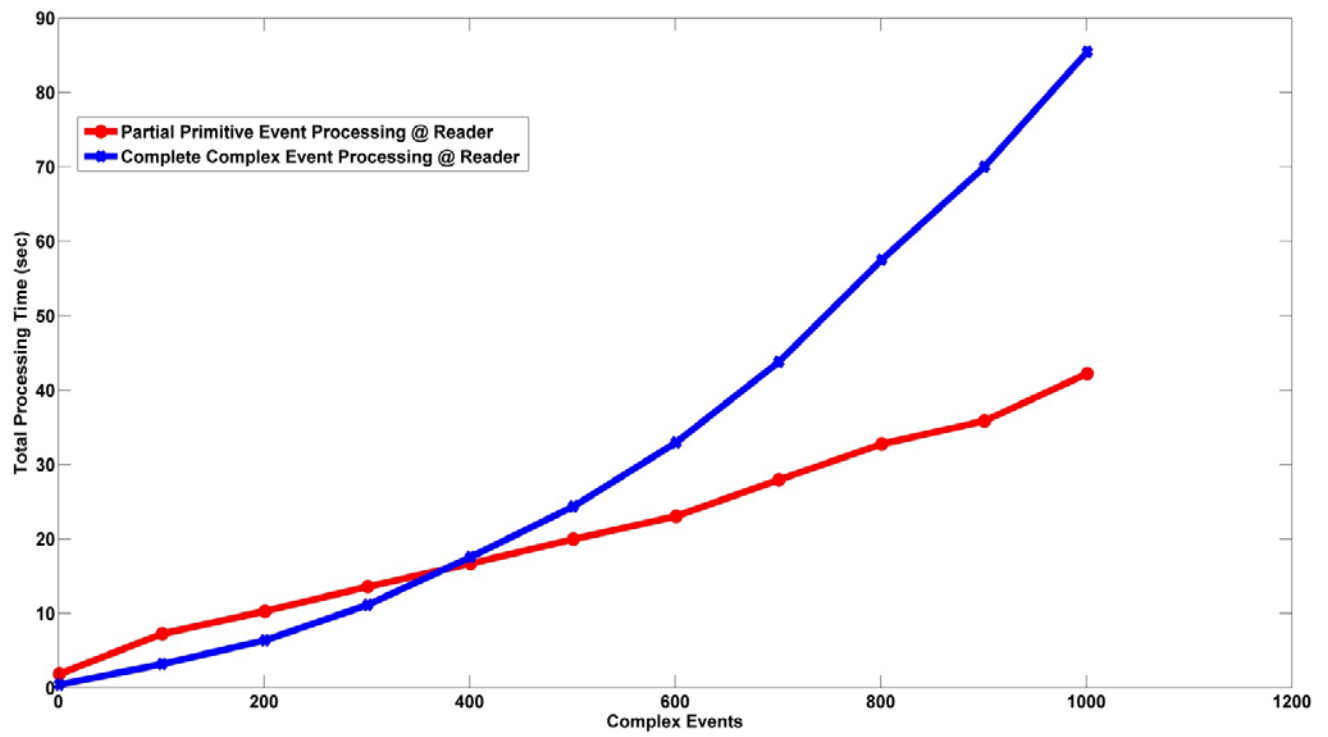


Fig. 6.14: Comparison of Complex Event Processing delay

Chapter 7

Discussion

In this section, potential issues relating to the implementation of this architecture are discussed that might arise and other concerns that potential future adopters might have. Further, corresponding measures are explained that have been implemented and can be adopted in the future to mitigate these issues.

7.1 Data Security and Privacy

Many organizations could have concerns over allowing their data to be transported and/or stored by a third party cloud service provider. Further, different employees of the same/different organization could also tamper with the data without appropriate permission using the Web based UI. In this part, steps to address these concerns are explained. In order to ensure secure communication between the RRA and the RWS, Advanced Encryption Standard (AES), a leading form of symmetric encryption is used. In symmetric encryption, the client and server share a key which is used for encryption and decryption of messages. Since the initial key transmission between the RWS and RRA can become a security threat if intercepted, it has been designed in a way where the RRA generates an initial encryption key which the user inputs into the Web based UI when initially registering that particular reader. In an organization, this deployment can also be done systematically since the reader stores the encryption key in a file allowing the organization to replace the encryption key on the reader with one they generate themselves when deploying readers. At this point the RWS knows which encryption key is associated with each reader. As a result all communication can be done securely. When sending messages to the RWS, the RRA uses a unique identifier in clear text (MAC address as mentioned before) followed by its messages in encrypted form. Upon receiving the message, the RWS can

identify the reader by its identifier and choose the appropriate security key to decrypt the message. It then processes the message and encrypts its acknowledgment reply using the appropriate key for that reader. The reply is then sent back to the reader in encrypted form.

Another area of security is the initial use of HTTP (Hyper Text Transfer Protocol) through the Web based UI that is used by the individual end-user or administrator. The HTTP protocol sends messages in clear text allowing a spoofer to intercept these messages and identify raw tag information. In order to solve this problem, HTTPS (HTTP Secure) has been used on the Web based UI. As a result, all information transported between the end-user and the UI is done in an encrypted manner using the HTTPS standard which is already support by both browsers and web servers across the board. The first prototype of the UI is run using a self-signed certificate to encrypt HTTPS messages. In a production system it would function in a similar manner, except, instead of using a self-signed certificate the UI server would use a certificate signed by a 3rd party such as VeriSign. This adds an extra level of security to the end-user because in addition to using HTTPS for secure communication, there is now also an authority to verify the UIs identity by providing a signed key. The User administration page on the User Interface also allows an administrator to control the amount of access different users have to the entire system. As an example, some users after logging into the UI can only visualize tag data, while other users with higher powers can perform actions like add/delete tag information, issue commands to readers, add new readers to the supply chain etc.

7.2 Network/Connectivity Failure

The RWS server overloads and network interruptions can be a problem for RRAs that rely on the cloud for immediate offloading and processing of data. In the initial prototype type of the system, this issue became clear when the server would become overloaded and crash leading to a

domino effect where the RFID client would then be unable to send future messages to the server and crash. As a result, steps have been taken to ensure the system is not only optimized and highly available, but can also deal with outages of servers or the network in the worst case scenario. In order to accomplish this, the RRA was designed to deal with unresponsive RWS or network outages. In normal operation, the RRA attempts to send messages to the server at a given interval, say every 10 seconds, flushing its record of all tags it has read up until that point. The RRA does this at a fixed period to provide the benefit of batching RFID reads to minimize network overhead while using a reasonably small amount of memory to store all of the reads from the past period. In order to support network outages, if the RRA attempts to send the last read tags message to the RWS and either fails to connect to the RWS or gets no response, it stores those messages to its local file system and sets a flag in memory to indicate that there are unsent messages. On the next iteration it detects that there are pending messages and sends those pending messages first followed by its current set of messages. If it fails again it continues to store the additional messages to the file system. Eventually the RWS comes back online and the client is able to send its queue of messages at which point it flushes the pending messages from disk. Similarly, if an RRA has not been in contact with RWS for a long time due to network outage or after waking up from sleep mode, it can request an initialization command to the RWS, where the RWS retrieves all the commands and events previously generated for that particular reader and send it back for the reader to commence operations with the pending user requested command.

7.3 Large data traffic from multiple readers

In order to prevent overload of the RWS server as a result of numerous readers sending long batches of messages following a network outage, the RWS has been coded in a defensive manner

to protect itself from crashing in this scenario. When the RWS receives a message to be processed, it spawns several threads to process each portion of the request. If at any point it fails to spawn a thread, it briefly pauses to allow for other threads to finish their process and then retries. In a worst case scenario, the RWS is so inundated with processing some of the requests that it fails to accept requests from other readers. At this point, the defensive code on the RRA allows it to postpone sending its messages until the next iteration as discussed in the previous section. Eventually the RWS catches up with all of the readers and the system is back in its normal state. Modern day computing is so fast that it should not take very long for the RWS to catch up with all of its readers. Further the design of the system lends itself for well to scalability. No information is ever stored on the RWS server and no state information is used. The reader stores all of its tag information until it receives acknowledgment that the RWS successfully wrote to the database. Given this design, there could be an unlimited number of RWS servers handling requests from readers. Since it does not matter which reader talks to which RWS server, readers could send requests to a load balancer which then forwards the request to an available RWS server. Using round-robin DNS entries, there could be multiple load balancers and in the scenario that one them is down the reader will simply resend the message to a different load balancer on the next request. Similarly, the use of a distributed database on the back-end allows for unlimited growth and scalability of the database that backs the various RWS servers.

7.4 Customization and Potential Applications

In this work, web based architecture was presented, that would be suitable to warehouse based applications. However, it is likely that many applications would have more customized needs according to their varied domains of operations. As an example, a lot of RFID operations involve

sensor tags where besides, tag ID and time of occurrence, other physical parameters are also reported, such as temperature. However, to enable the RRA-RWS combination to include temperature data in the periodic message reporting cycle, a very minor change needs to be done in the underlying code, such that the temperature data is appended to the message containing the tag ID and date-time. Such customizable options can be offered easily to the client if this service is offered commercially. Further, in the Web based UI, the user can see the various details of the tags read by the readers, however, some clients would prefer to download some of this data to their own applications, instead of simply viewing them on a browser interface. Again, it is not challenging for clients to develop their own algorithms using the cloud service provider SDK (in this case Amazon Web Service SDK) to download data from the cloud based database. Finally, in this architecture, the procedure to create and evaluate events was demonstrated, by selecting tags from the list, followed by specifying the time range within which the event should occur etc. Although we demonstrated complex events from the point of view of the reader, the tag and its time of occurrence, but this can be easily translated into something more relevant to the end user such as location data (translated to reader), object description (translated to tag) and time of occurrence using a database of pointers. Some critics could argue that there might be more user-friendly ways to create events for RFID readers, such as drag and drop based GUIs. This kind of an event creation GUI was also shown by [41]. However, modifying the present interface to integrate a more aesthetic user interface is not challenging from a research standpoint and hence has not been implemented as the goal was to demonstrate that complex events creation and evaluation can be performed in such an architecture in real time with a high fidelity to the users without requiring direct connection between the users and the RFID readers. Also, implementation and evaluation of Complex Event Processing was demonstrated. Although, a

complex event comprising of only two primitive events was shown in this paper, complex events can get more convoluted and can comprise of several primitive events, however, the underlying functionality to process such events would simply involve additional iterations of the basic evaluation mechanism discussed in this paper.

Chapter 8

Implementation of an RFID enabled Electric Vehicle Parking Lot

One million electric vehicles (EVs) and plug-in hybrid electric vehicles (PHEVs) are expected to be in use by individuals and fleets by 2015 [56]. Unmanaged EV charging will add to peak grid load and would require additional generation capacity [57], [58]. Charging must be scheduled intelligently in order to avoid overloading the grid at peak hours and to take advantage of off-peak charging benefits. EVs can also serve as an energy resource through vehicle-to-grid (V2G) operation by sending electricity back into the grid thereby preventing or postponing load shedding [59], [60]. Charging and V2G services must be optimized for grid load while guaranteeing owner schedule and range requirements are met.

A system leveraging mobile devices and application to facilitate “smart” charging has been proposed [61]; however integration of the mobile component with a charge scheduling component is not specified. In addition, the described system does not account for discrepancies between specified user charge profiles and actual distance traveled and times of arrival and departure. A conceptual framework for V2G implementation has been developed [62], however EV owner input into the system has not been considered. The market penetration of smart technologies and Advanced Metering Infrastructure (AMI) has resulted in smarter EV charging techniques which minimize charging cost to the consumer and grid load at peak hours. Shao et al. [63] proposed a quick charging method where a higher load (240V, 30A) is drawn to enable quicker charging during evening (after 6 p.m.) and off-peak hours. Home-based and off-peak charging (9pm to 11 am) is also considered by Yu [64].

In this paper we present a system that performs electricity price optimized scheduled charging and V2G operation of EVs in a parking garage using owner charge profiles - charge scheduling optimized for electricity price is implicitly optimized for electricity demand. A literature review of RFID, V2G, and charge scheduling is conducted in Section 8.1. In Section 8.2 the system architecture and its individual components are described. The charge scheduling algorithm is detailed in Section 8.3 and results are presented in Section 8.4. Concluding remarks are in Section 8.5.

8.1 Literature Review

8.1.1) Radio-Frequency Identification (RFID): RFID technology has been used for some time in parking lots in the form of Near Field Communication (NFC). The typical role of NFC in parking lots is to grant entry to authorized users. Our system would employ an RFID reader at every access gate to read an entering vehicle's tag. Once the tag's ID has been read, it is transmitted to the system middleware which performs a database lookup. The middleware will either grant the vehicle access and assign it to a parking spot or deny access.

Porter *et al.* discuss the implementation of RFID based vehicle mileage logger at gas stations [65]. They discussed rates of getting successful reads by the RFID mileage loggers from devices placed in vehicles and the issues that affect the read rates of the RFID readers. They also discussed an architecture where a middleware bills the drivers of the vehicles with the toll based on the distance they drove on toll highways after collecting the data through the RFID readers. They also implemented GPS based technologies to supplement the mileage collection process. Theo *et al.* discuss the concept of an Energy Name Service (ENS) for the smart grid energy infrastructure [66]. The concept they have explained is inspired by the principle of Domain Name System (DNS) and Object Name System (ONS) which is very closely related with the

concept of internet of things using RFID tagged objects. The philosophy of their work is that how every entity in the energy domain (like charging stations, vehicles etc) would be given an ID, which would be helpful in identification and hence execution of seamless transactions for charge consumption between multiple geographical entities. This would be similar to the mobile phone service where a user would be billed for their usage regardless of the location of usage of the mobile phone service. They also introduced the concept of Internet of Energy similar to the internet of things where every object related to the energy food chain is interconnected to each other through the internet. Their paper discussed the architecture of such an energy network and methodology for the energy naming service where an object is given an energy ID based on certain parameters such as geographical location, type of object etc. Song discusses the implementation of a home electricity management box installed at every household which acts as an intelligent node to optimize the consumption of electric energy [67]. They propose using the RFID enabled car keys as instruments to authenticate EVs for charging.

8.1.2) Charge Scheduling: Soares et al. have proposed a Particle swarm optimization (PSO) based approach to perform V2G based charge scheduling [68]. Their charging plan also encompasses distributed power generation systems such as fuel cell, solar etc which contribute to the net power generation. Their goal is to minimize the generation cost which includes generation production cost and V2G discharge payment. They compare their optimization approach with General Algebraic Modeling System (GAMS) based optimization software. However, they don't explain the technique employed by the GAMS software to optimize the charge scheduling which makes it challenging to assess the two techniques. In the final result they show that the GAMS based software results in lower total costs but the PSO based approach has a lower execution time. However, this claim is also hard to validate as very little light has

been thrown on the technique employed by the GAMS optimization software. Venayagamoorthy et al. proposed a real-time digital simulator based vehicle parking lot performing V2G transactions [69]. They have used a binary particle swarm optimization technique to control the power flow to and from a vehicle. The goal of their optimization technique is also to minimize the cost of charging a given EV. In this paper, they analyze the effects of large bidirectional power surges to the batteries and the inverters of individual plug-in vehicles as they are switched from a state of charging to discharging. They concluded that grid faults can be detrimental to vehicles performing V2G transactions unless advanced control and protection is provided. Hutson et al. extended the work by Venayagamoorthy on V2G based charge scheduling in parking lots [70]. The charge optimization was again based on the BPSO algorithm previously discussed. They considered two cases. The first case study takes the price curve and finds the best selling price for each vehicle over the desired Departure State of Charge (SOC) and the best buying price for each vehicle under the desired departure SoC. Case 2 allows for multiple transactions to occur for each vehicle throughout the day leading to more profit oriented charging. However, the Binary PSO (BPSO) algorithm being stochastic, the same solution is not found each time leading to a standard deviation of 0.045% of the average. Wu et al. introduced the concept of intragrid which aggregate the contributing EVs under an umbrella to act as a single unit feeding and taking energy from the grid [71]. They considered different scenarios where either the cost of charging the EVs or the emission from the charging or both can be minimized. The optimization algorithm used is Particle swarm optimization. They achieved their optimum solutions after 1000 iterations. Saber et al. propose the concept of an intelligent Unit commitment (UC) by using V2G to reduce operational costs and emissions [72]. BPSO was used to perform scheduling of thermal units while Balanced PSO was used to determine the number of

V2G units to be connected to grid to minimize the cost of operation and emission by the thermal units. Different scenarios were simulated by optimizing the cost, emission and combined goals. Schieffer et al. have proposed a decentralized charging strategy in their work [73]. First they employ linear programming to optimize charging duration by minimizing charging events during peak load (or high priced) hours. On completion of this step they use probability density functions indicating the distribution of charging slots to determine the exact time choices for charging. They have many similarities with our work such as charge time optimization, division of entire day into smaller charging/discharging time intervals etc which would be highlighted in greater detail during algorithm discussion.

8.1.3) Vehicle-to-Grid (V2G): Guille and Gross [60] present a conceptual framework for implementation of V2G based on bi-directional energy transfer between vehicle and grid and aggregated use of EVs as generation and storage devices. Aggregated EVs can provide grid services such as up and down regulation, load leveling, and peak shaving more economically and with less environmental impact than current systems. EVs must be aggregated because individually their battery capacity is small and would not make an appreciable difference at the grid level. In contrast with unmanaged EV charging that can add to reserve and regulation requirements, aggregated EVs can be charged during off-peak periods thereby leveling grid load and reducing these requirements. The proposed framework emphasizes the Aggregator as the enabling entity for V2G. The Aggregator has the following communications relationships: 1) each EV sends status data and receives charging control signals 2) the Energy Service Provider (ESP) receives charging and sends load levelization requirements 3) the independent system operator or regional transmission organization ISO/RTO sends resource requirements and receives resource availability data. This work presents seminal ideas for future V2G work,

however specific components of the proposed framework are neglected including EV owner input into the system and charge scheduling for random arrival and departure times.

Kempton and Tomic [59] present profit calculations for using EV fleets for V2G services including up and down regulation. Assuming an availability of 17 hours per day for 252 RAV4 EVs, an initial state-of-charge of 30-50%, a required range of 36 miles, and circuits able to handle 6.6 kW, profits range from \$144,800-912,000 for regulation down/up market prices from 12.9/14.0-39.7/62.5 US\$/MW-h. When a 15 kW limit is considered the range is \$358,000-2,102,000. The results include equipment costs and battery life degradation.

Han et al. [74] present a method for optimally controlling EV charging to maximize EV regulation service revenue. The model developed accounts for varying electricity and regulation price over time, a variable time the vehicle will be parked, a variable amount of charge needed, and a maximum charge rate. A maximum revenue of \$.42 is determined, assuming a 20 kWh battery, maximum charge/discharge rate of 2 kW, an arrival time of 00:00, and departure time of

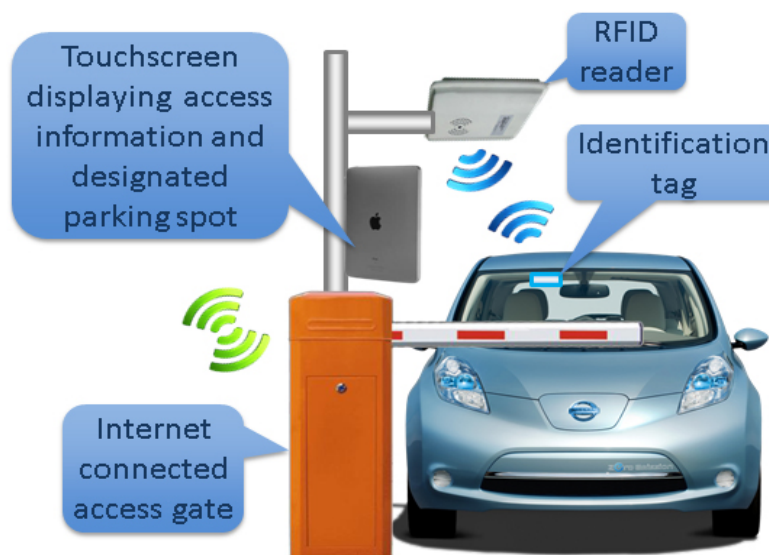


Fig. 8.1 Parking garage access gate

12:00.

8.2 System Architecture

8.2.1) Overview: EV owners will register one or more vehicles with the system. Each registered vehicle will wear an RFID access tag that will allow it to enter enterprise owned and affiliated garages. The tag's unique ID is used by the Parking Garage Aggregation Middleware (PGAM) to lookup the associated owner and act on his/her charging profile and billing information. At the access gate (Fig. 8.1) the driver will be designated a numbered parking spot based on availability. Once, the owner plugs in to the EV Supply Equipment (EVSE), the PGAM checks for an existing charging profile. If none is found he/she will be prompted to enter one via the EV Command Portal (EVCP) application on their mobile device or charge station touchscreen (Fig. 8.2). Fig. 8.3 shows the sequence of events when a car arrives at the gate. If the user fails to enter a profile within a given time window, the system uses a default. The EV owner can

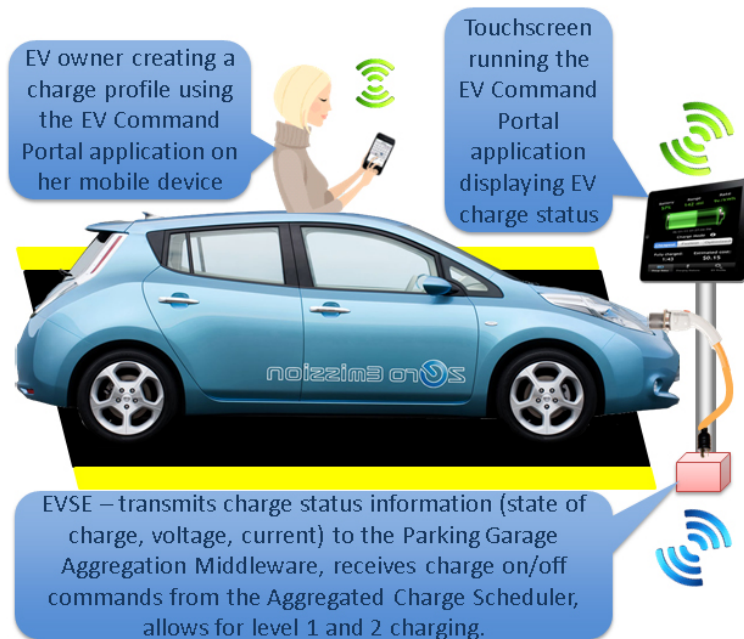


Fig. 8.2 Charging station equipment and activities

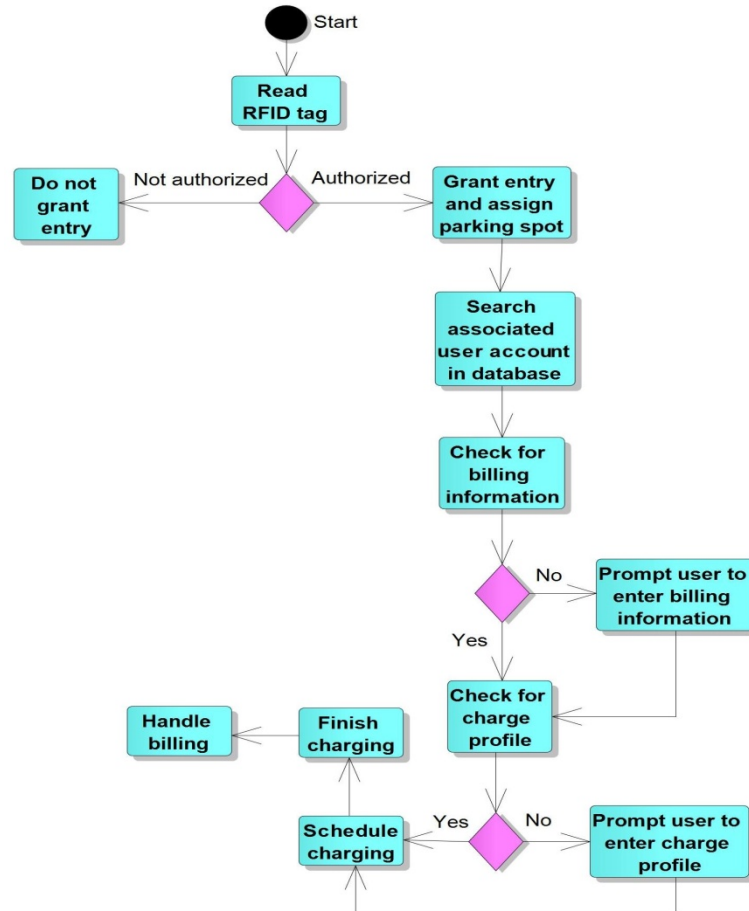


Fig. 8.3 Event sequence for EV arrival at parking garage

use the EVCP to monitor the status and control the charging modality of his/her vehicle within system parameters at any time. Based on user charge profile the Aggregated Charge Scheduler (ACS) will schedule charging to meet user charge requirements, minimize cost, and maximize profit from V2G services. Fig. 8.4 shows the flow of the aforementioned data among system components. Range anxiety is one of the main obstacles to consumer adoption of EVs. The problem stems from 1) limited range compared to conventional gasoline vehicles 2) inadequate charging infrastructure [75].

8.3 Aggregated Charge Scheduler

8.3.1) Overview: The Aggregated Charge Scheduler (ACS) optimizes EV charge scheduling for minimal cost using user charge profiles and electricity price as a function of time. By optimizing charge scheduling for electricity price it is implicitly optimized for electricity demand. The ACS sends a control signal to each EVSE to charge, discharge, or turn off according to the created schedule.

The ACS receives an owner charge profile from the PGAM which includes time of arrival (t_{arr}), time of departure (t_{dep}), buffer time, T_{buff} , initial state of charge ($ISOC$), and final state of charge ($FSOC$) of their EV. Charging must be completed T_{buff} minutes before the owner's scheduled departure. The value of T_{buff} is calculated by the PGAM based on owner adherence to his/her specified departure schedule. Once the ACS receives a charge profile its task is to charge the EV from $ISOC$ to $FSOC$ within the time span of t_{arr} and $t_{dep} - T_{buff}$.

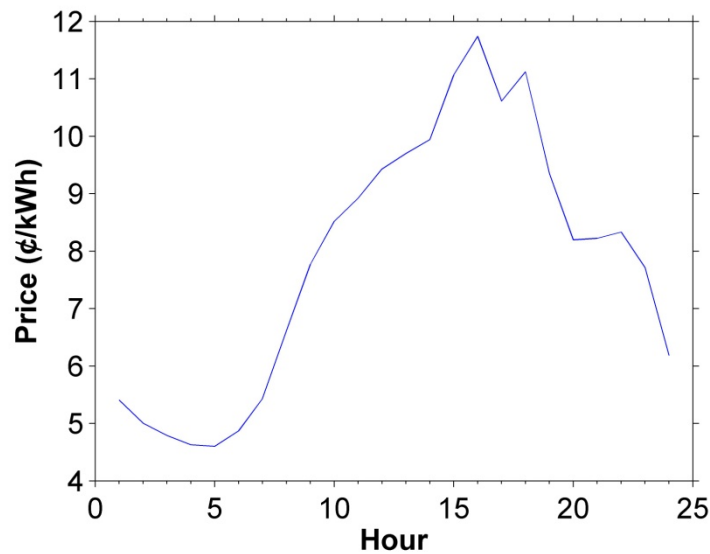


Fig. 8.4 Electricity price [76]

8.3.2) Algorithm Description:

1) Charge Scheduling:

The 24 hour period of a day is quantized into smaller time intervals which are the smallest units of time used by the algorithm to schedule the occurrence of a charge, discharge or no-activity.

The electricity price curve (Fig. 8.4) is also quantized into time intervals where each interval has a fixed price, P_i . Using the charge profile for the given EV and the *ISOC* and *FSOC*, the charge duration, T_{ch} , is calculated. In order to calculate T_{ch} , the charge profile data of lithium-ion batteries of the Nissan Altra from Madrid *et al.* [76] is used. If T_{ch} is greater than the duration of the stay of the car (8.1), the algorithm rejects the plan and notifies the PGAM to inform the user of the impossibility of charge completion within the supplied parameters.

$$T_{ch} \leq t_{dep} - T_{buff} - t_{arr} \quad (8.1)$$

Otherwise the algorithm calculates the number of charge intervals required for charging, N_C , from T_{ch} . The algorithm chooses N_C charge intervals (C_i) with the lowest P_j in the interval $(t_{arr}, t_{dep} - T_{buff})$ to charge the EV.

2) V2G Operation:

If the owner has opted to participate in V2G, the algorithm chooses N_C' C_i' intervals with the lowest P_j to charge the EV and N_D' D_i' intervals with the highest P_j to send energy from the EV back into the grid (i.e. sell excess charge at the highest possible price) for maximum profit.

$$\sum_1^{N_C'} C_i' - \sum_1^{N_D'} D_i' = 0 \quad (8.2)$$

$$N_C' = N_D' \quad (8.3)$$

Purchasing additional charge at cheap time intervals and selling them at higher priced time intervals would generate net profit. It must be noted that the V2G based additional charge and

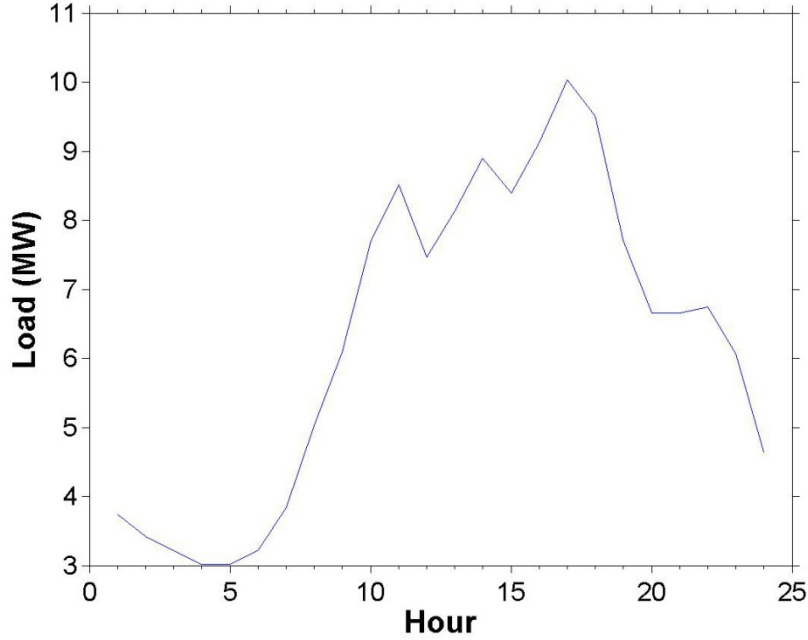


Fig. 8.5 Actual Load [76]

discharge intervals (C_i' and D_i') are equal such that when the EV owner departs, the SOC of his battery is $FSOC$ (8.2), (8.3). In scheduling charging and discharging for V2G operation the algorithm must ensure the vehicle's SOC never exceeds 100% or goes below 0% (8.4).

$$\begin{aligned} t_{arr} \leq t < t_{dep} - T_{buff} \\ 0 \leq SOC(t) \leq 100 \end{aligned}, \quad (8.4)$$

8.4 Results

8.4.1) Overview: The goals of scheduled charging optimized for electricity price are exploiting off-peak charging benefits and avoiding charging during peak load hours. In addition, while vehicles are parked and idle their energy storage capacity is utilized to alleviate grid load during peak demand.

8.4.2) Simulation Setup: In order to validate our algorithm, simulations were run using actual day-ahead electricity prices (Fig. 8.4), assuming all chargers deliver 6.6 kW, assuming all cars are Nissan Altras and have the charge profile given by Madrid et al. [76], and using two EV

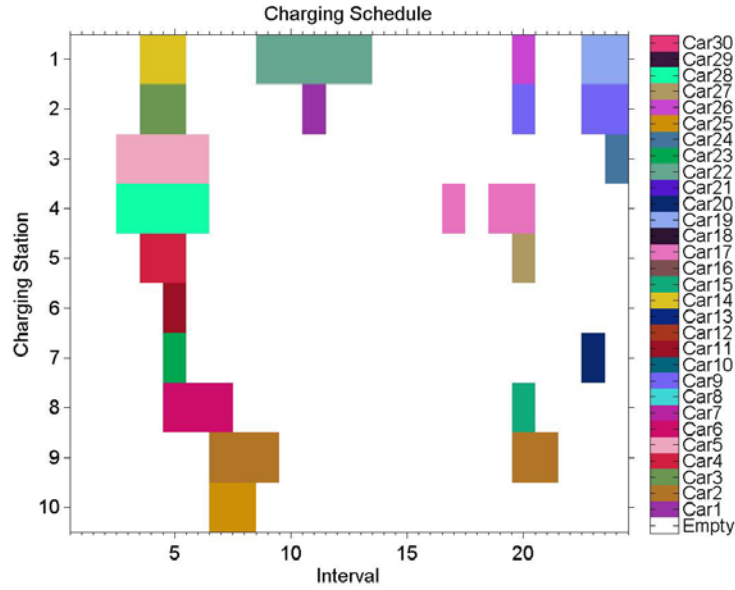


Fig. 8.6 Charging schedule for 30 cars, 10 charger

driver scenarios: 1) variable schedule and charging requirements 2) enterprise commuter schedule and charging requirements.

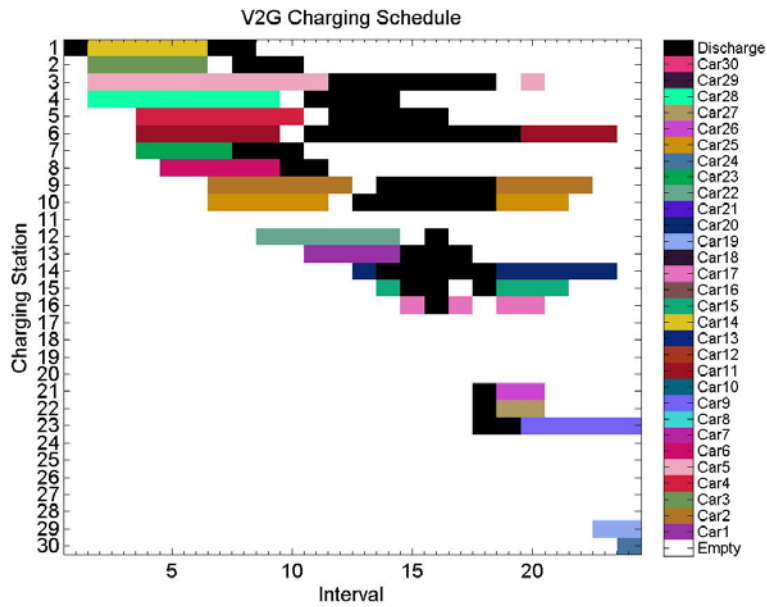


Fig. 8.7 V2G schedule for 30 cars, 10 chargers

Charge scheduling without V2G was simulated using a hypothetical parking garage with 10 chargers and 30 vehicles using the variable scenario. Cost and power usage results are compared with unmanaged charging for the same scenario.

The optimal charge interval duration for maximum V2G profit is determined. Maximum V2G profit is simulated for both types of EV owners.

8.4.3) Charge Scheduling: Charging is scheduled during the cheapest intervals an EV is parked. Comparing the price curve (Fig. 8.4) with the charging schedule for 30 variable scenario cars (Fig. 8.6) it is obvious that between intervals 10 and 20, when the price is greater than its median value of 8.2 ¢/kWh, charging is minimal. Those cars that are being charged during that interval have time constraints that limit them from being charged at any other time. The V2G schedule for the same 30 cars (Fig. 8.7) shows a similar dearth of charging from approximately interval 10 to 20. Most of the discharging happens during this interval when electricity price and demand is highest.

Scheduled charging is more cost-effective than unmanaged charging. The average total cost over 1000 trials of scheduled charging of 30 variable scenario vehicles entering a parking lot of 10 chargers over a 24 hour period was \$2.77. Unmanaged charging, which is defined as charging the vehicle as soon as it parks, of the same vehicles cost \$3.07. The savings for the variable scenario was 10%. The same simulation was run for the enterprise commuter scenario with 30 vehicles using 30 parking spots instead of 10, to accommodate all the overlapping vehicles. Total average cost was \$8.45 when their charging was unmanaged versus \$7.87, a savings of 7%.

Scheduled charging also reduces load during peak demand. For the load curve (Fig. 8.5), we define peak load as the interval from 11 AM until 7 PM, when the load is higher than

its median from 7 AM until 9 PM. Over 1000 trials for the enterprise commuter scenario, unmanaged charging uses an average of 46.5 kW during the peak load interval versus 24.9 kW for scheduled charging – a reduction of 46%. For the variable scenario, unmanaged charging uses an average of 18 kW during peak load versus 7.89 kW for scheduled charging – a reduction of 56%.

8.4.4) V2G Profit: V2G services exploit vehicles’ idle time to provide an energy resource during times of peak load. The parking garage operator can incentivize EV owners to participate in V2G using the profits earned by providing V2G services. Profits are earned when vehicles are charged during off-peak times then send their stored energy back to the grid when demand is high. The effectiveness of V2G services is measured by profit per car, not including the cost of charging to fulfill owner charge profile requirements.

V2G profit per car based on the number of incoming cars and charging stations is shown for incoming cars under the variable (Fig. 8.9) and enterprise commuter scenarios (Fig. 8.10). It

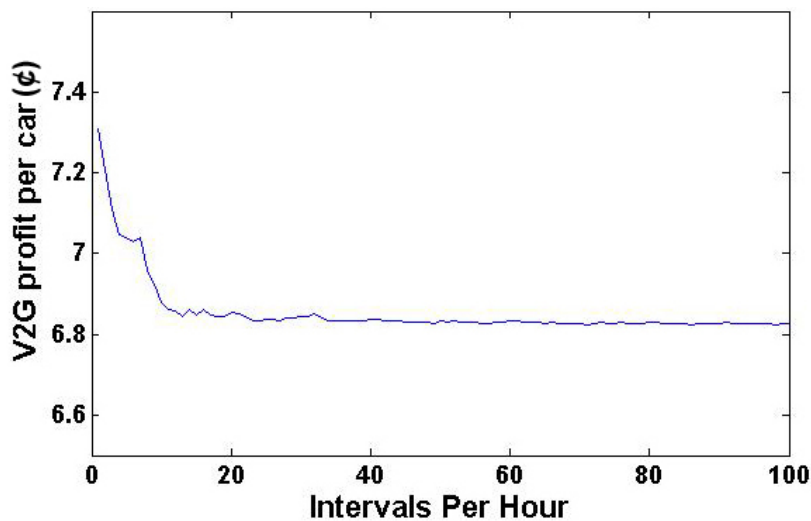


Fig. 8.8 V2G profit per car as a function of the number of intervals per hour

depends on the number and variability of incoming cars, number of charging stations in the parking structure, and the electricity price curve. Each car in the variable car scenario has a random initial arrival time, *ISOC* and random *FSOC* and departure time that are greater than their respective initial counterparts. In the enterprise commuter scenario, arrival times are evenly distributed from 7 AM to noon, departures are evenly distributed from 4 to 9 PM, *ISOC* is log-normally distributed with a mean of 22.3 and a standard deviation of 12.2 [77], and *FSOC* is fixed at 100%.

One hour was determined to be the charge interval duration that maximized V2G profit per car (Fig. 8.8). Cars used in determining this value were given a random *ISOC*, *FSOC*, arrival time, and departure time. The test was run with 10 chargers, 100 cars, and averaged over 1000 trials. The difference between the *FSOC* and *ISOC* variables determines how much time is required to charge the client car – the charging time is not included in V2G profit calculations. However, longer charging times reduce V2G profits by reducing the time available for V2G. Also since cars are used as energy storage, arrival and departure times limit when V2G can occur for each

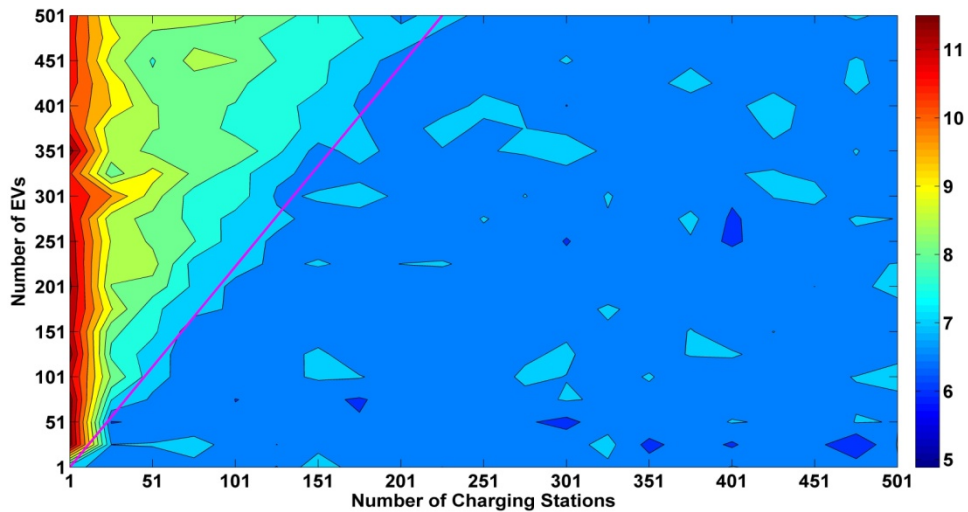


Fig. 8.9 V2G profit (€) contour plot for variable scenario

car. V2G profits increase as the ratio of parking duration to required charging time increases. In order to determine the maximum V2G profit per car, contour plots were generated for the variable scenario (Fig. 8.9) and enterprise commuter scenario (Fig. 8.10), showing V2G profit per car as a function of number of EVs and number of charging stations. The number of incoming cars and charging stations was varied from 1 to 1001 in increments of 50. Each figure plots 441 different combinations of incoming car and station numbers run over 1000 iterations. A charge interval duration of one hour and buffer time of zero were used. Each contour plot has two regions separated by a saturation limit line. This line represents the car to station ratio where every incoming car undergoes V2G and every station is utilized. The slope of the line demarcating the saturation limit is dependent on the variability of the incoming cars.

The variable incoming car scenario has a saturation limit slope that is greater than one, where the car to station ratio increases with the number of incoming cars. Since there are no constraints on the entry and departure times of incoming cars it is possible for a charging station to accommodate more than one car. Also, as the pool of incoming cars increases, there is an

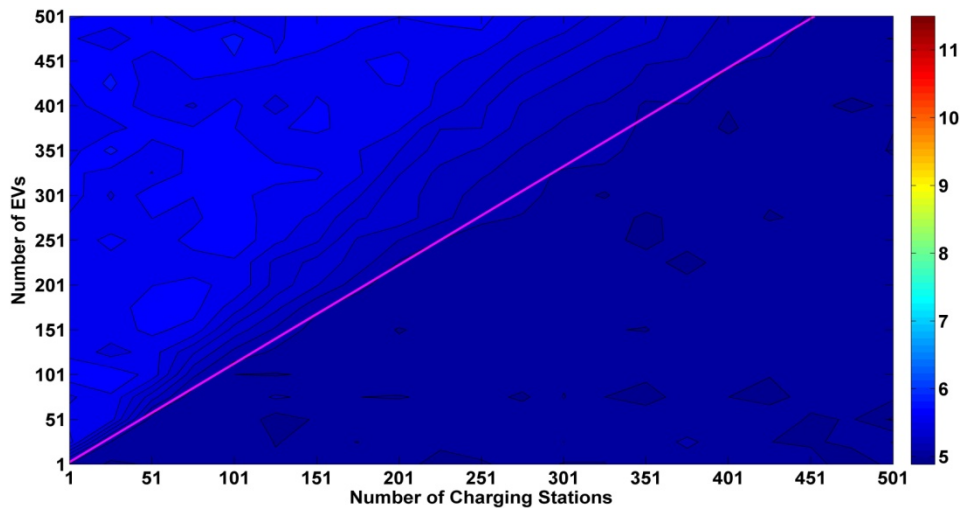


Fig. 8.10 V2G profit (€) contour plot for enterprise commuter scenario cars

increased likelihood of a car with a later arrival time to fit (park) into a previously occupied charging station after the previous car has left. Below and to the right of this saturation limit line is the charging station overcapacity region where every incoming car is able to park, charge, and go through V2G optimization. Increasing the number of charging stations or decreasing the number of cars in this area will have no effect on the average V2G profit per car, which is 6.9 cents. This particular profit value is a function of the electricity price curve and would vary as the curve varies. In this region, every incoming car is accommodated by an available charging station and thus the V2G profit per car is representative of the entire car population. For example, 1 car at 1 parking station, over multiple iterations, will have the same mean V2G profit per car as 200 cars at 1000 stations for a given electricity price curve; as they will have similar optimized charge-discharge schedules, giving similar profits. Above the saturation limit line is car oversaturation region where there are insufficient charging stations to accommodate every incoming car. Unlike the overcapacity region, increasing the number of cars or decreasing the number of charging stations in the car oversaturation region significantly affects the V2G profit per car calculation as discussed previously. Maximum profit per car over the entire contour area is 11.7 cents.

For enterprise commuter cars, the latest possible arrival time is noon and the earliest an occupied station is free is 4 PM. Because of these restrictions, it is impossible to have more than one car per station for a large duration (between noon and 4:00 pm) and thus the saturation limit line has a slope of approximately one. Average V2G profit per car is 5.6 cents – this is lower than the variable scenario because of the fixed *FSOC* requirement of 100%, versus a random *FSOC* greater than *ISOC*. Thus on an average, each enterprise commuter scenario car has a

longer charging duration and charging stations have less time available for V2G, leading to lower profit.

8.5 Conclusion

A comprehensive system leveraging mobile and RFID technologies, aggregation middleware, and an aggregated charge scheduling algorithm, that effectively schedules charging and V2G operations for cost savings and peak load reduction, has been presented.

Intelligently scheduled charging yields a cost savings of 7% for enterprise commuters and 10% for drivers with variable schedule and charging requirements. Peak load can be reduced by 46% for enterprise commuters and 56% for drivers with variable schedules and charging requirements. V2G services that utilize vehicles' idle time, when they are parked but not charging, can generate a net profit for the parking garage operator. A maximum profit of 11.7 cents per vehicle was determined to be achievable for vehicles under the variable scenario and 5.6 cents per vehicle for enterprise commuters.

The proposed system would be well suited for implementation in an enterprise environment where a large number of EVs could be aggregated to substantially impact peak load alleviation and act as a significant energy resource.

Chapter 9

Conclusion

In this paper the design and implementation of an RFID tracking solution based on web services and cloud computing resources has been shown. The emphasis was on shifting greater part of data processing to the lower level (i.e. the readers) and cloud resources. The ability of the readers to perform previously high level tasks and the possibility of shifting data processing and storage to a third party cloud computing platform gives it the benefits of low setup cost and time, remote controllability and easy scalability, hence an increased business value. The architecture was shown to perform better than a traditional localized middleware based platform for large number of tags (>400). Further, the RFID reader based Reader Resident Application and the Reader Web Service were both load tested for event processing using novel techniques where some part of or the entire event was evaluated in the reader itself. Primitive events had the least processing delay if they were only partially processed at the reader. The same was true for Complex Events for a large number of tags (>400), while for small number of tags (<400) processing delay was least if the complex events were evaluated at the reader level. The event processing performance of this architecture was found comparable and in some cases better than traditional event processing approaches available on literature. Ideas and techniques were discussed that have been implemented in this architecture to impart it additional robustness and usability.

The RFID based event processing concept was integrated with a mobile device based application powered by a automated charge/discharge scheduling algorithm for multiple Electric Vehicles in a parking lot, allowing an EV user to choose his charging options, ensuring him the most economical charging, given his selections.

REFERENCES

- [1] IDS Packaging Information Resource Center, (March 2012),
www.idspackaging.com/packaging/us/home.aspx, March 2012.
- [2] K. Finkenzeller, “RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification”, New York, NY, USA: John Wiley & Sons, Inc., 2003.
- [3] C. Roberts, “Radio frequency identification (RFID)”, *Computers & Security* 25 (2006) 18 - 26.
- [4] “AutoID ISO 18000 White Paper”, AutoID, (October 2007),
www.autoid.org/2002_Documents/sc31_wg4/docs_501-520/520_180007_WhitePaper.pdf ,
October 2007.
- [5] Oswal P., Foong M., “RFID vs Contactless Smart cards- An unending debate”, Forst & Sullivan, Oct 2006, www.frost.com/prod/servlet/market-insight-top.pag?docid=83467478,
March 2012.
- [6] D. Paret, “Technical state of art of "Radio Frequency Identification - RFID" and implications regarding standardization, regulations, human exposure, privacy”, *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, SOC-EUSAI '05, ACM, New York, NY, USA, 2005, pp. 9-11.
- [7] Rees R., “ISO Supply Chain RFID Standards”, *Automatic ID Techniques*, May 2004,
portal.etsi.org/docbox/ERM/open/RFIDWorkshop/RFID_20%20Richard%20Rees_BSI.pdf
- [8] R. Want, “An introduction to RFID technology”, *IEEE Pervasive Computing*, vol. 5, pp. 25–33, 2006.

- [9] R. Weinstein, "RFID: A technical overview and its application to the enterprise," IT Professional, vol. 7, no. 3, pp. 27–33, 2005.
- [10] B. Glover and H. Bhatt, "RFID Essentials (Theory in Practice (O'Reilly))". O'Reilly Media, Inc., 2006.
- [11] C. Thompson, "Radio frequency tags for identifying legitimate drug products discussed by tech industry," Am J Health Syst Pharm, vol. 61, no. 14, pp. 1430–, 2004. [Online]. Available:<http://www.ajhp.org>
- [12] E. Cerami, "Web Services Essentials", S. St.Laurent, Ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2002.
- [13] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML) 1.0 (third edition), W3C recommendation", W3C, Tech. Rep., 2004. [Online]. Available: <http://www.w3.org/TR/2004/REC-xml-20040204>
- [14] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]
- [15] Das R., Harrop P., "RFID Forecasts, Players & opportunities 2008-2018", www.idtechex.com/research/reports/rfid_forecasts_players_and_opportunities_2008_2018_000193.asp
- [16] David H. Williams, "The strategic implications of Wal-Mart's RFID Mandate", July 2004, <http://www.directionsmag.com/articles/the-strategic-implications-of-wal-marts-rfid-mandate/123667>.

- [16a] “On the Utilization and Integration of RFID data into Enterprise Information Systems via WinRFID”, DETC2007-34731, Computers in Engineering Conference, Sep 4-7, 2007, Las Vegas, NV, X. Su, C.-C. Chu, B.S. Prabhu, R. Gadh.
- [17] Roberti M., “DOD Releases Final RFID Policy”, Aug 2004, www.rfidjournal.com/article/view/1080/1/1.
- [18] Roberti M., “Boeing, Airbus Team on Standards”, May 2004, www.rfidjournal.com/article/view/934/1/1.
- [19] Paul Faber, “Pharmaceutical E-pedigree- Biggest Supply Chain Topic of 2008”, Jan 2008.
- [20] X. Su, C.-C. Chu, B. S. Prabhu and R. Gadh, “On the Identification Device Management and Data Capture via WinRFID Edge-Server”, IEEE Systems Journal, 1(2), Dec 2007,
- [21] “The EPCglobal Network: Enhancing the Supply Chain”, Verisign White Paper, July 2008, <http://www.verisign.com/static/DEV044095.pdf>.
- [22] Wayne Kernochan, “RFID: Metadata in motion”, March 2006, <http://searchdatacenter.techtarget.com/tip/RFID-Metadata-in-motion>
- [23] S. Chalasani and R. Boppana, “Data architectures for RFID transactions,” Industrial Informatics, IEEE Transactions on, vol. 3, no. 3, pp. 246 –257, Aug. 2007.
- [24] O. Mylly, “Rfid data management, aggregation and filtering.”
- [25] S. S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. Sarma, “Managing RFID data,” in VLDB ’04: Proceedings of the Thirtieth international conference on Very large data bases. VLDB Endowment, 2004, pp. 1189–1195.

[26] H. Gonzalez, J. Han, X. Li, and D. Klabjan, “Warehousing and analyzing massive RFID data sets,” in ICDE ’06: Proceedings of the 22nd International Conference on Data Engineering. Washington, DC, USA: IEEE Computer Society, 2006, p. 83.

[27] Floerkemeier, C. Roduner, C. Lampe, M, “RFID Application Development with the Accada Middleware Platform,” Systems Journal, IEEE, Dec 2007

[28] MetaFi overview, Skyetek,
www.skyetek.com/ProductsServices/RFIDApplicationServices/MetaFiOverview/tabid/501/Default.aspx.

[29] Harvey V. Janelli, “Web Services Application for RFID”, Imobile Systems, 2005

[30] InfoSphere Traceability Server, IBM, October 2008, www-01.ibm.com/software/data/masterdata/rfid/

[31] Pappalardo D., “AT&T offering managed RFID service”, NetworkWorld, May 2006,
www.networkworld.com/news/2006/052906-att-rfid.html

[32] D. Bade and W. Lamersdorf, “An agent-based event processing middleware for sensor networks and RFID systems,” The Computer Journal, vol. 54, no. 3, pp. 321–331, 2011.

[Online]. Available: <http://comjnl.oxfordjournals.org/content/54/3/321.abstract>

[33] E. W. T. Ngai, T. C. E. Cheng, S. Au, and K.-h. Lai, “Mobile commerce integrated with RFID technology in a container depot,” Decis. Support Syst., vol. 43, pp. 62–76, February 2007.

[Online]. Available: <http://dl.acm.org/citation.cfm?id=1223916.1223958>

- [34] S. R. Jeffery, M. J. Franklin, and M. Garofalakis, "An adaptive RFID middleware for supporting metaphysical data independence," *The VLDB Journal*, vol. 17, pp. 265–289, March 2008. [Online]. Available: <http://dx.doi.org/10.1007/s00778-007-0084-8>
- [35] J. Schwierien and G. Vossen, "Id-services: an RFID middleware architecture for mobile applications," *Information Systems Frontiers*, vol. 12, pp. 529–539, November 2010. [Online]. Available: <http://dx.doi.org/10.1007/s10796-009-9214-8>
- [36] D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, rest and mashups to simplify RFID application development and deployment," in *Proceedings of the Second International Workshop on Web of Things*, ser. *WoT '11*. New York, NY, USA: ACM, 2011, pp. 9:1–9:6. [Online]. Available: <http://doi.acm.org/10.1145/1993966.1993979>
- [37] I. Zappia, D. Parlanti, and F. Paganelli, "Lisep: A lightweight and extensible tool for complex event processing," in *Services Computing (SCC), 2011 IEEE International Conference on*, July 2011, pp. 701 – 708.
- [38] F. Wang, S. Liu, and P. Liu, "Complex RFID event processing," *VLDB J.*, vol. 18, no. 4, pp. 913–931, 2009.
- [39] F. Wang and P. Liu, "Temporal management of RFID data," in *VLDB '05: Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 1128–1139.
- [40] Y. Bai, F. Wang, and P. Liu, "Efficiently filtering RFID data streams," in *In CleanDB Workshop*, 2006, pp. 50–57.
- [41] D. Sundaram, W. Zhou, S. Piramuthu, and S. Pienaar, "Knowledge-based RFID enabled web service architecture for supply chain management," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7937 – 7946, 2010. [Online]. Available:

<http://www.sciencedirect.com/science/article/B6V03-50297C4-2/2/6bcc1b5293adec94ca189b36b213f25c>

[42] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, and G. Borriello, "Building the internet of things using RFID: The RFID ecosystem experience," *Internet Computing*, IEEE, vol. 13, no. 3, pp. 48–55, 2009.

[43] Siorpaes, S., Broll, G., Paolucci, M., Rukzio, E., Hamard, J., Wagner, M., & Schmidt, A. (2006). *Mobile Interaction with the Internet of Things*. Embedded Interaction Research Group, 207, 7-10.

[44] Broll, G.; Rukzio, E.; Paolucci, M.; Wagner, M.; Schmidt, A.; Hussmann, H.; , "Perci: Pervasive Service Interaction with the Internet of Things," *Internet Computing, IEEE* , vol.13, no.6, pp.74-81, Nov.-Dec. 2009

[45] K. e. a. Romer, "Smart identification frameworks for ubiquitous computing applications," *Wirel. Netw.*, vol. 10, no. 6, pp. 689–700, 2004.

[46] X. Su, C.-C. Chu, B. Prabhu, and R. Gadh, "On the identification device management and data capture via WinRFID edge-server," *Systems Journal*, IEEE, vol. 1, no. 2, pp. 95–104, dec. 2007.

[47] N. Ahmed, R. Kumar, R. S. French, and U. Ramach, "U.: Rf2id: A reliable middleware framework for RFID deployment," in *In: IEEE IPDPS*, March 2007, IEEE Computer Society Press, Los Alamitos, 2007.

[48] T. Sanchez Lopez and D. Kim, "A context middleware based on sensor and RFID information," in *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07*. Fifth Annual IEEE International Conference on, 19-23 2007, pp. 331–336.

[49] S. Liu, F. Wang, and P. Liu, “Integrated data modeling for querying physical objects in RFID-enabled pervasive computing,” in MDM ’07: Proceedings of the 2007 International Conference on Mobile Data Management. Washington, DC, USA: IEEE Computer Society, 2007, pp. 140–145.

[50] A. Chattopadhyay, B. Prabhu, and R. Gadh, “Web based RFID asset management solution established on cloud services,” in RFID-Technologies and Applications (RFID-TA), 2011 IEEE International Conference on, Sept. 2011, pp. 292 –299.

[51] C. Floerkemeier and M. Lampe, “RFID middleware design: addressing application requirements and RFID constraints,” in SOC-EUSAI ’05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence. New York, NY, USA: ACM, 2005, pp. 219–224.

[52] M. T. Ozsu and P. Valduriez, “Distributed database systems: Where are we now?” Computer, vol. 24, no. 8, pp. 68–78, 1991.

[53] C. Arunabh. [Online]. Available: <https://164.67.192.236/Website/WebSite/Login.aspx>

[54] Amazon Web Services, Amazon, March 2012, [Online]. Available: <http://aws.amazon.com/ec2/>

[55] M. Wei, I. Ari, J. Li, and M. Dekhil, “Receptor: Sensing complex events in data streams for service-oriented architectures,” 2007.

[56] United States Department of Energy. (2011, February) “One Million Electric Vehicles By 2015,”

http://energy.gov/sites/prod/files/edg/news/documents/1_Million_Electric_Vehicle_Report_Final.pdf

- [57] J. Kiviluoma, P. Meibom, "Methodology for modelling plug-in electric vehicles in the power system and cost estimates for a system with either smart or dumb electric vehicles," *Energy*, Volume 36, Issue 3, pp. 1758-1767, March 2011.
- [58] M. Kintner-Meyer, K. Schneider and R.Pratt, "Impacts assessment of plug-in hybrid electric vehicles on electric utilities and regional U.S. power grids," Technical analysis. In 10th Annual EUEC Conference, Tucson, AZ, 2007. Pacific Northwest National Laboratory (PNNL).
- [59] W. Kempton, J. Tomi•, "Vehicle-to-grid power implementation: From stabilizing the grid to supporting large-scale renewable energy," *Journal of Power Sources*, Volume 144, Issue 1, pp. 280-294, June 1, 2005.
- [60] C. Guille, G. Gross, "A conceptual framework for the vehicle-to-grid (V2G) implementation," *Energy Policy*, Volume 37, Issue 11, pp. 4379-4390, November 2009.
- [61] J. Ferreira, V. Monteiro, J. Afonso, A. Silva, "Smart Electric Vehicle Charging System", 2011 IEEE Intelligent Vehicles Symposium (IV) Baden-Baden, Germany, June 5-9, 2011.
- [62] S. Shao, M. Pipattanasomporn, S. Rahman, "Challenges of PHEV penetration to the residential distribution network," *Power & Energy Society General Meeting, 2009. PES '09. IEEE* , vol., no., pp.1-8, 26-30, July 2009.
- [63] X. Yu, "Impacts assessment of PHEV charge profiles on generation expansion using national energy modeling system," *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE* , vol., no., pp.1-5, 20-24, July 2008.

- [64] Porter, J.D.; Kim, D.S.; , "An RFID-Enabled Road Pricing System for Transportation,"
Systems Journal, IEEE , vol.2, no.2, pp.248-257, June 2008
- [65] Theo, L.; Jonas, F.; , "Using the Energy Name Service (ENS) for electric mobility
roaming," eChallenges, 2010 , vol., no., pp.1-7, 27-29 Oct. 2010
- [66] Wang-Cheol Song Authentication System for Electrical Charging of Electrical Vehicles in
the Housing Development Security-Enriched Urban Computing and Smart Grid
Communications in Computer and Information Science, 2010, Volume 78, 261-266
- [67] Soares, J.; Sousa, T.; Morais, H.; Vale, Z.; Faria, P.; , "An optimal scheduling problem in
distribution networks considering V2G," Computational Intelligence Applications In Smart
Grid (CIASG), 2011 IEEE Symposium on , vol., no., pp.1-8, 11-15 April 2011
- [68] Venayagamoorthy, G.K.; Mitra, P.; Corzine, K.; Huston, C.; , "Real-time modeling of
distributed plug-in vehicles for V2G transactions," Energy Conversion Congress and
Exposition, 2009. ECCE 2009. IEEE , vol., no., pp.3937-3941, 20-24 Sept. 2009
- [69] Hutson, C.; Venayagamoorthy, G.K.; Corzine, K.A. "Intelligent Scheduling of Hybrid and
Electric Vehicle Storage Capacity in a Parking Lot for Profit Maximization in Grid Power
Transactions." IEEE Energy 2030. Nov. 2008.
- [70] Diyun Wu; Chau, K.T.; Shuang Gao; , "Multilayer framework for vehicle-to-grid
operation," Vehicle Power and Propulsion Conference (VPPC), 2010 IEEE , vol., no., pp.1-
6, 1-3 Sept. 2010

- [71] Ahmed Yousuf Saber, Ganesh Kumar Venayagamoorthy, Intelligent unit commitment with vehicle-to-grid --A cost-emission optimization, Journal of Power Sources, Volume 195, Issue 3, 1 February 2010, Pages 898-911
- [72] Schieffer, S.V. (2010) To charge or not to charge? Decentralized charging decisions for the smart grid, *Semesterarbeit*, IVT, ETH Zürich, Zürich, Herbstsemester 2010
- [73] S. Han, S. H. Han, K. Sezaki, "Design of an optimal aggregator for vehicle-to-grid regulation service," in Proc. of IEEE PES Conference on Innovative Smart Grid Technologies, Gaithersburg, MD, Jan. 2010.
- [74] E. Tate, M. Harpster, P. Savagian, "The Electrification of the Automobile: From Conventional Hybrid, to Plug-in Hybrids, to Extended-Range Electric Vehicles," 2008 SAE International World Congress, 2008.
- [75] C. Madrid, J. Argueta, and J. Smith, "Performance characterization—1999 Nissan Altra-EV with lithium-ion battery," Southern California EDISON, Sep. 1999.
- [76] Y. Xu, L. Xie, C. Singh, "Optimal scheduling and operation of load aggregator with electric energy storage in power markets," North American Power Symposium (NAPS), 2010 , vol., no., pp.1-7, 26-28, Sept. 2010.
- [77] K. Qian, C. Zhou, M. Allan, Y. Yuan, "Modeling of Load Demand Due to EV Battery Charging in Distribution Systems," Power Systems, IEEE Transactions on , vol.26, no.2, pp.802-810, May 2011.

[78] Xiaoyong Su, Chi, Cheng Chu, B. S. Prabhu, and Rajit Gadh, "Creating a RFID data integration framework for enterprise information systems", *Int. J. Internet Protoc. Technol.* 4, 4 (November 2009), 221-231.

[79] Xiaoyong Su, Rajit Gadh, "A Rule Language and Framework for RFID Data Capture and Processing in Manufacturing Enterprise System", *International Journal of Internet Manufacturing and Services*, Volume 2, Issue 2, 2010

[80] Xiaoyong Su, Chi-Cheng Chu, B.S. Prabhu and Rajit Gadh, "Service Organization and Discovery for Facilitating RFID Network Manageability and Usability via WinRFID Middleware", WTS 2008, Cal Poly Pomona, Pomona, California, April 24-26, 2008

[81] Shiv Prabhu, Xiaoyong Su, Charlie Qiu, Chi-Cheng Chu, Brandi Schmitt, Rajit Gadh, "SpecimenTrak: an RFID system for tagging and tracking anatomical specimens", 24th Annual Scientific Session of The American Association of Clinical Anatomists, Jun. 16-20, 2007, Las Vegas.

[82] Xiaoyong Su, Chi-Cheng Chu, B. S. Prabhu, Rajit Gadh, "RFID Automatic Identification and Data Capture", Book Chapter in "The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems," *Wireless Networks and Mobile Communications Series*, Lu Yan, Yan Zhang, Laurence T. Yang, Huansheng Ning (eds.), Auerbach Publications, Taylor & Francis Group, March 2008, pp. 33-54

[83] Xiaoyong Su, Chi-Cheng Chu, B.S. Prabhu, Rajit Gadh, Brandi Schmitt, "SpecimenTrak: A Demonstration of the Anatomical Specimen Tagging and Tracking", 24th Annual Scientific Session of the American Association of Clinical Anatomists, Jun 17-20, 2007, Las Vegas.