INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Engineering Business Management
Special Issue on Radio Frequency Identification and Wireless Sensor Networks
Editor: Cristina Turcu

XRFID: Design of an XML Based Efficient Middleware for RFID Systems

Regular Paper

Indrajit Bhattacharya[1,*], Amit Kumar Gupta[2] and Uttam Kumar Roy[3]

1 Assistant Professor, Dept. of Comp. Applications, Kalyani Govt. Engineering College, Kalyani
2 Lecturer, MCA Department, DSMS Business School, Durgapur-12
3 Assistant Professor, Dept. of Information Technology, Jadavpur University, Kolkata
* Corresponding author E-mail: indra51276@gmail.com

Originally published in the International Journal of Radio Frequency Identification & Wireless Sensor Networks, ISSN 1847-9812

Abstract Radio frequency identification (RFID) technology can automatically and inexpensively track items as they are moved through the supply chain. This can automate the whole updating and management system, thereby making the system work with a much smaller workforce and reducing the error that can occur because of interference by human beings. One of the major advantages RFID provides is that it does not require direct physical contact with the objects and also does not require the object to be placed in its 'Line-of-Sight'. This has given it an edge over other auto-identification systems, like bar-codes. The recent proliferation of RFID tags and readers would require dedicated and very efficient middleware solutions that manage readers and process the vast amount of captured data according to the need of various applications. RFID middleware is the software sitting in between various RFID readers and the enterprise applications. Extracting meaningful information out of huge amount of scan data is a challenging task. In this paper we like to analyze the requirements and propose a design for such an RFID middleware. This paper demonstrates how to enable the middleware to handle a large amount of RFID scan data and execute business rules in real-time. The conventional existing middleware solutions show dramatic degradation in their performance when the number of simultaneously working readers increases. Our proposed solution tries to recover from that situation also. One of the major issues for large scale deployment of RFID systems is the design of a robust and flexible middleware system to interface various applications to the RFID readers. Most of the existing RFID middleware systems are costly, bulky, non-portable and heavily dependent on the support software. Our work also provides flexibility for easy addition and removal of applications and hardware.

Keywords RFID, Readers, Tags, Middleware, XML, XML-RPC

1. Introduction

Radio Frequency Identification (RFID) [4][6] is a means of identifying an object or a person using radio frequency transmission, generally from a distance. Like the extensively used barcode based systems and optical recognition systems (OCR), RFID is also an auto-identification system. RFID can be understood as the 'next generation bar code', but this description only does partial justice to the huge advantages RFID has over bar codes.

Early RFID implementations used to be driven by external mandates, but along with significant technological improvements, more readily available component options, and cost reductions, the technology has proven its value in driving significant operational efficiencies, and RFID has gained a much broader adoption [1].

Today, industries are looking beyond the realm of compliance, as they seek competitive advantages and integrate RFID much earlier into their production processes. Innovative companies are expanding the use of RFID in their supply chain, logistics and asset tracking operations [8]. As a result, they are achieving demonstrable improvements in supply chain visibility, forecast accuracy, reduced out-of-stock situations and reduced counterfeiting.

RFID system has Tags that can carry information like serial numbers, model numbers, color, place of assembly or other types of data as well. This information gives the unique identity of the item (object) to which this tag is attached. This identification occurs when these tags move in the vicinity of any RFID reader. An RFID reader is a device that can access the information of any RFID tag in its vicinity by establishing a wireless communication path using radio frequency. The information the readers obtain from the tags is then sent to the middleware [11][15] that sits above the readers. The middleware processes this information according to the need of the RFID applications. These applications can access these processed data by communicating with the middleware. The middleware, in this regard functions like a server that collects information from the readers and provides services to the applications on the other side of it.

Most of the existing RFID middleware systems are costly, non-portable and non-flexible [10][13][14]. Most of them show performance degradation when the number of simultaneously working readers are increased beyond a certain limit. The objective of our work is to analyze the current and required scenario, design and develop efficient and flexible middleware software for RFID systems incorporating the analyzed requirements.

We have simulated the unique tag ID generation [5] through the use of random number generation and the support is provided completely in the software. The designed middleware provides the support for the user applications to communicate with the hardware (readers and tags, in this case). We have designed a middleware that support simultaneous communication of multiple applications with the RFID hardware. The middleware must provide all data processing capabilities like filtering, grouping and duplicate data removal.

The middleware provides the storage for the tag information periodically sent by the readers in different files. The queries generated by the applications are performed on these files. A simple user interface is provided to communicate with the middleware. The request from the applications is sent to the middleware in the form of simple XML-RPC requests. The response against this request is transferred back to the application in simple XML format. The required information for the application is parsed from the database stored in the middleware and appropriate XML response files are generated. These XML response files are sent to the applications for further processing.

2. Related Work

A lot of work has been done in the field of RFID and its applications [15]. The main focus of all those developments had been the middleware and the RFID applications. And there is a huge scope for research in the area.

Because of recent increase in the interest generated in RFID field, a large number of small and big players have started showing interest in the field. And most of the RFID middleware solutions developed are commercial. These include “BizTalk RFID” from Microsoft, “RFID Middleware” from Sun, and “Fusion” from Oracle, to name a few. RFID and RFID middleware in particular, has been the centre of research for quite some time. “WinRFID” by UCLA and “Accada” by ETH Zurich are some of the products generated as a result of research in the field [2].

The Sun Java System RFID software, which is middleware software, is a part of the Java Enterprise System (JES). It has four components namely, the RFID Event Manager, the RFID Management Console, the RFID Information Server, and a software development kit (SDK). The RFID Event Manager helps in the capture, filtering, and storage of the events generated by the RFID readers. The RFID middleware console (a browser based management interface), allows configuration of the attributes and parameters of the middleware. The information server stores and queries the EPC related data, and manages inter-Enterprise handling of data. The SDK provides a development platform for building custom applications. The Sun middleware presents the hardware as logical readers to the applications, where each logical reader may consist of one or more physical readers. The applications, according to their needs, select one or more logical readers and apply processing parameters to the entire group [3].

The WinRFID from University of California Los Angeles (UCLA) has certain unique features like, hiding of communication details from the end-users, network management on a large-scale, intelligent data processing and routing, support for hardware and software interoperability, provision for system integration and system extendibility, etc. It has novel algorithms and data-structure schemes capable of processing large amounts of data, rectifying errors in real-time, identifying patterns, correlating events to each other, reorganizing data and recovering from unwanted faults and exceptions. It provides support for simultaneous working of sundry readers and tags at different frequencies, and using different protocols. An XML based framework is used that helps the filtered data from the tags to be formatted as per the customized requirements [2].

Accada by ETH Zurich helps in RFID application development. The Accada platform manages readers, filters and aggregates RFID data, and helps in interpreting the RFID data. It uses EPCglobal based specifications for the reader protocols, the application level event specifications and the EPCIS capture and query interface to handle RFID data flow across enterprises. The platform has three main modules- the reader, the middleware and the EPC information services module. The Accada reader implementation uses standard edition of SUN Java Virtual Machine and not a micro edition[7].

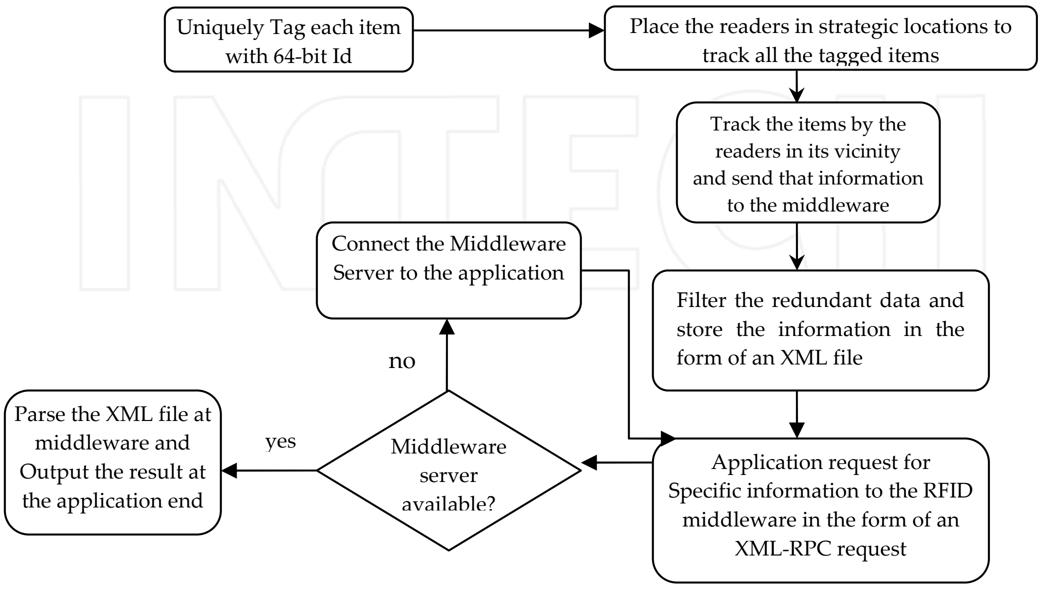
The Biztalk RFID middleware solution from Microsoft supports plug-and-play architecture to provide support for both standard and non-standard devices. It supports an event processing engine that manages the RFID events by creating business rules. It also provides real time visibility of the RFID data.

3. Proposed middleware and its sub-systems

3.1. System Overview

The solution that we have proposed here consists of developing an efficient and flexible RFID middleware system, right from the scratch. At the beginning, we have created user specified number of readers, each reader executing in its own separate zone. These readers have been assigned co-ordinate values in a user specified geographical region, i.e., we placed the generated readers at specified locations in the region. The co-ordinates are specifically calculated and generated so as to have an optimum efficiency for the readers. We also stored these co-ordinate values for later use.

After that, we provided the option to create as many number of tags as the user may want. These tags are placed at random locations inside the same region. The coordinates of these tags are randomly generated, within the limits of the region, so as to have the feel of simulation. Each and every tag gets a unique identification number that is 16-hex bits (64-binary bits) long. The flowchart to implement the proposed system is shown in figure 1.



Uniquely Tag each item with 64-bit Id
Place the readers in strategic locations to track all the tagged items
Track the items by the readers in its vicinity and send that information to the middleware
Filter the redundant data and store the information in the form of an XML file
Application request for Specific information to the RFID middleware in the form of an XML-RPC request
Middleware server available?
yes
no
Parse the XML file at middleware and Output the result at the application end
Connect the Middleware Server to the application

Figure 1. Flowchart of the proposed RFID Middleware system

3.2. Middleware Functions

The middleware software is supposed to perform the following:

- To receive packets from the readers lying at lower level.
- To disseminate the data, i.e. to decide which applications are interested in what processed information, and to what latency extent.
- To process, filter and aggregate if required the received information according to the needs of the various applications.
- To perform Load Balancing of the loads at the middleware [9][12], whenever it is required.
- Transmit the finally processed information to the respective applications at the upper layer.
- To provide for appropriate privacy and security.

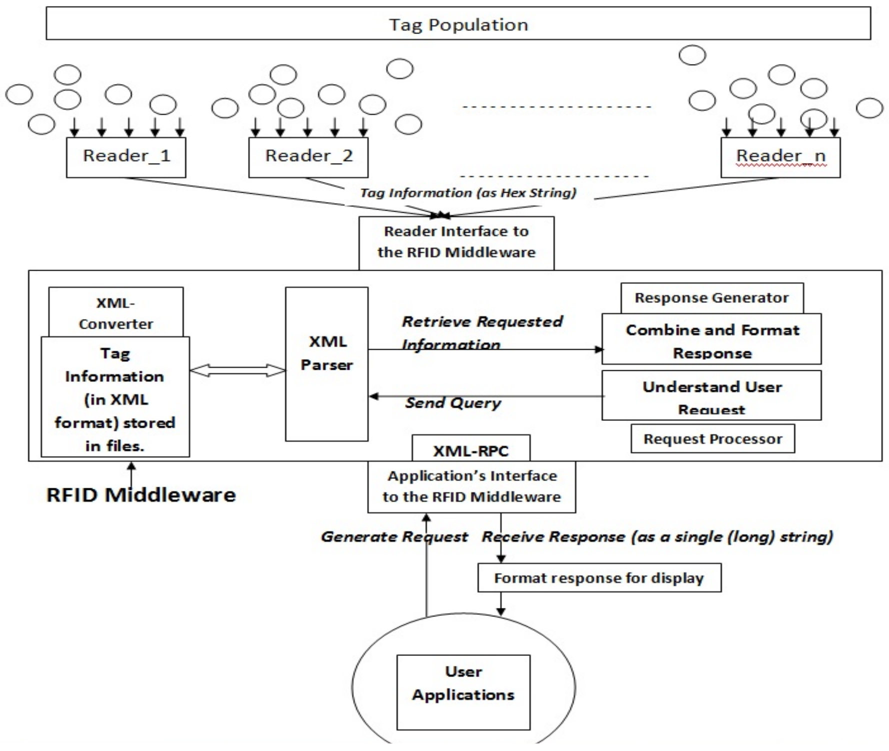
The communication between the user applications and the middleware is done using XML-Remote Procedure Call (XML-RPC, in short). There are numerous other methods for remote communication available but we have chosen XML-RPC over others because it is simple to understand, is very light weight, clear and easy to implement, and is also flexible in its functionalities.

3.3. Proposed Middleware and its sub-systems

The proposed middleware system consists of the following sub-systems, as shown in Figure 2 below, and is described later on.

The middleware obtains tag information from the readers through the RFID Reader-Middleware interface. The middleware stores this information in files, strictly in XML-format. The raw cleaned (verified and filtered) tag data from the readers is formatted in a variety of ways to a high-level XML based representation, using an XML-converter. When a user application (client) generates a request, it is sent to the middleware, using XML-RPC. The request processor present in the middleware understands this request and sends an appropriate query to the XML parser. The XML parser then retrieves required fields' information and sends it to the response generator, which combines the information into a string and sends the formatted response to the client. The client, in its turn, performs further formatting of this response locally, for display or some other use.

The information is filtered, cleaned, aggregated and adapted as per requirements. We have tried to provide data in a format agreeable to decision making at the application layer. The data is designed in such a way that it is possible to easily search through the whole information based on key fields.



Tag Population
Reader_1
Reader_2
Reader_n
Tag Information (as Hex String)
Reader Interface to the RFID Middleware
XML-Converter
Tag Information (in XML format) stored in files.
XML Parser
Retrieve Requested Information
Send Query
Response Generator
Combine and Format Response
Understand User Request
Request Processor
XML-RPC
Application's Interface to the RFID Middleware
RFID Middleware
Generate Request
Receive Response (as a single (long) string)
Format response for display
User Applications

Figure 2. Proposed Middleware and its Sub-systems

3.3.1. Why XML?

The extensible markup language, XML, is a way to mark-up text in a structured document. XML is a simplification of the complex SGML standard. SGML, the Standard Generalized Markup Language, is an international (ISO) standard for marking up text and graphics. The best known application of SGML is HTML. Although SGML data is very easy to write, it's very difficult to write a generic SGML parser. When designing XML however, the authors removed much of the flexibility of SGML making it much easier to parse XML documents correctly.

XML data is structured as a tree of entities. An entity can be a string of character data or an element which can contain other entities. Elements can optionally have a set of attributes. Attributes are key/value pairs which set some properties of an element.

We have used the XML based representation because it facilitates the ease of data consumption by enterprise applications such as warehouse management, supply chain management, enterprise resource planning and others. This is because most of these systems have adapters to export XML based data and wizards to design templates to parse the XML tree within these systems are also quite established. XML is mainly used to satisfy customized requirements.

Each tag consists of the following fields (with total size 64-bits):

Manufacturer	***Product Type***	***Product Sub-Type***	***Unique ID***	***Date-of-Manufacture***	***Period-to-expire* (in weeks)**	***Price* (in US Dollars)**
8-bits	8-bits	8-bits	8-bits	16-bits	8-bits	8-bits

A sample XML-file in our designed format and using our tags that we have used to store the information is given below:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<items>
<tag>
<manufacturer>a8</manufacturer>
<p_type>a8</p_type>
<p_subtype>1f</p_subtype>
<u_id>cd</u_id>
<d_o_m>d374</d_o_m>
<p_o_e>e6</p_o_e>
<price>d8</price>
</tag>
```

All the tags' information gets stored in the above format. When the middleware obtains a user request, which again is in XML-format, from the applications, it parses this XML file to abstract out the required information and send this back to the requesting application for further processing or display.

3.3.2. About XML-RPC

XML-RPC is a simple, portable way to make remote procedure calls over HTTP. It can be used with the programming languages Perl, Java, Python, C, C++, PHP and many others. The XML-RPC implementations are available for the operating systems UNIX, Windows and the Macintosh.

XML-RPC values are encoded as XML to carry out their work. It supports the data types int, string, boolean, datetime, array, struct, and base64 (raw binary data of any length encoded using Base64).

4. Implementation and Result

4.1. Experimental Set-Up

We have developed the RFID middleware system on a Java platform. Firstly, unique tag IDs are randomly generated for the system. There is an option for the user to specify the number of tags to be placed in the user specified area. The tag IDs are 64-bits in length, and contain seven different fields. To ascertain that no duplicate tag IDs get generated, we have used the linear search method. Then we converted these tag-IDs in XML-format using customized tag-names, and stored them in an XML file. We used co-ordinates (2-dimensional) to position the tags and readers in the area.

After that we designed the middleware that acts as the server. The Readers gather information regarding the tags in their vicinity and send that information periodically to the middleware. The middleware is the remote machine that collects all information regarding all the readers and tags in the system. It is the information repository for the whole system. It uses XML-RPC as the technology for its communication with the clients. The client sends request to the server for obtaining appropriate information. The middleware server then parses the XML file and selects out only the required information, combines the whole information collected above in a single string, and sends this string back to the client. Using this method (XML-RPC) to communicate has made the system very light-weight.

Finally, we developed the user applications. Here, we have made a simple user interface to select fields and make requests. This interface has been made using the

Swings of JAVA. When the user completes making the request and submits, the result gets displayed in another window in a tabular format. At first, the client receives the information from the server in codes (hexadecimal) which is then converted into human understandable form by the client program. The client employs appropriate conversions for the system. One important point that we would make here is that we have tried to make the system compact by using the least possible number of bits for the date field. The server is unaware of the complexities; it just sends the 16-bits raw data to the client. We have made it the responsibility of the client to convert that raw data into appropriate date format so that human beings can understand it. This is one of the various techniques we have employed to make the middleware server light-weight. The following figures demonstrate some snapshots of the implemented system:

The result obtained when the client requested for information relating to the available manufacturers, product types, unique IDs and their date of manufacture is shown in figure 4.

We have carried out our experiments with varying number of tags and readers in a departmental store scenario over a 300x300m^2 area. We have observed the load on the readers that is the number of tags that are identified by the readers for different number of tags and readers that are kept active for different cases. The distribution of loads among the readers are shown in a bar diagram as shown in the figure 5. On an average, it may be observed that our proposed system effectively reduces the load on the readers and hence help faster retrieval of information.

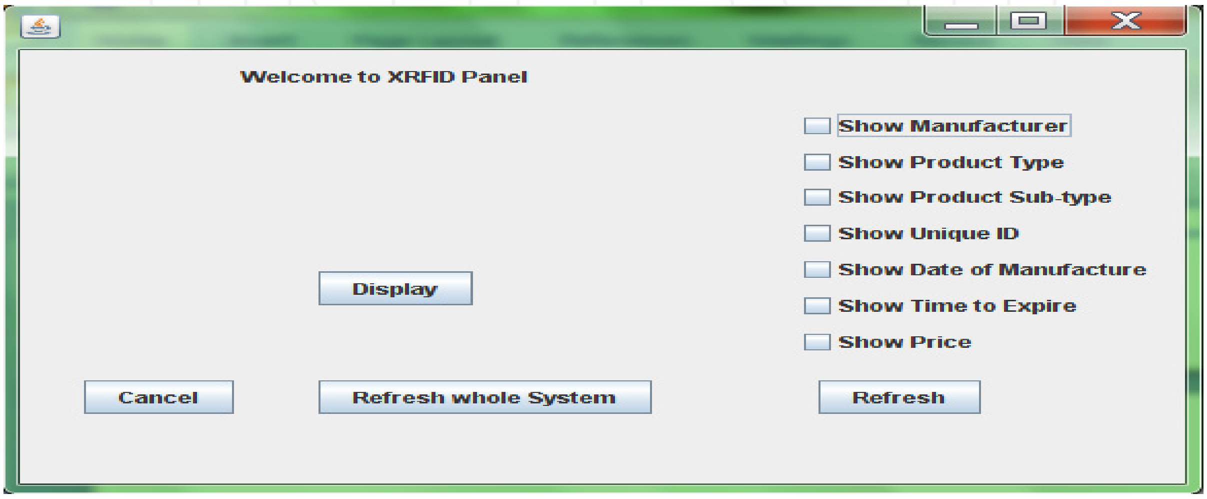
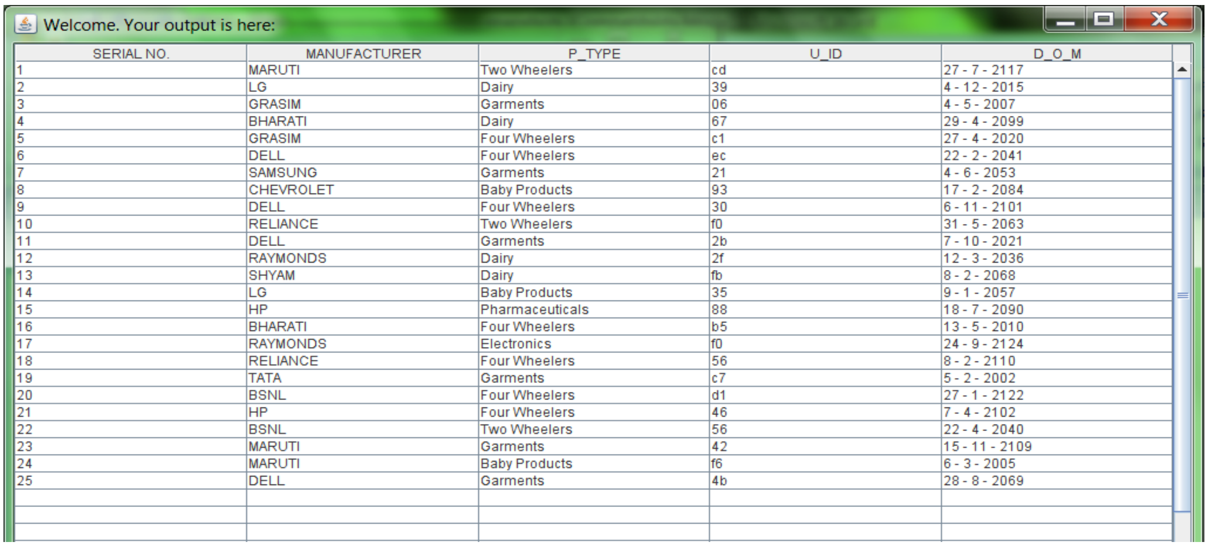


Figure 3. The user interface for retrieving information


Welcome. Your output is here:

SERIAL NO.	MANUFACTURER	P_TYPE	U_ID	D_O_M
1	MARUTI	Two Wheelers	cd	27 - 7 - 2117
2	LG	Dairy	39	4 - 12 - 2015
3	GRASIM	Garments	06	4 - 5 - 2007
4	BHARATI	Dairy	67	29 - 4 - 2099
5	GRASIM	Four Wheelers	c1	27 - 4 - 2020
6	DELL	Four Wheelers	ec	22 - 2 - 2041
7	SAMSUNG	Garments	21	4 - 6 - 2053
8	CHEVROLET	Baby Products	93	17 - 2 - 2084
9	DELL	Four Wheelers	30	6 - 11 - 2101
10	RELIANCE	Two Wheelers	f0	31 - 5 - 2063
11	DELL	Garments	2b	7 - 10 - 2021
12	RAYMONDS	Dairy	2f	12 - 3 - 2036
13	SHYAM	Dairy	fb	8 - 2 - 2068
14	LG	Baby Products	35	9 - 1 - 2057
15	HP	Pharmaceuticals	88	18 - 7 - 2090
16	BHARATI	Four Wheelers	b5	13 - 5 - 2010
17	RAYMONDS	Electronics	f0	24 - 9 - 2124
18	RELIANCE	Four Wheelers	56	8 - 2 - 2110
19	TATA	Garments	c7	5 - 2 - 2002
20	BSNL	Four Wheelers	d1	27 - 1 - 2122
21	HP	Four Wheelers	46	7 - 4 - 2102
22	BSNL	Two Wheelers	56	22 - 4 - 2040
23	MARUTI	Garments	42	15 - 11 - 2109
24	MARUTI	Baby Products	f6	6 - 3 - 2005
25	DELL	Garments	4b	28 - 8 - 2069

Figure 4. Sample output observed at the application end

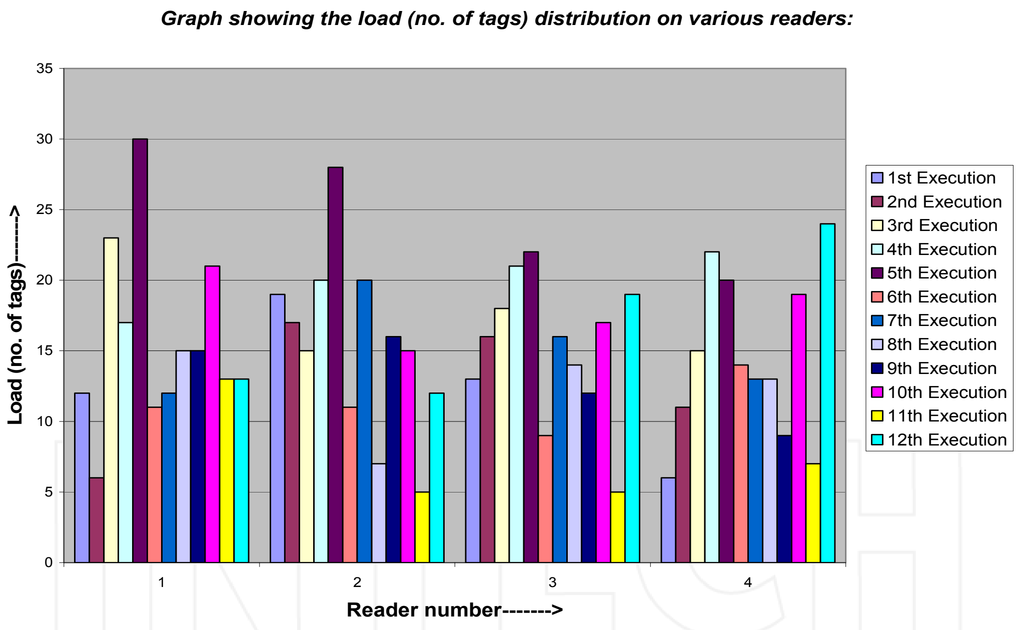


Figure 5. Graph showing the load distribution on readers

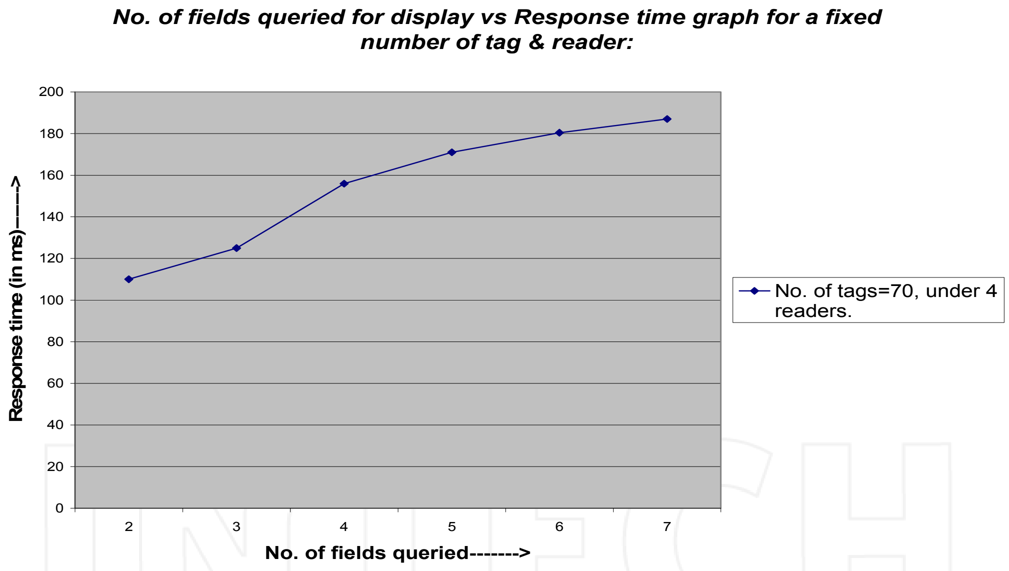


Figure 6. Graph showing the response time of the system for various queries from the application

In the second experiment we have considered the amount of delay between a query that has been submitted to the middleware system by the application and the requested information being available to the user at the application. Figure 6 shows the amount of delay in milliseconds. It can be observed, in general, that the amount of delay increases with the number of field information of the tagged items that need to be parsed at the server end.

In our third experiment we have calculated the time (in ms) required to identify the tags by the readers. Firstly we fixed the number of readers and then increased the number of tags and calculated the time to detect all the tags. Then we repeated the same experiment by increasing the number of readers and obtained the result as in figure 7. It can be observed that an increase in the tag population has a much greater impact on the response time rather than increasing the number of readers.

Lastly we carried out the experiment to find out the delay (in ms) to detect a fixed number of tags varying the number of readers in the simulation area. The same experiments have been carried out with different number of tags and figure 8 shows the experimental result. Here again, we see that abruptly increasing the number of tags suddenly increased the response time of the system, while the number of readers has little effect on this.

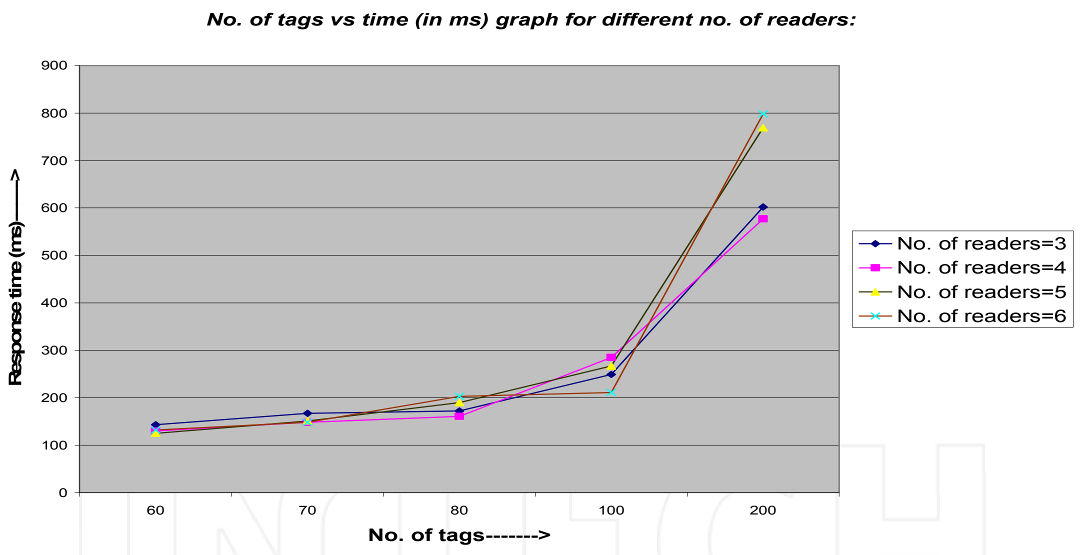


Figure 7. Graph showing the time required to identify the tags by the readers

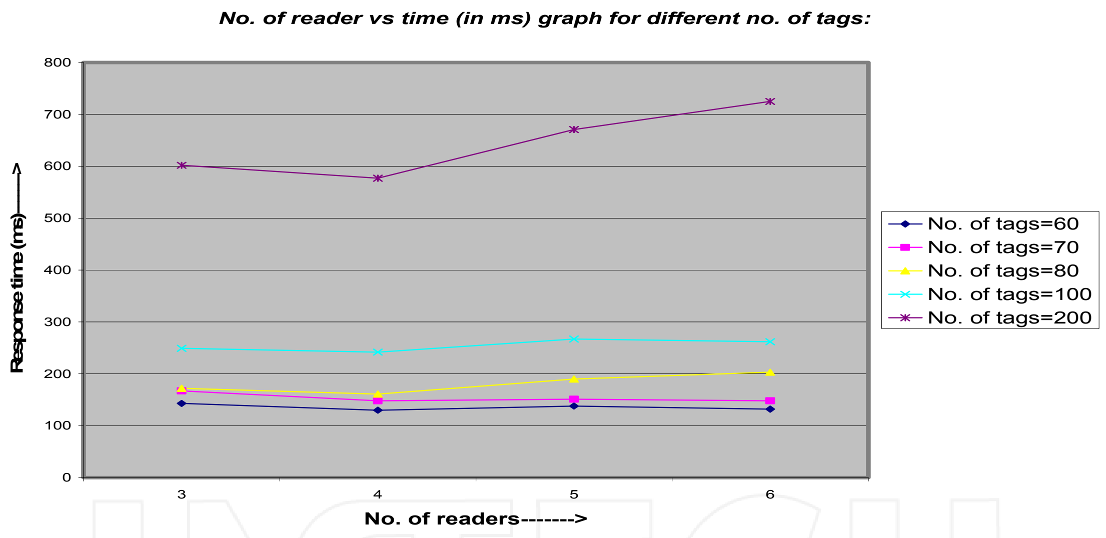


Figure 8. Graph showing the time required to detect a fixed number of tags by increasing the number of readers

5. Conclusion and Future Work

In our work, we have developed RFID middleware software that is more efficient as well as flexible compared to some of the other middleware solutions available today. To prove our point, we have chalked out some of the unique features incorporated in our system. These include the following:

1) Our system views the data obtained from a tag as a byte stream. It replies to the user's query from the server in the form of sequence of strings. The client accepts these strings and then parses them to take out the required information. This tag abstraction gives our system the freedom to choose amongst various types of tags available and make the system more robust.
2) The use of XML has helped the system to be more efficient as well as backward compatible to the existing HTML applications.
3) The system supports a large number of tags and readers.
4) The various fields can be selected for information retrieval just by a single click from the specially designed user interface. So, a novice also can easily use the system.
5) Reliability: The system uses processing rules to retrieve useful information from the raw data at the client side. So, if the information received from the server is not distorted while being transferred, i.e., if the XML-RPC is error-free then the system is completely reliable to provide the correct information.

6) Scalability: Only a single middleware is present that provides a common reader management interface. Complete reader and tag information is situated at the middleware. So the middleware is light-weight and efficient in terms of time and a number of readers and tags can be added over the existing applications without much effort.

7) Load Balancing: We have proposed a solution to load balancing that is quite simple to implement but equally effective. In the case of a conflict arising out of a particular tag falling into the vicinity of two or more different readers, the solution we have proposed is that the reader having a smaller load will have the right over this tag (i.e., will store its information). Load of a reader here means the total number of tags assigned to it at any instant of time. In case the multiple readers reading a particular tag have the same load, then the reader reading the tag first will own it. This solution has not yet been implemented in our system, but it can easily be added in future works.

8) Data Management: The information in our system is managed at a single location, the middleware; so it is a centralized data management system.

9) The security as such has not been considered so far, as our main aim was to device the communication of information from server to clients. But, in future works, appropriate security mechanism can be added to the system so that it can be treated as a full-fledged middleware solution.

10) The complete communication details have been kept hidden from the end-users, so that they do not get involved in the complex processes running behind such a simple front-end.

6. References

[1] R. Weinstein, "RFID: A Technical Overview and its Application to the Enterprise", *IT Professional*, Vol.7, pp.27-33, May 2005.

[2] B S Prabhu, X. Su, H. Ramamurthy, C.H.I.C. Chu, and R. Gadh, "WinRFID: A Middleware for the Enablement of Radiofrequency Identification (RFID)-Based Applications", Mobile, Wireless, and Sensor Networks: Technology, Applications, and Future Directions, 2006.

[3] "Sun Microsystems Inc.Sun Java System RFID Software", February, 2006. http://www.sun.com/ software/ products/ rd/ index.xml.

[4] B. Glover and H. Bhatt, "RFID Essentials", O'Reilly, 2006.

[5] "EPC Class 1 Gen2. EPCglobal Tag Data Standards Version 1.3", March 2006.

[6] K. Finkenzeller, "RFID handbook", Wiley Hoboken, NJ, 2003.

[7] C. Floerkemeier, C. Roduner, and M. Lampe, "RFID Application Development with the Accada Middleware Platform", IEEE Systems Journal, Special Issue on RFID Technology, Vol 10, December 2007.

[8] A. Brewer, N. Sloan, and T.L. Landers, "Intelligent Tracking in Manufacturing.", Journal of Intelligent Manufacturing", vol.10 pp. 245-250, March 1999.

[9] Jian Feng Cui and Heung Seok Chae, "Developing Load Balancing System for RFID Middlewares Using Mobile Agent Technology", LNCS 2007, Vol. 4496/2007, pp. 757-764.

[10] Jian Feng Cui and Heung Seok Chae, "Mobile Agent based Load Balancing for RFID Middlewares", 9th International Conf. on Advanced Communication Technology, 12-14 Feb, 2007, pp. 973-978.

[11] Christian Floerkemeier and Matthias Lamp, "RFID middleware design – addressing application requirements and RFID constraints", Proceedings of International conf. on Smart Objects and Ambient Intelligence, Oct 2005, pp.219-224.

[12] Piyush Maheshwari, "A Dynamic Load Balancing Algorithm for a Heterogeneous Computing Environment", Proceedings of the 29th Annual Hawaii International Conference on System Sciences, 3-6 Jan 1996, vol.1 pp.338-346.

[13] Ashad Kabir, Bonghee Hong, Wooseok Ryu, Sungwoo Ahn, "LIT Middleware: Design and Implementation of RFID Middleware Based on the EPC Network Architecture", Proceedings of 18th ACM conf. on Information and Knowledge Management, 2009, pp. 221-229.

[14] Siti Zaiton Mohd Hashim, Mardiyono, Nurulhaini Anuar, Wan Mohd Nasir Wan Kadi, "Comparative Analysis on Adaptive Features for RFID Middleware", International Conf. on Computer & Communication Engg., ICCCE-2008, pp.989-993.

[15] Jameela Al-Jaroodi, Junaid Aziz, and Nader Mohame, "Middleware for RFID Systems: An Overview", 33rd International Conf. on Computer Software Applications, 2009, pp. 154-159.