Middleware for RFID Systems: An Overview

Jameela Al-Jaroodi, Junaid Aziz, and Nader Mohamed
College of Information Technology
UAE University, Al Ain, P.O. Box 17551, UAE
{j.aljaroodi, mjunaid, nader.m}@uaeu.ac.ae

Abstract
RFID-Based Systems have started to play a significant role in the automation of many business processes such as ERP systems and object tracking. Several efforts were made within the research community and commercially to enhance the development, deployment and utilization process of these systems. More importantly some research has also started in developing ways to use middleware to facilitate these processes. In this paper we provide an overview of some of the research efforts done and highlight the main issues and challenges of using middleware for RFID systems.

1. Introduction
A large number of organizations aim to reduce operational costs and increase productivity. However, this requires adapting the business models, enhancing the infrastructure and advancing the technology solutions used. RFID technology offers several advantages to businesses to streamline their work and optimize operations. RFID (Radio Frequency Identification) is generally used to trace tagged objects and is considered as a generation ahead of barcode systems. RFID tags allow for bulk identification of objects and could carry far more information about the tagged object than a barcode could, which allows for better organization and operations based on stored information. Because of the advantage of its reading flexibility, RFID is adopted in a wide range of applications [1,2,3,4]. Industries like Retail outlets, Pharmaceuticals and Manufacturers specially, have shown a great interest in the application of RFID technology into their day to day business operations.

The data collection, identification and processing requires some sophisticated software solutions to accompany the RFID system. In addition, as businesses grow and diversify, they require more complex software solutions to support their operations. As a result middleware has become an important component in many types of systems and RFID systems are no exception. Based on the report of Frost & Sullivan, ‘World RFID Middleware Markets’, the scale of RFID middleware markets is estimated as about 220 million dollars in 2011 [6]. RFID middleware usually sits between the reader and software application which uses the EPC [5] data. The main objective of RFID middleware is to collect large amounts of raw data coming from a heterogeneous RFID environment, filter them, compile them into useable format and dispatch them to the computer systems [7]. It is most likely used in the cases when the data needs to be shared at more than one location at a time, such as in supply chain systems where many readers are distributed across factories, warehouses and distribution centers. Generally RFID middleware systems are made of three Layers as shown in Figure1. Service management layer connects the system with the real world. The main job of this layer is to receive queries from the computer systems or other middlewares and filter the results from the data retrieved from the readers. The data management layer gathers the raw data, filters it, cleans it and puts it in a usable format for processing. The device management layer is responsible for managing and monitoring the readers deployed in the system.

![Figure1. RFID Middleware Architecture](image-url)
This survey provides a general overview of some of the most relevant RFID middleware systems and highlights the modern solutions and goals that still need to be achieved. The paper is structured as follows. Section 2 provides an overview of the current RFID middleware solutions. Section 3 provides the analysis of these solutions and highlights the open research challenges. Section 4 discusses some related work and Section 5 summarizes the contributions of the paper.

2. RFID Middleware

According to Cheng-Ming [8], developing an RFID-enabled application with real time data processing and process decision-making capability is a complicate effort. It requires meeting the following challenges, where such application must be:

- Flexible to accommodate future changes
- Flexible to accommodate on-going business rules updates
- Flexible for process intelligent evolutions
- Capable of integrating with other new technology
- Architecture efficient to be able to process high volumes of data intelligently and in real-time
- Interoperable with variety of devices and systems
- Highly reliable for mission critical usages
- Scalable according to the business demands
- Secure thus protecting the data and supporting access control
- Compliant with on-going standards

Considering the above challenges, the design of an RFID middleware should be capable of addressing issues such as reliability, scalability, load balancing, security and data management. In light of these issues, the remainder of this section surveys most relevant middleware platforms proposed by different researchers.

2.1 Multi-agent Based RFID Middleware

Multi-agent based RFID Middleware (MARM) [9] is designed using the concept of Agent Oriented Software Engineering (AOSE) which is a new paradigm in software engineering and currently being practiced in a wide variety of applications. The main motivation behind designing MARM was to design a middleware for processing and managing data produced by RFID systems for an asset management application used by Central University of Technology (CUT) in South Africa. MARM uses SMURF [10] for smoothing the unreliable data captured by RFID readers and a temporal data model proposed in [11] to facilitate application event processing. MARM architecture was implemented using PASSI methodology [12], which is one of the AOSE methodologies for designing and developing MAS (Multi Agent Systems). The architecture is divided mainly in three layers which are device management layer, data management layer and interface layer. The client applications at the interface level specify the data they wish to receive and the kind of data they want to be reported by communicating with the data management agents such as ECReport-Request manager and ECReportSpec Manager respectively. Data management agents in turn communicate with the appropriate reader agents in the device layer, collect the raw data, process it, generate the requested reports and send them all back to the client application. Two ontology class diagrams have been modeled to achieve the objective of successful and timely coordination among agents, one of them is Domain Ontology Description (DOD) and the other is Communication Ontology Description (COD). DOD describes the ontology of middleware design by representing predicates of the domain and some agent actions. COD describes the ontology about the knowledge of agents and the association between the communicating agents. MARM is designed to provide not only the functionalities such as data capturing, smoothing and event processing but also the processing of application level events within the middleware.

2.2 Accada

Accada which has been renamed Fosstrak [13] is an open source RFID platform. Its implementation is based on EPCglobal Network Specifications. Fosstrak is developed after an extensive research work done at ETH Zurich. It provides an RFID infrastructure to the RFID Community benefiting the researchers and application developers with the implementation of low-level message bindings with EPC Network components. This fosters the use of EPC Network and availability of modifiable software components according to their requirements rather developing them from scratch. Fosstrak platform consists of three modules: reader, filtering and collection middleware, and EPC Information Services (EPCIS). Reader module performs filtering, synchronous/asynchronous data dissemination as specified in EPCglobal reader protocol and reader management. It can be used in three different modes. In the first mode it wraps a proprietary RFID reader protocol and runs on a separate server. In the second mode, Fosstrak reader is used in simulation mode in which it will make use of...
its built in simulated RFID readers belonging to different vendors. This can be handy especially in the case no physical RFID reader is available. In the third mode, Fosstrak reader implementation is deployed on the RFID reader itself to provide data filtering and dissemination capabilities. Once the readers capture the relevant tag data, they notify the Fosstrak filtering and collection middleware, which combines the data arriving from different readers in a report that is sent to the subscribed applications according to a predetermined schedule. The interface between filtering and collection middleware and a host application is based on EPCglobal Application Level Events (ALE) specification [14]. Since it also provides the aggregation functionality, it can omit the redundant read events from different readers originating from the same location. EPC Information Services (EPCIS) [15] is a component responsible for receiving the RFID data from the filtering and collection middleware, translating them into business events and making them available to the application. It comprises of three different modules: EPCIS repository, EPCIS capture module which receives the RFID data and EPCIS query module which provides the facility to retrieve events from the repository. The current Fosstrak implementation does not provide persistence for RFID data in the filtering and collection middleware module and it provides limited support for proprietary reader protocols.

2.3 WinRFID

WinRFID [16] is an RFID middleware, designed and developed at The University of California, United States. Its architecture is based on five layers: RFID hardware layer, protocol layer, Data processing layer, XML framework, and Data presentation layer. The RFID hardware layer handles the deployment, configuration and communication related issues with readers, tags and the I/O module of its readers. There are three components in this layer: reader, tag and I/O. The protocol layer abstracts the most common reader-tag protocols such as ISO 15693, ICode, EPC Class 0 and EPC Class 1. It runs a protocol engine whose responsibility is to parse and process the tagged data in the raw format as per any of the standard protocols mentioned earlier. The data processing layer implements the rules to process the raw data coming from the protocol layer by ways of removing the duplicate reads and verifying the tag reads. Any abnormality at this layer is handled through different forms of alerts such as emails, messages and user defined triggers. The clean and formatted tag data then gets forwarded to the xml framework layer to be presented in xml format to pave the way for this data to be used by variety of different enterprise applications as well as warehouse management and supply chain management systems. The data presentation layer is an application layer which uses the data supplied from the xml framework layer for visualization and decision making. WinRFID is designed using the .NET Framework which facilitates adding functionality to the applications by means of runtime plug-ins. To reap this benefit, WinRFID has an adaptable rule engine therefore end users can easily add their own rules by way of plug-ins.

2.4 Hybrid Middleware

Hybrid middleware [17] is a middleware based on group communication in peer-to-peer (P2P) networks. It is purely designed for an electronic parking management system (EPMS) in a University campus. Hybrid middleware architecture consists of overlay groups of peer nodes such as peer group membership manager, group communication coordinator, parking event manager and service connectors. Group membership management such as create or destroy or change group membership is handled by the group membership manager node. The group communication coordinator node ensures the affective and timely coordination among the member nodes so that any change in the parking status gets detected with minimum delay. Forwarding all parking events to the database service is the job of the parking event manager node. Service connectors perform the management control of individual nodes. Hybrid middleware is currently designed to handle two type of parking events: ENTERING and LEAVING which occur when a vehicle enters the parking slot and when it leaves the parking slot respectively.

2.5 RF²ID

RF²ID (Reliable Framework for Radio Frequency Identification) [18] is designed and implemented with the aspiration of providing a middleware with traits such as reliability, load balancing, high throughput, scalability and data organization. The architecture of the middleware is built on the concept of virtual reader (VR) and virtual path (VP). Each VR is associated with a set of physical RFID readers (PR) in its geographical region. The main responsibilities of each VR are the following: filtering the data coming from the connected PRs and from other VRs, timestamping the data, virtual path management and query management. Virtual Paths (VPs) are logical channels dynamically created by VRs in order to provide functionalities like high scale reliability of the data, efficient query responses and load balancing among the readers.
2.6 LIT Middleware

Logistics Information Technology (LIT) [19] is a middleware addressing both application needs and the constraints of RFID technology. It is designed using the concepts of Application Level Events (ALE) and EPC Information Services (EPCIS) proposed by EPCglobal. ALE software architecture of the middleware consists of four layers: Application Abstraction, State-based Execution, Continuous Query and Reader Abstraction Layers. In order to achieve high performance the state-based execution model [20] has been used in this architecture. The EPCIS software architecture consists of three layers: Query Service Layer, Repository Layer and Capturing Service Layer. The Data Source Access Component at the repository layer has been designed to provide persistence and connections to different databases such as Oracle, MS SQL Server, and MySQL etc.

2.7 REFiLL

REFiLL [21] is a lightweight filtering and collection middleware platform based on EPCglobal network architecture. It is designed as a programmable environment, which provides flexibility to the developers of RFID applications in the presence of heterogeneous hardware platforms and anecdotal requirements. REFiLL is positioned between the reader virtualization layer and the Application Level Events (ALE). REFiLL engine comprises of filters and output managers. Filters are used to receive input from the readers or other filters. Each filter has the following attributes: input ports, output ports and event processors. Filters are defined and interconnected with the help of editable XML-based markup files which are then compiled into java code to provide the functionality of collection and filtration of RFID events. Output managers provide the functionality of receiving the processed data from the filters and abstractions to the output systems such as RDBMS, Files, Sockets and Web Services.

3. Analysis and Open Issues

Because of its cost effective and cutting edge features, most of the industries started using RFID technology which gave birth to many issues related to data processing, performance and system reliability in general. In the past and even now researchers are trying to find the most appropriate and effective techniques to overcome these issues which are discussed in this section. On the other hand due to its rapidly growing market some more issues are arising which still remain unresolved or missing an appropriate solution. In this section we also discuss the most important challenges faced by RFID middleware and ways to cope with them.

Most of the middleware solutions discussed in Section 2 are heavily relying on finding ways to handle the processing and management of raw data coming from RFID readers effectively. This is achieved after applying the following steps: removing the duplicate reads, combining the data coming from different readers and efficient data dissemination. Since a typical RFID middleware handles large amounts of data coming from numerous RFID readers and shares it with the enterprise applications, it is necessary to have a reliable system in place which ensures an uninterrupted flow of correct and required RFID data. This is achieved by using different event management techniques. Based on our findings we are providing the comparison of the middleware solutions under discussion in Table1 related to their scalability, reliability, load balancing, security and data management.

A typical RFID network can have hundreds of RFID readers attached to it while in some cases this number can increase to thousands. Data coming from all of these readers will gather at a particular point and thus would create a large volume of data. Handling such amounts of raw data can pose a serious challenge to the performance of the middleware. Therefore in order to ease this process a distributed network technique is clearly required in which each node is responsible of gathering the raw data from its designated data source according to its preset load limits. Handling such a large amount of data also poses security threats such as unsafe data processing and privacy threats such as using the RFID data for malicious activities. Threats can also be aroused by humans or the environment in management and technical areas as discussed in [22]. These arguments clearly justify the need to have a middleware system with a protocol level security and message level security. The example of such system is proposed in [23] which deals with message level security such as client authentication, access control mechanism, and security of the messages transported between the enterprise applications and the data providers.

4. Related Work

RFID and its applications have been used for a while now in industry and have proven to facilitate several business processes and simplify and speed up many operations that used to be time difficult and time consuming. For example, in warehousing and inventory control, RFID systems have simplified object registration and tracking throughout its journey within and outside the organization.
Table 1. Comparison of RFID Middleware

<table>
<thead>
<tr>
<th>RFID Middleware</th>
<th>Reliability</th>
<th>Scalability</th>
<th>Load Balancing</th>
<th>Data Management</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARM</td>
<td>SMURF (Statistical Smoothing for Unreliable RFID data)</td>
<td>Multi Agent Systems</td>
<td>Not Addressed</td>
<td>Temporal data model for RFID data</td>
<td>Not Addressed</td>
</tr>
<tr>
<td>Accada</td>
<td>Implementation of EPC Reader Protocol</td>
<td>Reader implementation in three modes: on a separate server, simulation mode, on RFID Reader</td>
<td>Subscription of Readers</td>
<td>EPCglobal Application Level Events Specification</td>
<td>Subscription feedback mechanism</td>
</tr>
<tr>
<td>WinRFD</td>
<td>Processing Rules</td>
<td>Distributed Middleware Modules</td>
<td>Not Addressed</td>
<td>Data persistence and Customizable business rule engine</td>
<td>Authentication and Access Restriction</td>
</tr>
<tr>
<td>Hybrid Middleware</td>
<td>Peer-to-Peer Group Communication Model</td>
<td>Multi-ring Peer-to-Peer Network</td>
<td>Peer-to-Peer systems</td>
<td>Large-scale decentralized event notification infrastructure</td>
<td>Node subscription level only</td>
</tr>
<tr>
<td>RF2ID</td>
<td>Virtual Readers</td>
<td>Virtual Paths between Virtual Readers and Physical Readers</td>
<td>Path Management</td>
<td>Name Server and Path Server</td>
<td>Not Addressed</td>
</tr>
<tr>
<td>LIT Middleware</td>
<td>Implementation of EPC Reader Protocol</td>
<td>Common Reader Management Interface</td>
<td>State-based Execution Model</td>
<td>EPCglobal Application Level Events Specification</td>
<td>Not Addressed</td>
</tr>
<tr>
<td>REFiLL</td>
<td>REFiLL Engine with implementation of EPC-RP (EPC Reader Protocol)</td>
<td>Lightweight programmable middleware framework</td>
<td>Not Addressed</td>
<td>REFiLL Filters</td>
<td>Not Addressed</td>
</tr>
</tbody>
</table>

It has become easy to locate, register and follow any object while it is being transferred or used. However, in the research community, the activities are not as intense. Several research teams started investigating the issues and challenges associated with the design, development and deployment of RFID systems. Yet quite few have approached middleware as an effective component to provide a spectrum of abstractions and features for RFID systems. Middleware can be used to support all levels of functionality in an RFID system starting from supporting data collection and management at one end and to facilitating application development on the other. A recent survey [24] was conducted comparing different research work done in academia and industry. This survey mainly focuses on comparing different RFID Middleware systems on the basis of data processing and data integration. It has considered a small scope of mostly commercial projects and did not cover many of the academic efforts in the area thus leaving some of the middleware systems such as Fosstrak and REFiLL behind. Our work is solely done by studying and comparing most relevant RFID Middleware systems proposed by different researchers on the basis of their scalability, reliability, load balancing, security and data management. There are some other RFID Middleware platforms available in the market, which are developed by different vendors such as SUN, BEA Systems and IBM. Since there are either very limited or no technical publications detailing these products therefore their comparison has not been included in this work.

5. Conclusion

Although RFID is not a new technology, research and development have just picked up the pace in the recent years due to its wide support and application domain especially in supply chain management. Researchers are still trying to find the solutions which can help the RFID Industry to deal with data management issues and performance related issues in a better and more efficient way. In this work we have tried to discuss different RFID middlewares proposed by different researchers. Based on our discussion we have concluded that a scalable, robust, efficient and most importantly secure RFID Middleware is required to meet the challenges faced by RFID Industry.

Acknowledgments

This work is supported by UAEU Research Grant # 01-04-9-11/08.
References