Middleware Architectures for RFID Systems: A Survey

Haitham S. Hamza, Mohamed Maher, Shourok Alaa,
Aya Khattab, Hadeal Ismail, Kamilia Hosny
ANSR Lab, Cairo University
Giza, Egypt
Email: {hhamza, mmaher, salaa, akhattab, hismail,
khosny}@ansr.cu.edu.eg

***Abstract* — Radio Frequency Identification (RFID) technology has advanced considerably over the last decade, and has become one of the dominate technologies to realize emerging Internet of Things (IoT) applications. The increasing demand for adopting RFID coupled with the diverse types of RFID systems (e.g., readers and tags) gave rise to the challenging problem of integration of heterogeneous RFID systems. Accordingly, RFID middleware technologies have received an increasing attention in both research and industry community. This paper reviews existing main RFID middleware systems and compares their main features. Observations regarding the capability of existing middleware systems are also discussed.**

Keywords-RFID; Middleware; Semantic middleware; Interoperability, Internet of Things (IoT)

I. INTRODUCTION

Auto-identification technology has widely emerged during the last few years due to the need for identifying (people, things) in many applications. Radio Frequency Identification (RFID) is a wireless identification system based on electromagnetic field (Radio Waves) to transfer data [1]. Recently, RFID technology has been widely used in various domains and applications including: supply chain, retail management, infrastructure and asset monitoring. The wide-spread of RFID usage can be attributed to its several advantages, such as: no line-of-sight needed, simultaneous and bulk readings, ability withstand environmental conditions, and possibility to read/write on tags. RFID consists of two main components: *Tags (transponders)* and *Readers (transceivers)* [2]. Figure 1 illustrates the typical components of RFID systems.

In RFID systems, tags can be classified into three main types; namely, *passive* **(**also known as pure passive, reflective, or beam powered), *Semi-passive/Active*, and *Active.*

Passive tags obtain their operating power from the reader as the reader sends electromagnetic waves that induce current in the tag's antenna. The tag in turn reflects the RF signal transmitted and adds information by modulating the reflected signal.

Semi-passive/Active tags are powered by internal batteries that are used to run the microchip's circuit and to broadcast a signal to the reader; generally ensure a longer read range than passive tags.

Active tags make use of a battery to maintain data in the tag or power the electronics that enable the tag to modulate

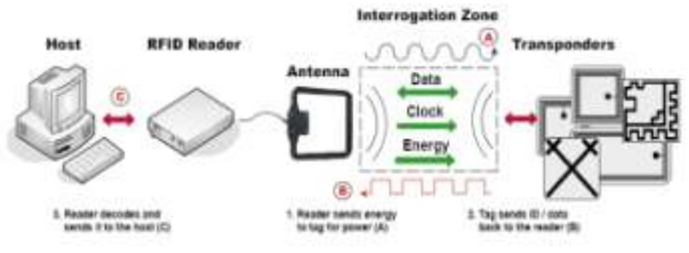
Interrogation Zone
Host
RFID Reader
Antenna
Transponders
A
C
Data
Clock
Energy
B
3. Reader decodes and sends it to the host (C)
1. Reader sends energy to tag for power (A)
2. Tag sends ID / data back to the reader (B)

Figure 1. Typical RFID Systems [15]

the reflected signal and communicate in the same method, as in the other passive tags, but has a wider range.

One variation among the various types of tags is in the *coverage area*, which ranges from few feet in passive to several meters in active. Another aspect that differentiates among the various tags is their ability in terms of the read/write of data. In Read only tags**,** the memory is factory programmed and cannot be modified after manufacturing. Clearly, this type is cheaper compared to the read/write tags. In *Read/Write* tags**,** data can be written and read. Data on the tag can be dynamically altered, and hence, it is more expensive compared to the read-only chips. *Write Once Read Many (WORM)* is another type of tags where the data can be added once but never changed and can be read many times.

Despite the wide-spread of RFID technology, it still faces several challenges that prevent their full exploitation in emerging applications. Among these challenges is the heterogeneity of reader types and standardization of communication techniques. These challenges can greatly limit the usages of heterogeneous data in various applications. Accordingly, there was an increasing interest in the RFID technology to develop middleware systems that allow for communication across various RFID readers without the need for changes or upgrades in the core of the middleware.

The increasing challenges that resulted from the increasing diversity in RFID technologies have led to development of several RFID middleware systems. Accordingly, in this paper we attempt to survey existing RFID middleware systems developed in various research projects and compare their features and capabilities. This survey does not cover middleware systems that target the integration of sensor devices, or those that handle only a limited set of features for data or communications as Bitmap

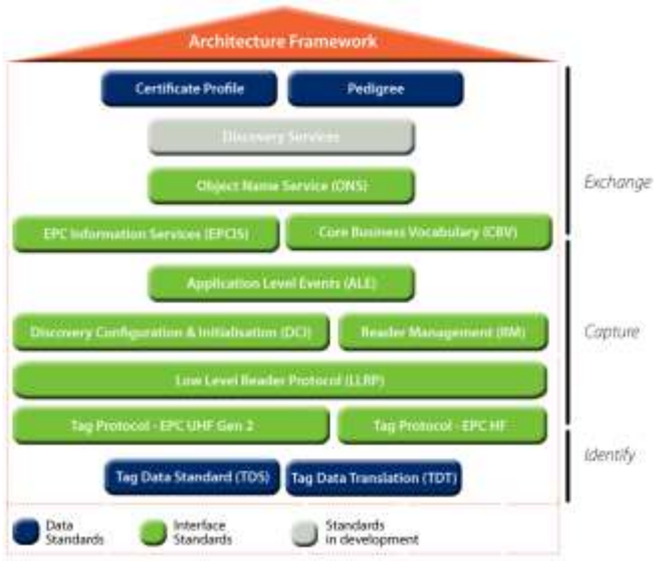
Architecture Framework
Certificate Profile
Pedigree
Discovery Services
Object Name Service (ONS)
EPC Information Services (EPCIS)
Core Business Vocabulary (CBV)
Application Level Events (ALE)
Discovery Configuration & Initialisation (DCI)
Reader Management (RM)
Low Level Reader Protocol (LLRP)
Tag Protocol - EPC UHF Gen 2
Tag Protocol - EPC HF
Tag Data Standard (TDS)
Tag Data Translation (TDT)
Exchange
Capture
Identify
Data Standards
Interface Standards
Standards in development

Figure 2. The EPCglobal architecture framework [3].

[5], REFill [6], and SMURF [5]. Also, we exclude hybrid middleware systems that mainly focus on a single application in a closed specialized domain.

The rest of this paper will be divided as follows. Section II provides brief overview about the concept of middleware in the context of RFID systems. Section III reviews existing RFID middleware systems. Section IV presents a comparison between the existing middlewares and the concluded gap. Section V presents the conclusion.

II. RFID MIDDLEWARE OVERVIEW

The word “middleware” is typically defined differently by different RFID vendors. For the purpose of this work, we use the definition given in [3], where a middleware between two layers is defined as the intermediate layer responsible for facilitating communication between the two layers, and preparing output from the first layer as input to the second layer, and vice versa. Data is prepared through collection, filtration, and aggregation. Mapping this basic definition to the context of RFID, a middleware should address issues related to:

Heterogeneity of tags and readers: In typical real-life applications, installed readers are not all from the same vendor, and hence, they deal with different types and formats of tags (passive or active, readable or writable).

Tag/Reader collision: In typical operational environments, it is possible that repeated readings of the same tag or different tags are sent to the same reader causing missing reads

The diversity of applications: Different applications use different types and formats of data that are collected from tags.

The huge amount of collected data: Large amount of RFID data needs to be processed, stored or directed at once to their destination.

Lack of context: The context of operation is important in dealing with the collected data and their meanings.

Determining needed number of readers: It is important to identify the best locations suitable to install readers in order to ensure sufficient coverage suitable for the area under consideration.

Based on the above, we can deduce that a typical middleware may need to provide the following functionalities:

Hardware Abstraction: Dealing with different readers despite their different types/interfaces.

Duplicate removal: Discarding redundant readings.

Data Filtering: Obtaining only needed data from the incoming readings.

Data Aggregation: Collecting/Redirecting data to their destination (time based, location based, etc.).

Report Generation: Generating reports depending on some predefined actions.

Business Rules Compatibility: Storing needed data in the desired formats for further usages/processing.

Application Connector: Giving the facility to different applications to deal with RFID systems and get needed information despite their different formats (connector for each application type).

In terms of RFID Middleware and standardization problems, it is worth pointing that the EPC [4] global standards aims at supporting the use of RFID and standardizing its way of communication. The EPC framework is summarized in Figure 2. As shown in the figure, the following are the main layers in the framework: (i) The *Reader Management (RM)*: responsible of monitoring the health of RFID readers, (ii) *Low Level Reader Protocol (LLRP)*: overcoming the gap of not providing middleware providers with access to enough Gen2 air protocol details as much as needed, (iii) *Reader Protocol (RP)*: abstracting reader details and easier for application programmers to use, (iv) *Application Level Event (ALE)*: responsible of observing and reporting events (no business context included), and (v) *EPC Information Sharing (EPCIS)*: supporting capture and query interfaces for business to business communications.

III. EXISITING RFID MIDDLEWARE

In this section, we present the main RFID middleware systems and review their key features and capabilities.

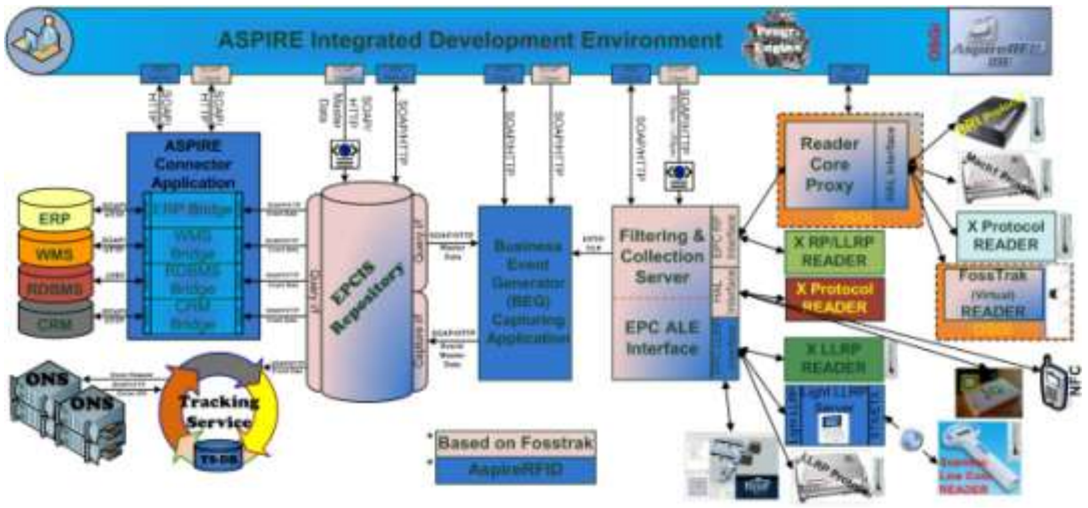

Figure 3. Aspire Middleware [7]

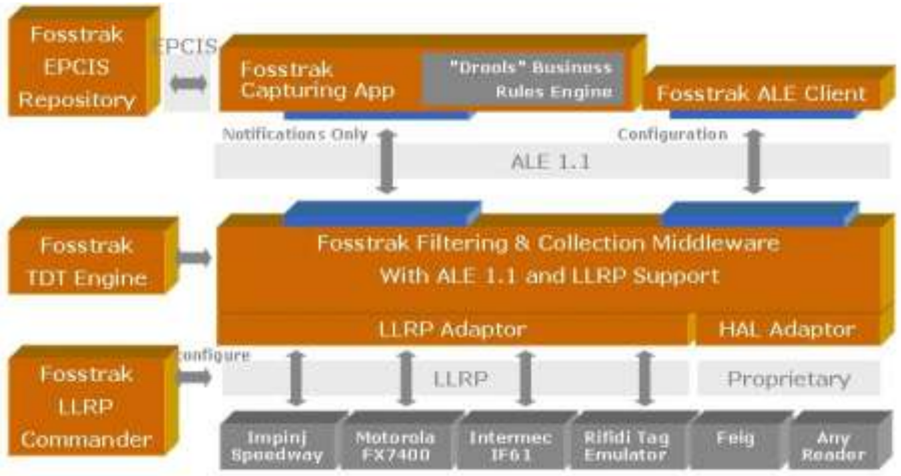

Figure 4. FOSSTrak Middleware [8]

A. *Aspire Middleware*

The structure of the Advanced Sensors and lightweight Programmable middleware for Innovative RFID Enterprise applications (Aspire) middleware is shown in Figure 3 [7]. Aspire consists of various layers as follows. The *Hardware abstraction layer (HAL)* unifying the way of interaction with multiple readers dealing and interacting with multiple protocols. Its implementation is divided into different modules (for reader simulators and one for each reader manufacturer). The *Reader Core Proxy (RCP)* layer is located between the readers and the ALE, and it helps in the communication between reader supporting protocol X and corresponding Filtering and Collection reader protocol interface (RP, LLRP). ALE layer converts data from its raw form to reports by collecting relevant information and creating reports that are being subscribed at by applications, *Business Event Generator (BEG)*, between the Filtering and Collection and Information Sharing. ALE layer can be seen as a specific instance of an EPC-IS capturing application that parses EPC-ALE reports. It fuses these reports with business context data using the assigned business event from the company's business metadata to serve as guide and accordingly prepares EPC-IS compliant events. *EPCIS* is the heart of the architecture carrying data to be shared, capturing events, and making them available to be queried by different applications. The last component is *Connectors* that abstract the interface between the ASPIRE Information sharing repository and the enterprise information systems.

B. *FOSSTrak Middleware*

The structure of the Free and Open-Source Software for Track and Trace (FOSSTrack) is shown in Figure 4 [8]. FOSSTrack consists of four separate modules: (i) EPCIS Repository that enables users to exchange EPC-related data with trading partners through the EPCglobal Network, (ii) *Tag Data Translation (TDT) Library* that translates one representation of EPC into another representation, (iii) Filtering and Collection Middleware with ALE and LLRP Support. It takes the EPC network role of data filtering and aggregation. It also provides report generation and generating events for the EPCIS repository, and (iv) *LLRP Commander* that describes an interface between RFID readers and clients that provide means to command an RFID Reader to inventory tags (read the EPC codes carried on tags), read tags (read other data on the tags a part from the EPCcode), write tags, and execute other protocol-dependent access commands (such as 'kill' and 'lock' from EPCglobal Class 1 Generation 2). However, the standard defines how to retrieve reader device capabilities and facilitate the addition of support for new air protocols.

C. ACCADA Middleware

ACCADA [9] consists of three separate modules. The *Reader* module implements the EPCglobal Reader Protocol, which includes collecting, filtering, time aggregates and space aggregates and also supports write on tags. The Accada reader implementation can be used in three different modes the reader implementation which is deployed on a *separate server* using the built-in HAL in simulation mode to facilitate testing of RFID applications and scheduling detection. It can also be deployed on an RFID reader itself to provide data dissemination, filtering, and aggregation capabilities.

The *Filtering and Collection Middleware* module allows applications to define a subscription and create a report that is sent according to a pre-determined schedule to the subscribed applications. The interface between the filtering and collection middleware and a host application is based on the EPCglobal ALE Specification.

The *EPCIS* is responsible for receiving data from the filtering and collection middleware, translating them into business events, and making them available. It consists of three parts: *EPCIS capture* application that receives the captured RFID data, an *EPCIS repository* that provides persistence, and *EPCIS query application* that is responsible for retrieving events from the repository. This module provides sample capture and query applications that implement the corresponding interfaces and EPCIS repository that uses a relational database to store the EPCIS events.

D. CUHK Middleware

CUHK [10] is a flexible and cost-effective solution for RFID network deployment and configuration which follows EPCglobal and the ALE specifications. CUHK is designed as J2EE application hosted in JBoss server and connected database with JDBC. Users can access the RFID network using ALE Interface extended to support two functions read and write into the tag memory. Through Management console user can configure, control, manage and monitor all readers in RFID network. CUHK provides five basic functions: (i) *Data Acquisition*, allows receiving EPCs from one data source to another, (ii) Collecting data in *time intervals*, (iii) *Filtering*, that eliminates duplicate data and filters the needed EPCs, (iv) *Manipulating* data to reduce the volume of data, and (v) *Report Generation* using ALE API which allows users to specify in a high level what data is needed and in which format, and generate ECReports for given Event cycle.

CUHK interacts with readers through *ReaderAdaptors* that interface with different readers. ReaderAdaptors performs tag reads and submits it to *ReaderManager*. They also perform reader registration and make sure that EPCs sent to the middleware are distinct by removing duplicated reads. CUHK currently supports four service endpoints to communicate with external users: *ALEService*, *TagDataService*, *ReaderManager*, *and Notifie. ALEService* and *TagDataService*, both are accessible as web-Service using SOAP over HTTP. TagDataService implements the CUHK‘s tag data read/write extensions to the middleware. *ReaderManager* is an EJB service endpoint that allows reader registration and aggregates tag reads for middleware through interfacing with *ReaderAdaptor*. *Notifier* is responsible for communicating with subscribers using HTTP or TCP. CUHK handles multiple tags’ reads simultaneously without performance impact by using two database instances in-memory which used to store tag reads and the other in disk.

E. DEPCAS Middleware

The Data EPC Acquisition System (DEPCAS) middleware is a general-purpose middleware inspired by the modern SCADA software architecture [11]. It consists of four main layers: (i) *Middleware Device Manager (MDM)* for Data acquisition and initial data processing, (ii) *Middleware Logic Manager (MLM)* for Data analysis and aggregation handling the transformation of raw data into generated information based on specific logic, (iii) *Graphical user Viewer (GUV)* for human monitoring and controlling, and (iv) EPCIS Repository for external business communication providing long term storage for EPC events.

F. Biztalk

Biztalk [12] is a Microsoft developed middleware consisting of the following main layers: (i) *Device Service Provider Interface (DSPI)* through which all devices communicate enabling device abstraction, (ii) *Event processing engine* that provides a platform for RFID business processes to execute and process tag-read events including filtering capability, (iii) *Object model (OM)* and *APIs* **that** provides APIs that helps to quickly design and deploy an end-to-end RFID process. The OM covers items as Device management, Process design and deployment, Event tracking, Health monitoring, (iv) *Designers, tools and adapters*, and (v) *various enterprise applications*.

G. LIT Middleware

Logistics Information Technology (LIT) [13] implements the concepts of both ALE and EPCIS layers of the EPC-Global standard. The *ALE* layer consists of four sub layers:

- *Application Abstraction Layer (AAL),*
- *State-based Execution Layer,*
- *Continuous Query Layer*, and
- *Reader Abstraction Layer,*

These sub-layers perform the base role of the ALE layer of grouping, filtering data, duplicate removal and hardware abstraction. The *EPCIS* layer, which represents the business layer and is the connection to applications.

H. Sun Java System RFID Software

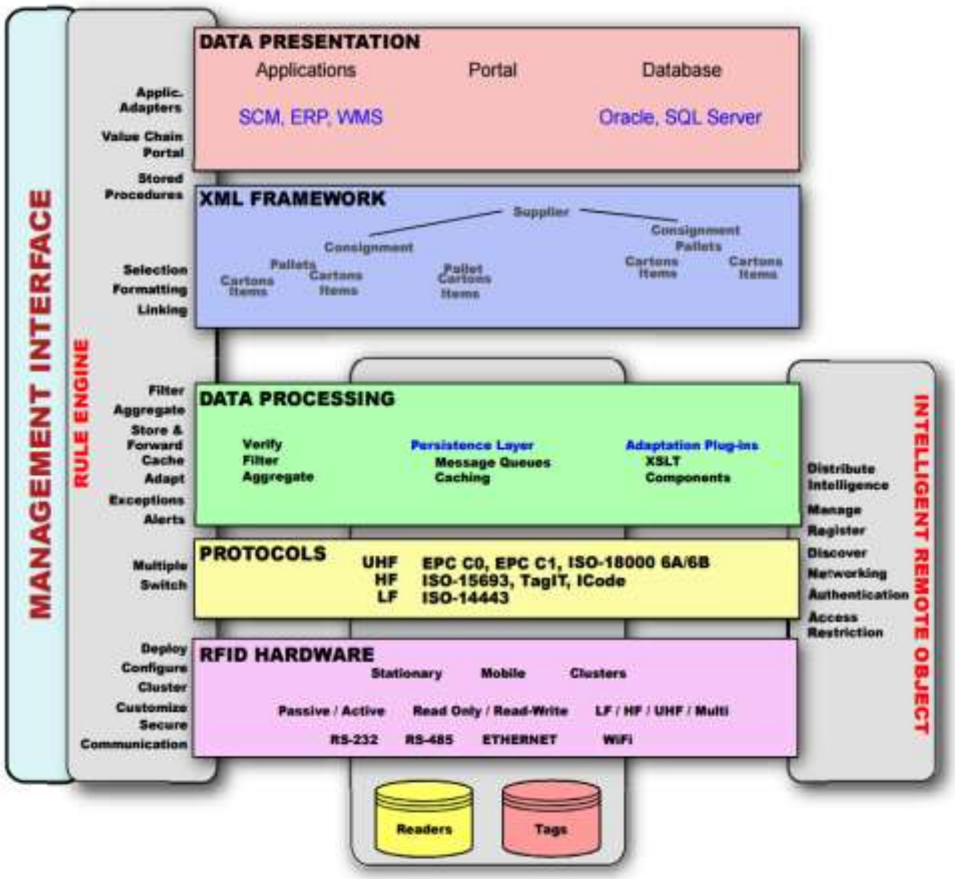

Figure 5. WinRFID Middleware [15]

Sun Java System RFID Software [14] is one of the first entrants into the market, designed by Sun Microsystems Inc. It provides a Java-based Middleware platform. The design of software conforms to the EPCglobal ALE software criterion that provides a high level reliability and scalability and also simplifying the task of integrating with multiple existing back-end enterprise systems. It consists of four components: (i) *RFID Event Manager*, which depends on Jini based system that facilitates capturing and filtering. Its main goals are to interface with readers, gather events, filter and feed relevant events to the RFID information system, (ii) *RFID Management Console*: is a browser based graphical interface used to manage and monitor the RFID Event Manager. It allows the user to control the readers, such as filters and connectors, (iii) *RFID Information Server*: that is a J2EE application that functions as an interface for capture and query of EPC-related data and also maps EPCs from low level observation into high level business function, and (iv) the *Software Development Kit (SDK)*: used specially for clients to be able to extend the product rather than using the components as they are shipped.

I. WinRFID

The main components of the WinRFID founded by UCLA (RFID research at WINMEC: Wireless Internet for Mobile Enterprise Consortium) are shown in Figure 5 [15]. It is developed on Microsoft .NET framework composed of five main layers each of different responsibilities: (i) *Physical layer* that deals with the hardware/readers, tags and other sensors, (ii) *Protocol Layer* that abstracts the reader-tag protocols, (iii) *Data processing layer* that process the data streams generated by the reader network and filtering them, (iv) *XML Framework* that handles data and information representation, and (v) *Data presentation* that presents data based on the requirements of the end-users or different enterprise applications.

J. SAVANT

SAVANT [16] is one of the early RFID models developed by Auto-ID Center. It can be considered as a data router that performs operations over the data received from readers such as capturing, monitoring, aggregation, and transmission. It consists of three main layers: (i) *Event Management System (EMS)*, (ii) *Real-time in-memory data structure (RIED)*, and (iii) *Task Management System (TMS)*.

Event Management System (EMS) provides a Java-based platform for different types of RFID readers. EMS is implemented on *Edge Savants (SE)*, which is connected to readers to collect data from tags. EMS consists of some components;

- Reader Interface,
- Reader Adapter,
- Event Loggers (or Consumer),
- Event Queues (or Forwarders), and
- Event Filters,

Real-time in-memory data structure (RIED) is an in-memory database designed to store event information generated by SE. Events sent by readers go into maintenance and organization by SE, then filtering and logging into the database using loggers. Events loggers in EMS need a database that can handle many transactions in a second. RIED does not really offer more transactions per second, but a better performance. In addition, it is easily accessible via applications like JDBC or even a Java interface. It also supports SQL common command and a subset of the data manipulation operations.

Task Management System (TMS) that serves as an OS for managing processes. It can perform different functions, send or receive product information to another middleware, schedule and remove tasks on other systems, and send product information to remote supply-chain management servers**.**

K. RF^{2}ID Middleware

RF^{2}ID stands for Reliable Framework for Radio Frequency Identification [17]. RF^{2}ID is a middleware designed to achieve scalability, reliability, load balancing and high throughput through the proposed architecture. Its idea is based on the concept of *Virtual Readers* (VR) and *Name Server* and *Path Server*.

Virtual Readers (VRs) are responsible each for a group of physical readers and *virtual paths* connecting VR, it performs multiple tasks such as:

- Data management (filtering and time stamping),
- Path management (overload management), and
- Query management,

Name Server and *Path Server* are responsible of keeping track of locations of physical readers and paths between them at any point of time.

L. FlexRFID

FlexRFID [18] is a three-tier architecture with *Back end application* layer, *Middleware* layer and *Hardware* layer. The hardware layer can be decomposed into four main layers:
(i) *Device Abstraction Layer (DAL)*: responsible for dealing with different devices and data sources regardless of their different characteristics. This is done through:

- *Data Source Abstraction Module (DSAM)* that provides standard view of data regardless of the data source protocol or air interface,
- *Device Abstraction Module (DAM)* that provides an interface to access different devices with functions as (open, close, read, write, etc.), and
- *Device Management and Monitoring Module (DMMM)* that loads and unloads libraries or reader adapters.

(ii) *Business Event and Data Processing Layer (BEDPL):* is located between the DAL and the AAL, and is responsible for duplicate removal, data writing, data filtering, data aggregation, data transformation and data dissemination.
(iii) *Business Rule Layer (BRL)*: is more like an authorization layer that grants or denies access to data, resources and services based on stored policies and rules
(iv) *AAL*: is responsible for facilitating communication between hardware and applications.

M. DeftRFID

DeftRFID [19] is a middleware system that is close to the FlexRFID middleware. It consists of three main layers: (i) *Application Interface Layer (AIL)*, which is responsible for application communications with physical world of readers and tags, (ii) *Data Processing Layer (DPL)*, which is responsible for data filtering, aggregation, transformation (based on stored rules), storing and querying, and (iii) *HAL* layer, which overcomes hardware diversity dealing with not only RFID sensors, but also other sensors and other devices (alarm, motor, etc.) supporting multiple interfaces. It also provides reader management through orders such as: activate reader, shutdown reader, read tags and write tags. In addition, it is responsible for duplicate removals.

This middleware provides four main advantages: location independency, dealing with different devices, maintenance cost reduction and scalability.

N. SmartRF

SmartRF [20] is an open source RFID middleware. It provides the applications to access and interact with Hardware devices. The system is divided into three subsystems: (i) *HAL*, (ii) *Event and data management layer (EDML)*, and (iii) *AAL*.

HAL is responsible to interact and access the RFID Hardware not considering their various characteristics. One of its main components is Device Management Module that is responsible for loading only needed libraries to avoid the extra weight of the unneeded libraries, also HAL provides some functions as OpenDevice, ReadDevice and WriteDevice.

Event and data management layer (EDML) is the intermediate layer between HAL and AAL processing commands and responses from AAL to HAL. Also it is responsible for grouping and filtering received data from readers.

AAL, the application level layer, works as an interface for RFID hardware through an API representing SmartRF services.

IV. Systems Comparison and gaps

Comparison shown in Table I between the different middleware systems is based on two main categories: EPC standard compatibility, and the general features for different systems.

The mentioned middleware systems attempt to confront the typical RFID challenges using different implementations for data filtering, data aggregation, and hardware layer abstraction.

From the shown table, it can be deduced that some of the middleware systems are concerned about following the EPC standard. It can also be seen that the most common middleware features found are duplicate removal, data filtering and data aggregation. It is also noticed that business rules compatibility and application connectors are lacking in several of the systems, mostly due to their complexity.

TABLE I: SUMMARY OF MAIN RFID MIDDLEWARE SYSTEMS

	EPCglobal Standards Compatible					Corporation	Open Source	Language	Duplicate Removal	Filtering	Aggregation	Report Generation	Business Rules Compatibility	Hardware Abstraction	Application Connectors	Database
	EPCIS	ALE	LLRP	RP	RM											
ASPIRE [7]	Y	Y	Y	Y	Y	EU Funded Project	Y	Java	Y	Y	Y	Y	Y	Y	Y	Y
FOSSTRAK[8]	Y	Y	Y	Y	Y	ETH Zürich Institute	Y	Java	Y	Y	Y	Y	N	Y	N/A	Y
ACCADA [9]	Y	Y	N	Y	Y	ETH Zürich Institute	Y	Java	Y	Y	Y	Y	Y	Y	N/A	Y
CUHK [10]	N	Y	N	N	Y	The Chinese University of Hong Kong	Y	Java	Y	Y	Y	Y	N	Y	Y	Y
DEPCAS [11]	Y	Y	N	N	N	ETS de Ingenieros Informáticos Universidad Politécnica de Madrid	N	N/A	Y	Y	Y	Y	Y	Y	N/A	Y
BizzTalk [12]	Y	N	Y	N	N	Microsoft	N	.Net	Y	Y	Y	Y	Y	Y	N/A	Y
LIT [13]	Y	Y	N	N	N	Research Institute of Logistics Information Technology	N	Java	Y	Y	Y	Y	N	N	N/A	Y
Sun Java System RFID Software [14]	N	Y	N	N	N	Sun MicroSystem inc.	N	Java	Y	Y	Y	N/A	N	N	Y	Y
WinRFID [15]	N	N	N	N	N	University of California	N	.Net	Y	Y	Y	Y	Y	Y	N/A	Y
Savant [16]	N	N	N	N	N	Auto-ID Center	N	Java	Y	Y	Y	Y	N/A	Y	N/A	Y
RF^{2}ID [17]	N	N	N	N	N	Georgia Institute of Technology,	N	C	Y	Y	Y	N	N	Y	N	Y
FlexRFID [18]	N	N	N	N	N	Research Paper	N	.Net	Y	Y	Y	Y	Y	Y	Y	Y
DeftRFID [19]	N	N	N	N	N	Fujitsu R&D Center Co.	N	N/A	Y	Y	Y	Y	Y	Y	Y	Y
SmartRF [20]	N	N	N	N	N	Department of Computer Science and Engineering - Indian Institute of Technology, Kanpur	Y	N/A	Y	Y	Y	Y	N	Y	Y	Y

The table also shows that there is a lack of standardization between RFID systems, leading to reduced system interoperability as the way/format data is stored can affect massively on further data processing. We believe that the concept of data unification through semantic annotation and ontologies [21] can be applied on the collected data in order to increase flexibility and interoperability.

V. Conclusion

This survey presented an overview about the various RFID middleware systems. The EPCglobal standard provides a holistic view of what needs to be provided by any RFID middleware system. Based on this standard and on the reference to the various middleware systems discussed in this paper, it could be concluded that most common features found in existing RFID middleware systems are data filtering, duplicate removal, data aggregation, report generation based on user's requests, the use of a repository for storing data and further processing, and abstracting the hardware layer so that they can deal with any device.

It is clear that various middleware systems target different applications, and hence, they have different features. However, there is no universal middleware that fits all potential needs of emerging RFID systems. The considerable diversity of protocols, reader technologies, tags, and data formats call for real vision on developing more flexible middleware systems for future RFID applications. We believe that the development of a suitable middleware for future RFID requires the use of new technologies, such as semantic; in order to provide the needed flexibility and agility that fits the emerging needs of RFID-based systems.

Acknowledgment

This research is supported by the EU-Egypt Innovation Fund- Grant Scheme 2- EuropeAid/132-759/M/ACT/EG. Grant No. RDI 2/S2/68.

References

[1] C. Jechlitschek, "A survey paper on Radio Frequency Identification (RFID) trends", [online] Available: http://www.cse.wustl.edu/~jain/cse574-06/ftp/rfid/index.html. [Accessed 10 September 2015].

[2] S. A. Weis, "RFID (Radio Frequency Identification): Principles and Applications", MIT Interim report, MIT 2003.

[3] J. Burnell, "What is RFID middleware and where is it needed?", RFID Update (2006), [Online] Available: http://www.vdcresearch.com/_documents/news/press-attachment-1235.pdf. [Accessed 20 September 2015].

[4] D. L. Brock, "The electronic product code (EPC): A naming scheme for objects", Technical Report MIT-AUTOID-WH-002, MIT Auto ID Center, 2001.

[5] Q. Sheng, X. Li and S. Zeadally, "Enabling Next-Generation RFID Applications: Solutions and Challenges", IEEE Computer, vol. 41, no. 9 (2008), pp. 21-28.

[6] A. P. Anagnostopoulos, J. K. Soldatos, and S. G. Michalakos, "REFiLL: A lightweight programmable middleware platform for cost effective RFID application development", Pervasive and Mobile Computing, Vol.5, no 1, Feb. 2009, pp. 49-63.

[7] Aspire Wiki, [online] Available: http://wiki.aspire.ow2.org/xwiki/bin/view/Main/WebHome. [Accessed 11 September 2015].

[8] F. U. Bes, "Implementation of Fosstrak EPCIS RFID System", Czech Technical University, 2012.

[9] C. Floerkemeier, M. Lampe, and C. Roduner, "Facilitating RFID Development with the Accada Prototyping Platform", the Fifth Annual IEEE International Conference on Pervasive Computing and Communications, New York, 2007, pp. 495-500.

[10] "CUHK RFID Middleware—System Design Document", Report No: RFID-SDD, Ver 1, 2007. [Online] Available: http://mobitec.ie.cuhk.edu.hk/rfid/middleware/doc/Middleware_SDD_v1.0.pdf. [Accessed 20 September 2015].

[11] I. A. Cardiel, R. H. Gil, C. C. Somolinos, and J. C. Somolinos, "A SCADA oriented middleware for RFID technology", Expert Systems with Applications 39, no. 12 (2012): 11115-11124.

[12] Microsoft, (2010), "BizTalk RFID Architecture", Microsoft Developer network, [Online] Available: https://msdn.microsoft.com/en-us/library/dd352563.aspx. [Accessed 11 September 2015].

[13] A. Kabir, B. Hong, W. Ryu, and S. Ahn, "LIT Middleware: Design and Implementation of RFID Middleware Based on the EPC Network Architecture", in Dynamics in Logistics, First International Conference, LDIC 2007, pp. 221-229.

[14] P. Chrobak, "Overview of RFID Middleware", Advanced information technologies for management, AITM 2010, pp. 73-86.

[15] B. S. Prabhu, X. Su, H. Ramamurthy, C. Chu, and R. Gadh, "WinRFID - A Middleware for the enablement of Radio Frequency Identification (RFID) based Applications", Wireless Internet for the Mobile Enterprise Consortium (WINMEC), Los Angeles, Dec. 2005.

[16] "The savant version 0.1 (alpha)", Technical Manual MIT-AUTOID-TM-003, MIT Auto ID Center, 2002.

[17] N. Ahmed, R. Kumar, R. S. French, and U. Ramachandran, "RF2ID: A Reliable Middleware Framework for RFID Deployment", Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, 2007, pp. 1-10.

[18] M. E. Ajana, H. Harroud, M. Boulmalf, and H. Hamam, "FlexRFID: A Flexible Middleware for RFID Applications Development", Wireless and Optical Communications Networks, 2009. WOCN'09. IFIP International Conference, 2009, pp. 1-5.

[19] Y. Lu, W. Zhang, Z. Qin, Y. Meng, and H. Yu, "DeftRFID: A Lightweight and Distributed RFID Middleware", Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Sixth International Conference (2010) pp. 181-186

[20] A. Ghayal, Z. Khan, and R. Moona, "SmartRF: A Flexible and Light-weight RFID Middleware", e-Business Engineering, 2008. ICEBE'08. IEEE International Conference, 2008, pp. 317-324.

[21] A. Maedche, "Ontology learning for the semantic web", Springer Science & Business Media, 2002.