IN-TECH
intechweb.com

The Design of a Lightweight RFID Middleware

Fengqun Lin[1], Bocheng Chen[1], C.Y. Chan[2], C.H. Wu[2], W.H. Ip[2], Andy Mai[1], Hongyang Wang[1] and Wenhuang Liu[1]

[1]Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China
[2]The Department of ISE, The Hong Kong Polytechnic University, Hong Kong
Corresponding author E-mail: jack.wu@polyu.edu.hk

Abstract: *Radio Frequency Identification (RFID) middleware is often regarded as the central nervous system of RFID systems. In this paper, a lightweight RFID middleware is designed and implemented without the need of an Application Level Events (ALE) structure, and its implementation process is described using a typical commerical enterprise. A short review of the current RFID middleware research and development is also included. The characteristics of RFID middleware are presented with a two-centric framework. The senarios of RFID data integration based on the simplified structure are provided to illuminats the design and implementation of the lightweight middleware structure and its development process. The lightweight middleware is easy to maintain and extend because of the simplified and streamlined structure and the short development cycle.*

Keywords: *RFID, lightweight middleware, ALE module replacement, data extraction, two centric framework*

1. Introduction

RFID, a non-contact automatic identification technology, can automatically identify targets by RF signals returned from tags attached to the targets, and relevant data can be obtained without any human intervention. It can read data over various distances, and can read and write different sizes of data from or to the tags without a line of sight. Most importantly, RFID technology can identify moving objects, can identify a number of tags at one time, and can provide a fast and convenient operation (Hu, 2006). These extensive abilities enable RFID to work in a variety of harsh and adverse circumstances.

The RFID system is mainly consisted of three parts, the labels (or tags), readers and the middleware. The middleware is used to relate different readers and enterprise applications. Most of the middleware available today is commercial-based and has been developed by the commercial players, for example, Microsoft BizTalk RFID (A Crimson Consulting Group, 2007), Oracle Fusion (Knifsend, 2005) and Sun RFID Middleware (RFID Update, 2005). There are also some middlewares which have been developed for research purposes, such as the Accda (Floerkemeier et al., 2007). Since RFID standards have not yet been unified, there are some differences between different RFID products manufactured by various equipment manufacturers. One of the RFID middleware's important tasks is to overcome these differences. The middleware must handle multi-protocols and all data processing tasks, such as filtering, grouping and duplication removal. It must guarantee that a variety of data can be read by the multi-readers, and the received data can then be extracted, decrypted, filtered, converted, and finally imported to the enterprise information systems in the form of information (Weinstein, 2005). Thus, the users or officers can monitor, browse, select, query and perform other tasks, based on the information, through the application program interface (Sun et al., 2007).

In practice, an RFID system always works in an environment with multiple readers and multiple protocols. In order to ensure the system can adapt to all working environments, RFID middleware should be used because it provides the applications with a device-neutral interface to communicate with different hardware. The middleware design must be full-featured, fully compliant with international standards, and can support simultaneous communication of multiple applications with multiple RFID hardware. EPCglobal standards have been widely adopted by RFID middleware products, which are globally accepted standards that ensure global applicability. EPCglobal Inc leads in the development of industry-driven standards for Electronic Product Codes (EPC) to support RFID. It is driving the global adoption of EPC as a global standard to enhance visibility of products. However, there are still some shortcomings found in the standards, for example, it is complex, bulky, expensive, and is lacking in advance system management. It seems that the standards are not suitable for domestic uses and small-sized and medium-sized applications. Simplicity and flexibility, fulfillment of domestic needs, and easy implementation should be included in the design of middleware for the small-sized and medium-sized applications.

Ngai et al. (2008) pointed out that bridging the gap between RFID research and application is one of the greatest challenges. In the cases of domestic applications,

it is more important to create a simplified middleware than to develop a more sophisticated RFID middleware for domestic users. Therefore, the study of light-weight middleware in this paper is not merely the trimming off of large-scale middleware, but is a software aimed at meeting the needs of small-sized and medium-sized domestic users. By taking its practicality into account, with easy development, a new processing architecture is created, and simplicity and practicality are the main focuses. In order to drive the application of RFID in small-sized and medium-sized domestic enterprises (as well as large enterprises), more of the aforementioned solutions are needed, such as applying light-weight middleware scenarios.

The design of the lightweight middleware in this paper is a kind of simplified middleware, which omits the ALE structure from the EPCglobal standards and retains the character and basic functions of the middleware, especially the cross-application of various types of readers. In order to have a simple and clear framework, a few commonly used RFID readers are involved in the simple management system. The input and output information is compliant with EPCglobal standards in order to constitute a convenient development framework. As long as the particular enterprises have staff with basic programming capability, they can develop their own lightweight middleware system, based on their specific needs, with the proposed framework.

2. Related Work

RFID middleware plays a very important role between enterprise application and data collection using RFID readers. It is a message-oriented middleware (MOM) software and the information is transferred, from one module to another, in the form of a message. Messages can be transmitted in asynchronous format, so the sender does not have to wait for a response. Therefore, the functions of RFID middleware are not only for message transmission, but also, sometimes, have to provide data packet's analysis and dissemination, security assurance, error recovery, network resources allocation, routing, sequence optimizing for message and enquiry sequences, as well as troubleshooting tools and so on (Ding, 2006).

The key technologies, mentioned by Clark et al. (2003), are the EPC code, Savant, Object Name Service (ONS), EPC Information Services, and Physical Markup Language (PML), have been supported by many universities and many companies. EPCglobal Inc. has also provided an RFID middleware software standard, namely Application Level Events (ALE). This mainly covers the criteria on the mapping between the location and the reader or antenna, the time interval for data collection, the packaging for the data collected and the format of the report statements.

The middleware, designed by Sun Microsystems Inc., is based on the basic framework of the EPC network (Clark et al., 2003). It is a completely P2P solution, which strengthens Sun's core technology. The design of the Sun Java System RFID Software is conforms with the EPCglobal ALE software criterion. The RFID system proposed by Microsoft has a hierarchical structure (A Crimson Consulting Group, 2007). It is made of several layers, such as the equipment layer, the data collection and management layer, the event and workflow management layer, and so on. IBM has proposed a light-weight RFID bus framework, the main focus of which is to enable isolation of the existing IT system from the RFID subsystem (RFID News, 2004). Oracle has designed an Oracle Sensor Edge Server embedded in Server 10g. It actualizes the data collecting, grouping, rule filtering, packing and routing, and organizes and manages the internal data queue before transmission (Goyal et al., 2006).

Wang et al. (2005) are studying the time performance of Real-Time RFID, under the leadership of McFarlane in the Cambridge University Auto-ID Center. They are focused on the study of event operation of the production, automation, control and time database. Researchers of the UCLA WINMEC RFID laboratory developed a kind of RFID middleware, called WinRFID. Its development was based on the Microsoft .NET framework, XML framework and SOAP for data and event representation supports interoperability. WinRFID provides simple intelligent data capturing, smoothing, filtering, routing and aggregation, and is mainly used in the demo system (UCLA WinMEC RFID Laborary, 2007). The IBM Haifa laboratory has designed a "situation manager". The "situation" concept, extends the concept of a composite event with expressive ability, flexibility and usability. It is a tool that includes both language and an efficient run-time execution mechanism. It can handle event collection, detection and consumption, by combining and observing the current system status and the event history. Its theory is based on the study of the patterns of system work (Adi & Etzion, 2004).

3. The Structure of the Middleware and the ALE Norms

The main task of the middleware is to manage and configure various kinds of RFID readers, and overcome the differences caused by different kinds of readers. The middleware can receive the tag data, filter and convert the data based on the given needs, and then transfer the date to the application layer program. The EPCglobal organization issued the Application Level Event (ALE) norms, namely the ALE specification (EPCgobal Inc., 2005). It is used to study the RFID middleware in terms of basic functions, and the management between the middleware and its upper application interface. The EPCglobal organization defines a group of standard interfaces for the middleware to the higher layer application system, as well as the most basic functions of RFID middleware, which includes insulation, collection and filtration, as shown in Figure 1. Currently, most of the RFID middleware research and development has followed the ALE specification.

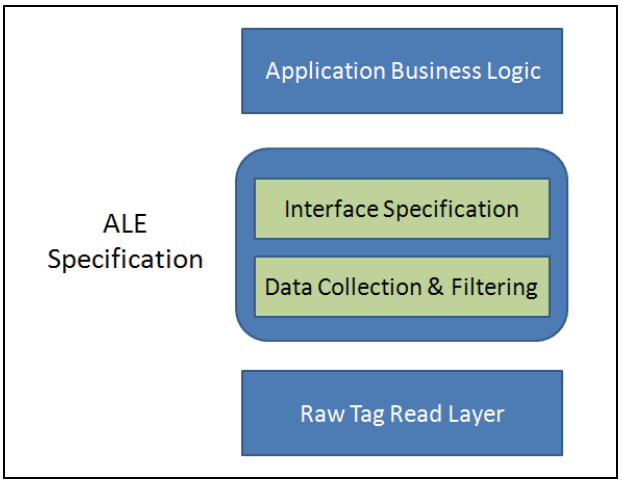

Fig. 1. The standard interfaces

In the ALE model, there are several basic concepts, such as Read Cycle (RC), Event Cycle (EC) and Report. RC is the smallest unit for the interaction between the RFID middleware and readers. An RC is a collection of EPCs which is related to the duration of a RC. The duration is related to the given antenna specification and its RF agreements. The output of the RC is the source data of the ALE layer. An EC can be one or more than one RC. It is the smallest unit of the interaction between the ALE and the users. EC can consider multiple readers as a whole unit because it treats readers from the user viewpoint. Report is defined as the data results which are provided by the ALE to the application layer (Liu, 2008), based on the EC previously defined.

4. The Design and Development of the Lightweight RFID Middleware

From the architectural aspects of the RFID middleware, it can be divided into two types, which are Infrastructure Centric and Application Centric. The first one is a RFID middleware centralized with the infrastructure. The second one is a RFID middleware centralized with the application program. A middleware designed based on the software specification, like ALE, is infrastructure-centric. The middleware is application-centric if it is designed based on the Application Programming Interface (API) provided by RFID reader supplier. This type of middleware is usually programmed in HotCode, picking up tag data directly from the given reader, and transferring them into an application program or database.

The lightweight middleware proposed in this research has the system architecture between the two types. In order to replace the ALE module, it has the EPCglobal structure and API-based program applications, with the assistance of the database integration and filtering method. Its advantages include a simple structure, easy development, and the systems and the development method being more practical.

4.1. The Structure and Implementation of the Lightweight Middleware

The RFID middleware is composed of the ALE criterion and API functions. In this section, the integration point, the temporary database, the filtering algorithm and the programming flow are discussed.

Integration Point: The characteristic of ALE is to provide a unified interface criterion. In the system realization level, it can be treated as a module with an engine, management console and interface. Each kind of adaptor has to be constructed based on the criterion and connected to the module, in order to enable the management of multiple readers and devices. In other words, ALE provides an integration point for the various kinds of hardware.

The Temporary Database : In order to eliminate the ALE module, a new integrated point for the RFID middleware system is created, and a MySQL database is built as the new integrated point for the data. Because the tag data can be easily read, the temporary database, in fact, is a database designed for temporary tag data storage. These tag data, received from all kinds of API transferring, should be filtered and updated on time, and then transferred to the main database for updating the content of the main database. Considering the attributions in the temporary database, the characteristics of the RFID readers and tags, and the main database requirements for the external tag data have been taken into account. In the temporary database, the tag's attributions can be simply described as reader No, antenna No, tag ID, read time and so on.

The Filtering Algorithm: The filtering function is mainly set to solve the problems of data chaos and redundancy in the temporary database, which result from the RFID characteristic, "read easily". In addition, once a tag is read, the tag's ID should not be lost in order to prove that the tag must have been presented in the reader's scanning area. The working principle of the system, which covers lightweight middleware, is shown in Figure 2 (reader-al means Alien RFID readers, and reader-mo means Motorola RFID readers).

The System Programming Flow Chart: The system is developed in the MySQL and C++ environment, and Motorola XR-440 Series readers have been used as an example to show the middleware composition, functions and procedures of the process. As a middleware, some features should also be provided, such as data conversion, however explanations on these features are not included in this article. The whole working process is shown in Figure 3.

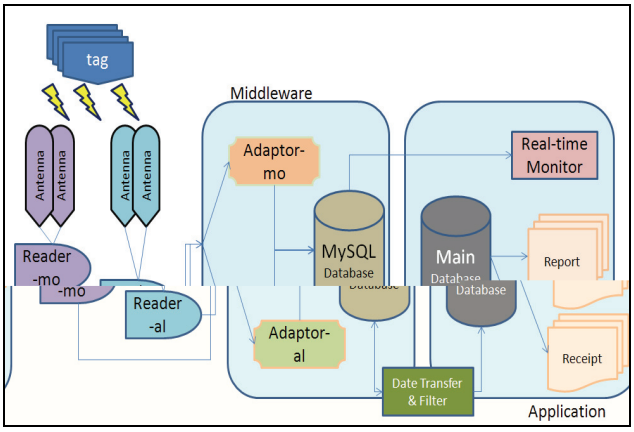

Fig. 2. The RFID system with the proposed middleware

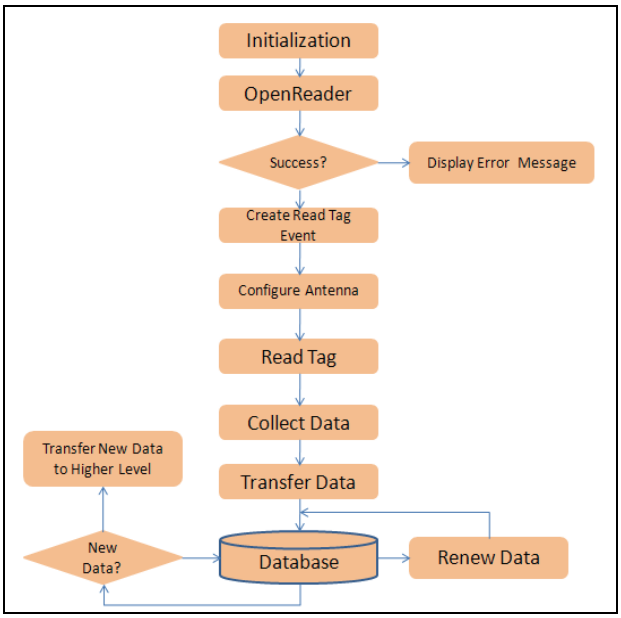

Fig. 3. The working processes without ALE module

The main function of the lightweight middleware can be described as follows:

1. To configure the accessed RFID reader and extract data from the reader
2. To collect data from the given reader, to write the data into the database, and to update the schedule-based data with the given filtering algorithm
3. To integrate different types of data received from different readers, and to complete the multi-protocol data conversion

5. The Programming Issues for the Lightweight Middleware

The Motorola XR-440 reader is taken as an example in the following. Also, the EPC Gen2 label is used, based on the EPC standards. The tag ID is 96-digit (bit code), so the tag ID can be shown in the form of a 24-digit-hexadecimal-string. In this study, Motorola's dynamic link library (DLL) functions are used to realize the data collection. The DLL file involves some interface functions, which include (1) To initialize a handle; (2) To configure IP address and TCP port for the reader; (3) Open a XR- 440 reader; (4) To configure the antenna; (5) To extract tag data from the reader buffer; (6) To close a XR-440 reader.
The realization process of the data extraction is shown in Figure 4. First, initialization is conducted, such as reading all kinds of initialization information, including the reader's IP, TCP port, the status of the antenna, in order to initialize the system. Then, a connection with the reader is established, so the reader can be recognized successfully. After that, the following operations are carried out three actions, including (1) initializing the list of tags (Taglist), which mainly distributes space for the tag list and resets all the variables; (2) configure the antenna by using the API function provided in the DLL, in order to enable it to read the tags; (3) the tags data can be read from reader's cache, transferred and displayed on the screen. The database loading operation can also be executed. The details of the realization process are illustrated in the sub-sections.

5.1. Initialization

Firstly, the IP address of the reader, which will be accessed, should be named before the working process. Thus, the computer can find the RFID reader by the IP address, and connect with it when the program begins to run. Besides the IP, there is some other information, such as the ON/OFF status of the antenna, the energy attenuation degree setting of the antenna, which should be identified. These parameters can be written into a file beforehand, for initialization, which means that the program will do the uniform initialization task. A specific example of the code is shown as following:

```
ifstream myfile(){
if myfile.open{
    getline(myfile, line);//IPAddress
    getline(myfile,line);
    //antenna1,2,3……
     getline(myfile, line);//power
}
myfile.close;
else printf(open_errormessage);
}
```

Fig. 4. The example code of initialization

5.2. Reader Connection

The operation process of the Reader Connection is the means to establish a connection with the assigned reader, and can be divided into three steps:
(1) To set TCP / IP parameters for the reader to access: The initialized data, read from the file, should be transferred to the reader by handles, and if it is not successful, a corresponding error message should be shown. The example code is shown as follows:

```
bool ConfigureTCPIP{
  status=configureTCPPort(Reader);
  if status=success
    status=configureIPAddress(Reader);
    if status=success
      return TCPIPsuccess=true;
    else printf(IP_errorMessage);
else printf(TCP_errorMessage);
return TCPIPsuccess;
}
```

Fig. 5. The example code of initialization

(2) To assign an API object for the reader : When assigning an API object for the reader and in setting the TCP / IP parameters, a successful message will be shown if the reader is found, and configuration of the reader can be undertaken afterwards. An error message will be shown if case of failure. An example code is shown as follows:

```
HANDLE OpenReader(){
 HANDLE Reader=0;
  if RFID_Open{
   if ConfigureTCPIP
    if RFID_OpenReader{
     printf("reader is found");
     ConfigureReader;
    }
   }
   else
       printf(API_errormessage);
   return Reader;
 }
```

Fig. 6. The example code of assigning API object

(3) To configure the reader: To identify the types of tags based on the reader types, corresponding settings for the program will be made. An example code can be shown as follows:

```
bool configureReader(){
if (!Reader)
    return false;
else
    maketagtypesenable(Reader);
return true;
}
```

Fig. 7. The example code of reader configuration

5.3. Tag-read-event

After reader connection, the system needs to create a tag-read-event to enable the reader to read the tags. Firstly, a tag-read-event name is prepared for the reader. Secondly, a tag-read-event is created. If successful, it returns to the event handler; if it fails, then returns 0. The example code is shown below:

```
HANDLE Event(){
   EventName=CreateEventName();
//Create a tag  read event name
   Event = CreateEvent();
   // create a tag read event
   if (!Event)
          return 0;
   status =Value(TagEventName);
   if (status != success){
          printf(event_errormessage);
          CloseHandle(Event);
          return 0;
   }
   return Event;
}
```

Fig. 8. The example code of creating a tag-read-event

5.4. Configuration of Antennas

To establish a list of antennas and set an initial value for each of them, the program will inquire into each antenna in order to access all tag data read by each antenna. The example code is shown in the following:

```
bool ConfigureAntennas(){
    antennasSuccess = FALSE;
    Antennalist[n];
    Antennalist[0] = 1;
    ……
    Antennalist[n] = 0;
    if(Value(Antennalist)== SUCCESS)
          antennasSuccess = TRUE;
    return(antennasSuccess);
}
```

Fig. 9. The example code of antenna configuration

5.5. Data Reading

First, the system should determine the time difference between the two scans, and estimate the duration of the tag which disappeared after the last read, and then the data stored in the reader cache will be read. If successful, various of tag ID information can be read, displayed and written in the temporary database. The example code is shown below:

```
void Read() {
……
Status = SUCCESS; //initiation
Status = ReadInventory(); //read
inventory of the reader
if (Status == SUCCESS)
   PrintTagList (); //print tag list on
to screen and insert tag id into
database
else
   printf(Read_errormessage);
}
```

Fig. 10. The example code of data reading

5.6. Display of the Data

To display the collected data on the screen after a successful tag read, there are usually two steps involved. (1) Data Update: The tag ID read is stored in an array, and a comparison of the new ID with the old ones after each reading operation is made. If the same ID is found, the tag's relative information will be updated, such as the read time, read count and so on. If not, a new tag ID array will be created with the relative information from the tag.

(2) Display on the screen: After each update, the real time data can be displayed on the screen, as mentioned before. The content to be displayed can be customized, such as: tag ID, reader No., antenna No., the last read time and so on.

5.7. Database Connection and Data Input to the Database

API functions provided in MySQL are applied to connect the temporary database and in composing the data extraction program. Firstly, the MySQL header file is included in the program, and the API function, *mysql_connect*, is used to connect the MySQL database. Secondly, the data read is written to the database through the functions, *mysql_query*, and *mysql_insert*.

The data written to the temporary database usually includes: tagID, reader, antennaNum, lastSeen. Among them, *tagID* means the I.D. of the tag; *reader* refers to the reader number, and *antennaNum* refers to the antenna number. These data will show the specific antenna of the specific reader which read the tag. The reader number and the antenna number may indicate the position and movement direction of the goods. *lastSeen* is responsible for showing the duration of a tag presented since the last detection. An example code is shown below:

```
mysql_init();
status=mysql_connect;
if status=success
mysql_insert(rfid, reader, antennaNum,
time) values (tagID, reader,
antennaNum, lastSeen);
```

Fig. 10. The example code of data input

After writing the data into the database, the tag information can be displayed by the MySQL Query Brower, and also can be extracted based on different needs. Table 1 is an example of the data in MySQL temporary database.

No.	Label ID	Reader	Antenna Num	Time
10	E2003411B802011207172573	01	01	2009-04-05 12:33:26
11	E2003411B802011207172564	01	01	2009-04-05 12:33:26
12	E2003411B802011207172575	01	01	2009-04-05 12:33:26
13	E2003411B802011207172565	01	01	2009-04-05 12:33:26
14	E2003411B802011207172561	01	01	2009-04-05 12:33:26

Table 1. The Data Retrieved form the MySQL Database

6. Conclusions

The design of a lightweight RFID Middleware structure and its implementation is reported in this paper. The lightweight design is mainly achieved by discarded the ALE structure of the EPCglobal. Although it is useful, it is too complex in most application cases, especially for domestic enterprises. The system structure and its working principles, especially the emphasized scenario of the new system integration point selection, are presented. Due to the absence of the ALE part, the system and its working principle are simpler and clearer than the traditional ones, and its new integration point for different kinds of RFID readers is feasible. Most enterprises will not purchase several kinds of readers for particular applications but a variety of models and standards may be found. Thus, in-house programmers can program their own lightweight middleware based on their specific needs following the steps and principles presented in this paper. This study reduces the complexity of RFID applications, provides guidelines for creating device-neutral development, and offers flexibility in middleware configuration. For example, the proposed design could facilitate the development of a tracking and tracing management system for exporting Shenzhen vegetables exported to Hong Kong. The system could be developed in a device-neutral and technology-independent development environment, based on the proposed lightweight middleware.

7. Acknowledgements

The authors wish to thank Tsinghua University and the Research Committee of The Hong Kong Polytechnic University for support in this research work.

8. References

Hu, Y.F. (2006). Survey on RFID technology. *Computer Era*, Vol. 12, pp. 3-4, ISSN: 1006-8228

A Crimson Consulting Group (2007). BizTalk Server 2006 Developer Productivity Study, Available from: http://www.microsoft.com/biztalk/en/us/r_d.aspx, Accessed: 2009-02-23

Knifsend, F. (2005). Oracle Fusion Middleware and Microsoft Interoperability: Addressing Enterprise-wide Needs, Available from: http://www.oracle.com/technology/products/middleware/pdf/fusion-middleware-microsoft-interoperability.pdf, Accessed: 2009-02-30

RFID Update (2005). Sun Releases 2.0 Upgrade to RFID Middleware. Available from: http://www.rfidupdate.com/articles/index.php?id=85, Accessed: 2009-04-25

Floerkemeier, C., Roduner, C. & Lampe, M. (2007). RFID Application Development with the Accada Middleware Platform. *IEEE Systems Journal*, Vol. 1, No. 2, pp. 82-94, ISSN: 1932-8184

Sun, J. & Chen, Q. (2007). The RFID Middleware Development in the World and China. *Logistics & Material Handling*, Vol. 2, pp. 98-101, ISSN: 1007-1059

Weinstein, R. (2005). RFID: A Technical Overview and Its Application to the Enterprise. *IT Professional*, Vol. 7, No. 3, pp. 27-33, ISSN: 1520-9202

Ngai, E.W.T., Moon, K.K.L., Riggins, F.J. & Yi, C.Y. (2008), RFID Research: An Academic Literature Review (1995-2005) and Future Research Directions. *International Journal of Production Economics*, Vol. 112, pp. 510-520, ISSN: 0925-5273

Ding, J., Li, J., Feng, B. & Guo, J.B. (2006). Survey on RFID Middleware. *Computer Engineering*, Vol. 21, pp. 9-11, ISSN: 1000-3428

Clark, S., Traub, K., Anarkat, D. & Osinski, T. (2003). Auto-ID Savant Specification 1.0, Available from: http://interval.hu-berlin.de/downloads/rfid/IT_Infrastruktur/6_WD-savant-1_0-20030911.doc, Accessed: 2009-04-20

RFID News (2004). IBM's Lightweight RFID Framework. Available from: http://www.rfidnews.org/2004/11/12/ibms-lightweight-rfid-framework, Accessed 2009-03-25

Goyal, K., Agarwala, M. & Genpact A. S. (2006). Using the Sensor Edge Server to Communicate to Sensor Devices, Available from: http://www.gouser.org/conference2006/presentations/Goyal%20Kirit%20-%20Using%20the%20Edge%20Server%20Whitepaper.pdf, Accessed: 2009-05-20

Wang, W., McFarlane, D. & Brusey, J. (2005). Timing Analysis of Real-time Networked RFID Systems, Available from: http://ivs.cs.uni-magdeburg.de/bs/tagungen/paper/rtn05S4_wang-RTRFID.pdf, Accessed: 2009-03-11

UCLA WinMEC RFID Laborary (2007). WINRFID RFID middleware system, Available from: http://www.winmec.ucla.edu/rfid/, Accessed: 2009-03-02

Adi, A. & Etzion, O. (2004). Amit-the Situation Manager. *The VLDB Journal*, Vol. 13, No. 2, pp. 177-203

EPCglobal Inc. (2005). The Application Level Events (ALE) Specification, Version 1.0, Available from: http://autoidlabs.mit.edu/CS/files/folders/13/download.aspx, Accessed: 2009-03-12

Liu, L. (2008). *Study of RFID Middleware for Product Anti-Counterfeitinging*, Beijing Industry University, MS dissertation, Beijing