RFID Middleware Framework for Ubiquitous Environment and Its Implementation

Young-Il Kim
RFID/USN Middleware
Research Team, ETRI
embroca@etri.re.kr

Tae-Su Cheong
RFID/USN Middleware
Research Team, ETRI
qlink@etri.re.kr

Joo-Sang Park
RFID/USN Middleware
Research Team, ETRI
kappa@etri.re.kr

Abstract

For widespread adoption for the RFID technology, RFID middleware is considered as the key enabler of the RFID evolution while it manages the flow of data between tag readers and enterprise applications and is responsible for the quality, and therefore usability, of the information. In this paper, we introduce the recent researches and developments of RFID middleware and explain the software framework of the RFID middleware which is currently developed in ETRI. In conclusion, our experiences with the current implementation are presented

1. Introduction

Ubiquitous computing is a new era in the evolution of computers. Since its introduction by Mark Weiser in late 1980's, the research direction has moved from an imaginary phase into a realizable research phase as the technology for system integration based on Internet has been rapidly evolved and the demands that all computer devices are seemly integrated into the environment and those integrations ultimately provide useful services to humans at any time and everywhere increase. Under the ongoing situation, Radio Frequency Identification (RFID) technology has recently gained a lot of attention as a technology to realize his vision, "ubiquitous computing". Moreover, the fact that the retail giant Wal-Mart and the U.S. Department of Defense issued mandates requiring their supplier to adopt RFID by 2005 accelerates the spread of the attention to the RFID technology and most business domains are considered as the deployment areas including supply chain. As with the Internet, RFID promises the drastic changes across the broad spectrum of business activities.

Until recently, RFID hardware devices like transponders and reader development have dominated the RFID market. However, the trend has shifted to RFID software and integration because RFID hardware without support of software systems that can aggregate data coming from multiple readers and pass it to back-end systems is of no use. The basic functions of RFID software starts with device monitoring and management. It extracts data from readers, filters and aggregates the information, and then sends it to enterprise system. Moreover, the market requires a kind of middleware software platform that provides the environment that the identified tag information is shared with internal or external applications, and manages the business process.

In this paper, we summarize the software requirements for RFID middleware platform and deliver the technical trends of RFID software. Also, we propose the RFID middleware framework and then introduce some aspects of our software implementation developed based on the framework.

This paper is organized as follows. In the Section 2, we present the software requirements of RFID middleware framework. Section 3 delivers general technology trends related with RFID. In the Section 4, we propose the middleware framework satisfying the requirements mentioned in Section 2 and introduces the RFID middleware software we developed. The final Section summarizes and presents the further work.

2. RFID Middleware Software Requirements

Generally, RFID systems consist of three main components: the RFID tag, which is attached to the object to be identified and serves as the data carrier, the RFID reader, which detects tags in the read range, and the RFID software system, which aggregates the data coming from tag, passes it to the back-end system and integrates the hardware devices with the back-end system. Again, in terms of the middleware software, the RFID software system is usually divided into two parts [1]. One is the RFID middleware, which works on the edge of the network unlike traditional middleware, manages the streams of tag data coming in from readers and links the devices to software infrastructure. The other is the conventional system integration middleware, which basically couples up a variety of applications which are distributed internally or externally with the enterprise infrastructure. As RFID technology is being adopted, the conventional middleware takes the role of routing tag data using different transport protocols, translating the data into the suitable format satisfying the application requirements and so on.

There are many capabilities in order to build up RFID software system and, especially, RFID middleware

framework capabilities include:

2.1. RFID Reader Management

There is a wide variety of RFID readers available in the market. Each one has its own device driver with data exchange protocol and supports various communication interfaces such as RS232, TCP/IP and so on. Also, some readers in the market meet the EPC-compliance requirements and some of others might be based on ISO standards. In the reasons, RFID middleware should support means to deploy, monitor and issue commands to readers via a general interface. In other words, the middleware hides the heterogeneity across RFID devices so that the upper layers have uniform access to device layers.

2.2. Data Handling

Basically, the RFID reader checks the existence of tags within the read range by periodic reading from hundreds to thousands of times per second. As a result, a large volume of data stream comes into the system, and there are a lot of noises and duplicate reads. Not all of the data may be of interest to the client applications, so filtering operation is required to eliminate such information that is either redundant or unnecessary. Additionally, a tag may not be seen for every read cycle of a reader because the RFID reader cannot report the tag read with 100% accuracy, so the tag reads must be smoothed.

2.3. Integration with enterprise system

The RFID middleware collects, filters and summarizes the data coming from tags and then should integrate the tag reads with back-end system - for example, SCM, ERP or WMS, etc - for the better business decision.

2.4. Share of data with partners and other applications

One of the ultimate goals by adopting RFID technology is to provide the ways to share the information about individual tagged object and related business descriptions with the trading partners and other applications. This leads to the real-time visibility for products over the supply chain and the improvement of business decisions.

2.5. Rule and exception processing

The industries expect that the adoption of RFID technology enhances efficiencies of activities across the business processes due to the characteristic of RFID technology - automatic identification. The RFID middleware provides the environment that users set the rules in order to generate the business dependant semantic events based on the tag reads, so it makes the RFID-based system to become a rule-based automatic business process system. Also, when the exceptional cases occur, the middleware system should cope with them by several methods like sending the alarm to an administrator's e-mail address or SMS to him.

3. Trends of RFID System Framework

In terms of the international standardization process related to RFID middleware, there are few ongoing activities except those by EPCglobal Inc., which is the de facto standards development organization for RFID system. EPCglobal Inc. mainly works on the RFID standards developments related to the domain like logistics and supply chain as long as the software perspectives are concerned and the possibility that the EPCglobal standards would be adopted as the de facto standards is quite high. Before moving onto the introduction of RFID middleware framework, we review how the RFID software standards has been evolved from Auto-ID Center to EPCglobal and get the idea of the overall framework in this section.

3.1. Auto-ID Center

The Auto-ID Center was founded in 1999 and is headquartered at MIT. The goal of its work is to build a worldwide standard and create building blocks that are needed for an effective infrastructure that no longer needs human interaction and that is accessible by the whole supply chain.

The infrastructure includes the following building blocks [2]: Electronic Product Code (EPC) that identifies each unique item, EPC tags that are attached to the product items, EPC readers that can detect these tags using radio frequency technology, Physical Markup Language (PML) that is XML-based language and is used to describe the products, PML Server that stores the product information represented by PML, Object Name Server that map the EPC to PML Server, and, at last, a middleware named Savant that connects all the pieces. The network infrastructure composed of the building blocks which are mentioned above is called as "EPC Network".

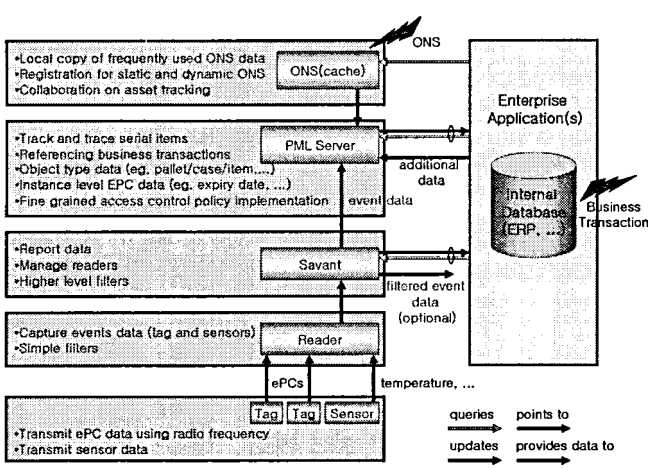

Figure 1. EPC Network Architecture (Auto-ID Center)

Figure 1 shows how EPC Network is organized and the

building blocks interact with each other and external applications. First, a tagged object is identified by a tag and a reader. EPC, the numbering scheme to identify each unique product instance, is written on the tag and the reader captures the code and sends it to the middleware called 'Savant' with the reader location information and the point of time when the tag is identified by the system. Savant filters and summarizes the tag reads, and sends the data to external applications including ERP and PML Server. PML Server stores the tag data and associated business description. It makes all data available in PML format to requesting services. At last, ONS provides a global lookup service to translate EPC into one or more network addresses of where PML server is located.

In the initial specification for the middleware [3], the savant server consists of an event management service (EMS), a real-time in-memory database (RIED) and a task management service (TMS) as internal components. RIED is an in-memory database which is optimized to achieve sort of performance gains by removing the complex SQL. The database is used to store tag reads event information for a short period. TMS performs the customizable tasks like data management or monitoring. EMS plays the key role in the savant server. It connects readers to applications by managing the event flow generated by the readers. In the later version of the specification [2], the focus moved from specific processing features into the interface between outer components.

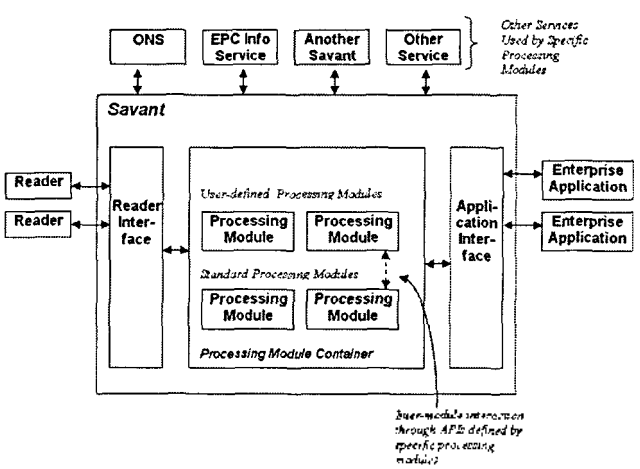

Figure 2. Savant Architecture

As seen in figure 2, Savant is a container of processing modules, which provides a specific set of features and may be customized to meet the needs for applications. All components including EMS, RIED and TMS are considered as the processing modules in this stage. The processing modules interact with outer services via two predefined interfaces – that is, Reader Interface and Application Interface. The Reader Interface provides the communication with the RFID readers and the Application Interface provides the connection to external applications including enterprise systems and other savant servers.

Most of commercial RFID middleware software systems which are currently available in the market follow the savant structure mentioned so far.

3.2. EPCglobal

EPCglobal, Inc. [4] is a joint venture company between EAN International and UCC (Uniform Code Council) to launch the efforts to drive global, multi-industry adoption of the EPC Network. All the research activities developed by Auto-ID Center moved onto the EPCglobal and its purpose is to commercialize the EPC RFID system.

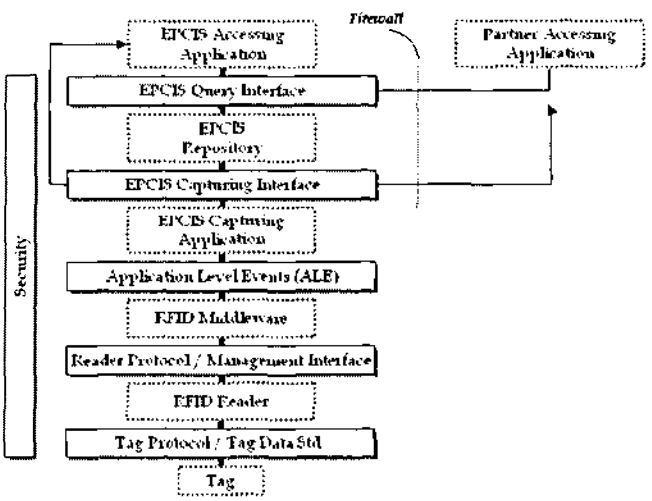

Figure 3. EPC Network Architecture (EPCglobal, Inc.)

Figure 3 represents the EPC Network Architecture refined by EPCglobal [5]. Comparing with the architecture originated at the MIT Auto-ID Center in Figure 1, the new architecture is focused on defining the interfaces between each layer regardless of the definition and implementation of internal processing structure for each component. In other words, EPCglobal governs only the standards for the interfaces, and their implementation is out of the scope.

As depicted in the diagram, EPCIS, which plays the same role of PML Server, sits at the highest level of the EPC Network Architecture and has the capture and query interfaces through which data is exchanged between EPCIS Capturing Applications, EPCIS Accessing Applications and EPCIS Repositories. That is, it allows external resources to have access to the tag reads data and associated business description through the standardized way. EPCIS Capturing Interface includes the following operations: tag commission/decommission, association among tagged items, tag observation and so on. EPCIS Repository is responsible for storing data coming through EPCIS Capturing Interface and making the data available for latter query via EPCIS Query Interface.

RFID middleware software resides in the layer below EPCIS. The middleware, formally known as 'savant',

processes the reduction of the data volume directly coming from various sources such as RFID readers and routes the events of interest to applications. The term 'savant' was deprecated by EPCglobal and is replaced to 'Application Level Events (ALE)'. The role of ALE interface is to maintain the independence between device infrastructure and the applications that are the consumers of the tag data. The interface defines the control and delivery of the filtered and collected tag read data and EPCIS Capturing Application has access to the middleware via the ALE interface in order to obtain the tag read data. EPCIS Capturing Application defines the metadata called 'ECSpec' that includes what readers are associated with, how the boundaries for tag reads collection are defined and how the tag list is reported. RFID middleware collects and reports the tag reads to applications by following the ECSpec. The development of the ALE specification is currently under way and it is expected to be ratified in the first half of the year, 2005.

4. The RFID Middleware Framework and Implementation

So far, we discuss the software requirements for RFID middleware, the evolution of EPC Network designed by EPCglobal as a reference, and take a look at the details of RFID middleware. In this section, we propose the RFID middleware framework and introduce the implementation of the middleware software based on the framework. The proposed RFID middleware is designed by adopting the EPC network architecture – especially, the earlier version.

The RFID middleware framework that we propose as in Figure 4 shows the high-level components to integrate between heterogeneous RFID devices and the back-end systems in a seamless manner.

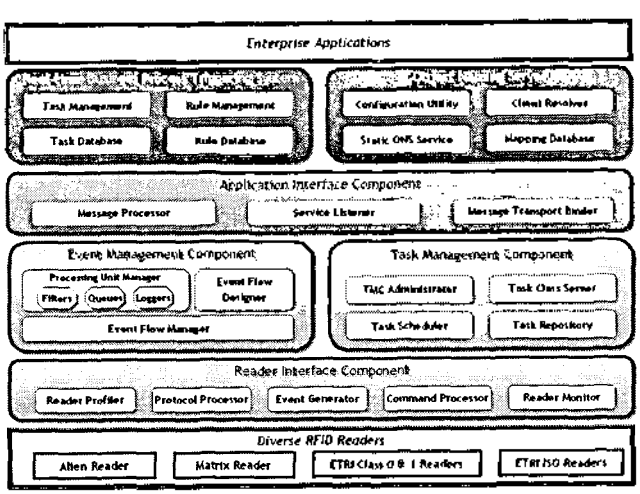

Figure 4. Proposed RFID Middleware Framework

Basically, the proposed system is layered into several levels spanning from the device layer up to the legacy application layer and, at each layer, a number of components need to be defined. These components are discussed in the following.

4.1. Reader Interface Component

The RFID middleware software should support various types of reader devices and allow users to configure, monitor and control all these devices. In order to meet those requirements, Reader Interface Component (RIC) consists of the modules as following.

Reader Profiler manages and maintains the data about the devices which are deployed to the middleware system. The data include the protocol which a device supports and the extra information which is needed to operate the device.

Protocol processor helps ensure the middleware has access to the RFID readers via various network communication options and support low-level integration with the hardware devices. Currently, readers are built on various communication interfaces - such as RS232, TCP/IP etc - and their own data exchange protocols to communicate with applications, so protocol processor enables readers from many different manufactures to interact with the middleware application with seamless way. Our middleware software currently supports only TCP/IP-based communication with readers, but we plan to extend the communication based on RS232 and USB.

Generally, tag reads from readers are sent to applications every time read cycle per each reader completes. In this case, as long as a tag stays at the read range, its tag read is continuously notified to applications. On the other hand, we introduce the event generation steps that serves to reduce the volume of incoming tag reads data. Event Generator defines three states per each tag and four different event types which are generated when the transition between two states happens. That is, the tag read is delivered only if something interesting happens – here, the transition between two different states occurs. Whenever a tag is seen for the first time, the event 'TagSeen' is generated. 'FirmRead' event is introduced when the tag appears present for a number of read cycles within limited period. If the tag previously generated a 'FirmRead' event, but has not seen for a pre-defined time interval, then the tag generates a 'TagExpired' event; Moreover, if the state of the tag remains at 'SoftRead' state and no more the tag is seen for a while, the event 'TagVanished' is issued. This graceful way gets rid of the redundant delivery process of the same tag data and therefore leads to significant volume reduction of incoming stream [6].

Command Processor converts the commands issued by users to reader-aware commands and then passes them to Protocol Processor. Reader Monitor is used to monitor the healthy of the deployed RFID readers and manage the readers through Graphical User Interface. Figure.6 shows the main window of RIC and users can check the location where the readers are installed and their aliveness through the status of icons. Moreover, users can issue commands to

a specific reader through GUI.

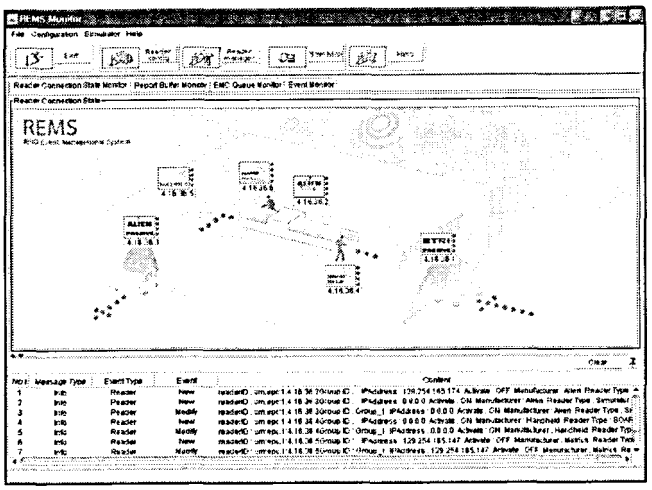

Figure 5. Reader Monitor GUI Window

4.2 Event Management Component

Event Management Component (EMC) performs filtering, aggregation, and routing the RFID event data coming from RIC, and then notifies the associated external applications of the data. The number of RFID event data flowed from RIC ranges from 10s of event per second up to more than 100s a second, so it is important to apply appropriate filtering processing on them. Filtering of that data can eliminate such information that is either redundant or that the client applications are not interested in. The filtering requirements mainly depend on the application domain.

Some of filtering examples that might be required of almost all applications are following. Basically, a reader reports redundant tag reads to applications as long as a tag stays in the region. A basic filter is to eliminate the redundant tag read events. In some cases, a tag cannot be seen for every read cycle of a reader because the RFID reader cannot report the tag reads with 100% accuracy. Besides, unwilling tag read is sometimes reported in case that an unexpected tagged object bypasses near the region. Coping with all the situations, so-called smoothing filter is applied. Another example is the ID-based filtering. Every tag has its own unique ID and, if ID matches the predefined bit pattern, then the filter passes it to applications while ID that doesn't match the pattern is thrown away. Also, filtering based on reader identity is one of significant filtering techniques. Lastly, multiple readers can report the same tag read if they are placed close to each other and they are related to the same semantic operation like readers to capture the entering of products in a warehouse. For that case, coordination filter is used to select a tag reads among multiple tag reads from readers. Actually, the basis of filtering tag read is whether an event is interested in client applications or not. All the filters discussed so far are supported by our middleware software as built-on filters.

By the way, the filtered data can be posted to other filters for further processing, or loggers for sending event data to external applications. When the filtered event data is conveyed to client applications, the way of data exchange and communication might be followed to the manner that applications are required. However, in terms of the middleware, it should also provide the common communication interfaces so that the interface leads to loosely couple the middleware from applications. We currently support several well-known ways to deliver the events. The methods include the notification via the standard network protocols such as HTTP/POST or SOAP, and via recording in persistent storage such as database and file.

All the filters, queues and loggers can be developed as unit modules and then they can be registered to EMC system through the Processing Unit Manager. It allows users to register the customized units so that they satisfy the domain-specific requirements.

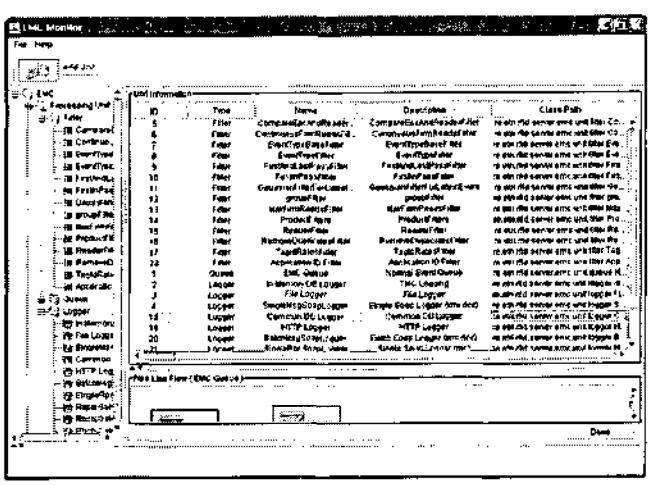

Figure 6. EMC Manager GUI Window

Event Flow Designer, as shown in Figure. 6, displays the registered processing units and allows users to model the process flow which describes how the event data can be filtered, buffered and logged. All the processing units may be connected in directed acyclic graph fashion to meet the needs for the domain. The designer component saves the flow diagram which is the result of the modeling as the XML format file. Event Flow Manager instantiates and executes the process modules by interpreting the XML configuration file.

4.3. Task Management Component

Task Management Component (TMC) allows users to define the custom tasks in order to manipulate the tag read data collected from readers and then executes the tasks periodically. User defines a custom task as a binary module like java class file and then registers it via TMC Administrator with the additional information on the schedule for execution. Based on the information, TMC

scheduler executes the task and maintains the execution history of the task.

4.4 Application Interface Component

Application Interface Component (AIC) offers interface so that application system could control readers. Service Listener received the requests from the application systems. It offers the communication function such as XML-RPC, SOAP-RPC, Web-service, etc. Message Processor analyzes the delivered order and passes it to the Command Processor of the Reader Interface. Then it receives a response from the Command Processor and returns it to the application system which called service.

4.5. Real-time Business Process Triggering System

Real-time Business Process Trigger System (RBPTS) is a kind of a rule engine that validates user-defined rules based on the tag read events and then invokes the services whenever the rules meet certain conditions. Rules are applied on the information when EMC feeds the tag read data via the custom loggers for RBPTS. The activity that this component is integrated with the business process management solutions is ongoing for the aim of business process automation triggered by the auto identification.

4.6. Object Naming Server

Object Naming Server is a directory service that matches the unique code assigned to each tagged product to one or more network location address (URL) of the servers which has extensive information about the product.

5. Conclusions

RFID technology is known to be well-suited to linking the physical and virtual world and is considered as a key technology to lead us to the ubiquitous computing world. Most of all, standardization is considered as a key factor in encouraging widespread adoption of RFID technology and, currently, several international organizations for standardization such as EPCglobal Inc., work on standardization in the field of RFID technology. Many other organizations also make an effort to derive user and system requirements, show diverse reference models incorporated with RFID technology, and validate their applicability.

In this paper, we draw the software requirements for RFID middleware and examine the technical trend of the RFID software system. Then, we propose the RFID middleware framework with the implementation issues. The major role of RFID Middleware is to collect huge amount of real-time tag read events coming from multiple readers, perform filtering operation on the data stream in order to reduce the data volume, and then provide the information to other existing client applications. It is important for the middleware to provide the uniform interface to client applications regardless of heterogeneity among readers from multiple vendors, having their own way of operation and communication. Moreover, the issue how the more meaningful information or knowledge can be produced through gluing the events captured by RFID readers with legacy systems might be resolved by adopting rule-based engine such as RBPTS.

The prototype system of the introduced middleware software suite is ready to be released soon. We are now working on the field test in order to apply to many business domains such as the warehouse management system and we also need to validate the system in terms of performance and flexibility. We hope to contribute our works to the growth of the RFID industry.

REFERENCES

[1] RFID Journal, "Middleware is the Key to RFID", http://www.rfidjournal.com/article/articleview/858/1/82

[2] Auto-ID Center, "Auto-ID Savant Specification 1.0", http://www.epcglobalinc.org/standards_technology/6_auto_id_savant-1_0.pdf

[3] Auto-ID Center, "The Savant Technical Manual, Version 0.1 (alpha)", http://www.autoidlabs.org/whitepapers/MIT-AUTOID-TM-003.pdf

[4] EPCglobal, Inc., http://www.epcglobalinc.org

[5] EPCglobal, "EPC Information Services (EPCIS) Version 1.0 Specification", Working Draft, Oct. 2004

[6] Auto-ID Center, "Auto-ID Reader Protocol 1.0", Sept. 2003