Design and Implementation of Sensor Tag Middleware for Monitoring Containers in Logistics Systems

Md. Kafil Uddin, Gihong Kim, Bonghee Hong
Department of Computer Engineering, Pusan National University, Republic of Korea
e-mail:{mdkafil,buglist,bhhong}@pusan.ac.kr

Abstract

Sensor tag having user memory can be used both for identification and storing sensor data. Monitoring sensor data in container is a big issue in logistics system. Containers in logistics system generate huge amount of sensing information which requires handling efficiently to provide response to the user queries. Moreover, collection of sensor data from sensor tag by using heterogeneous reader devices is also a big challenge in sensor tag deployment. To achieve these goals, a flexible middleware is needed that provides glue between applications and the heterogeneity of devices by facilitating optimized set of services at high performance to a scalable number of users.

In this paper, we propose Sensor Tag Middleware architecture that incorporates characteristics of standard EPCglobal middleware with several unique features such as reader abstraction, integration, high performance and scalability. In order to illustrate our proposed middleware we implement it for monitoring containers in port logistics.

1. Introduction

Recent advances in sensor technology, wireless Radio Frequency (RF) communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional RF capable sensor tags that are small in size and can transfer data to the reader when it is in reader interrogation area. Such tags are, in general, characterized by RF identification, user memory and sensing information. A large number of such tags can be deployed in container management in logistics systems to provide container information such as temperature, humidity, light and pressure as well as container security like door open or lock information. Thus, it is possible to create a physically linked world in which every container is numbered, identified, cataloged and tracked with various sensing information. However, these types of containers are widely used in Logistics Systems such as port logistics.

Sensor Technology has recently begun to find greater use in container management in port logistics, cold chain management [9] and supply chain management. In these domains sensor technology holds the promise to eliminate many existing business problems by bridging the gap between the virtual world of IT systems and the real world of products and logistical units. Common benefits include safe container transportation, more efficient material handling processes, quality control of temperature-sensitive perishables with continuous stationary monitoring, elimination of container damage due to huge inside pressure, and automatic tracking of product location in supply chain. However, the growing diversity of application and available hardware and software platform demands developers and administrators to come up with specialized software and deployment strategies for each of the platforms [7]. This is a time consuming and tedious task which in other, admittedly more “stable” domains has been addressed by abstracting from the physical view of a system to a logical model and implementing this logical model in terms of a middleware [10]. The general purpose of middleware systems is to provide powerful abstractions codifying the essential requirements and concepts of a domain and providing flexible means for extending it to the concrete physical environment [6]. This speeds up deployment and additionally pushes standardized APIs which simplifies application development and applications become “portable” over all physical systems included in the middleware. Moreover, well designed middleware architecture enables application modularity, adaptivity and repairability [8] by providing flexibility in architecture with the change in requirements.

We develop state-of-the-art sensor tag **Middleware** for **Monitoring Containers** (**MMC**), an integrated sensor tag middleware with RFID (Radio Frequency Identification) middleware compatible with EPCglobal architecture [4]. It supports reader abstraction for heterogeneous readers and increases performance and scalability by providing continuous query processing for sensing information.

In this work, we have analyzed those requirements the sensor tag middleware components should meet in order to manage containers with large deployments of tags having sensing capabilities, diverse heterogeneity in reader devices, and the amount of data those readers capture. The main contribution of this paper is to design software components for sensor tag middleware, which addresses both application needs and the constraints of sensor technology.

This paper is organized as on the basis of following sections. In Section 2, we list related works. Section 3 we illustrate the design and implementation of MMC with some unique features. We conclude Section 4 with avenues for further research.

2. Related works

2.1. EPCglobal Architecture

The EPCglobal Inc. [5] provides standard architecture for middleware framework. The architecture framework consists of three layers: (1) Identification layer, (2) Capture layer and (3) Exchange layer. EPCglobal provides several standards at all the three layers such as Data Standards, Interface Standards and standards that are under development. Therefore, importance given on EPCglobal standards in designing middleware architecture that helps easy adaptation of the system over the world.

The EPCglobal Network^{TM} [4] is a set of global technical standards aimed at enabling automatic and instant identification of items in the supply chain and sharing information throughout the supply chain. It defines standard specification for RFID middleware, typically known as Application Level Event (ALE). Reader collects tag data and sends to the middleware (ALE) by using Reader Protocol (RP) [3]. ALE performs collection, filtering and grouping on tag events send by the Reader. Following figure1 shows ALE framework defined by EPCglobal.

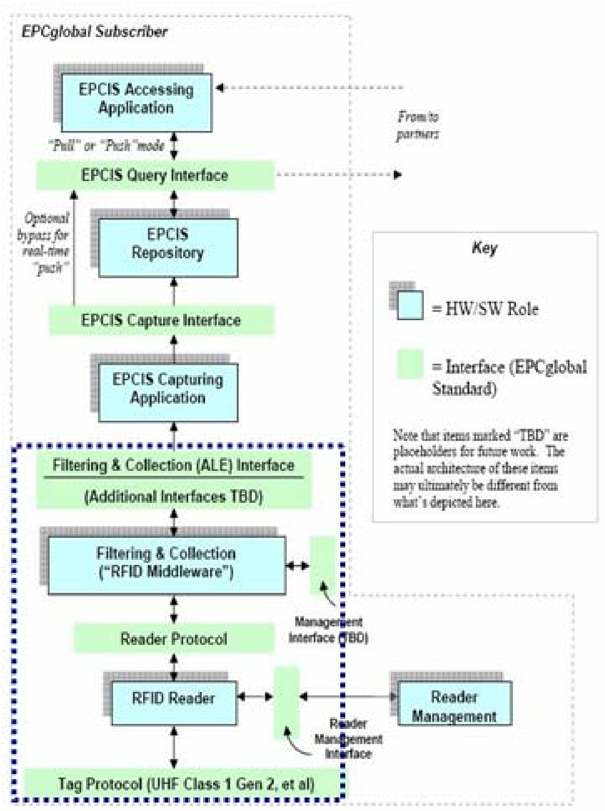

EPCglobal Subscriber
EPCIS Accessing Application
"Pull" or "Push" mode
From/to partners
EPCIS Query Interface
Optional bypass for real-time "push"
EPCIS Repository
EPCIS Capture Interface
EPCIS Capturing Application
Filtering & Collection (ALE) Interface
(Additional Interfaces TBD)
Filtering & Collection ("RFID Middleware")
Management Interface (TBD)
Reader Protocol
RFID Reader
Reader Management
Reader Management Interface
Tag Protocol (UHF Class 1 Gen 2, et al)
Key
= HW/SW Role
= Interface (EPCglobal Standard)
Note that items marked "TBD" are placeholders for future work. The actual architecture of these items may ultimately be different from what's depicted here.

Figure1. EPCglobal Middleware Framework

2.2. Application Level Event (ALE)

The EPCglobal specifies standard specification for ALE 1.1[1]. It consists of four specs: (1) Reading API (ECSpec), (2) Writing API (CCSpec), (3) Tag Memory (TMSpec) API, (4) Logical Reader (LRSpec) API and (5) Access Control (ACSpec) API. The key characteristic of ALE Reading API is duplicate data removal. Tags may be read several times within an event cycle where event cycle is the smallest unit of time defined by the ALE client. Reading API reduces middleware load by filtering data duplication. Writing API provides API for writing to the tags by ALE Clients. Clients define command cycle, the smallest unit of time, to write to tags. Various commands such as kill, lock, pwd, etc. are defined by the Writing API. Thus, Writing API facilitates ALE clients to write "*user data*" to the tags having "*user memory*". However, for accessing tag memory, ALE provides an interface, called TMSpec, through which clients can define symbolic names that refer to data fields to tags. The key role of TMSpec is mapping tag memory address to its logical name so that clients can read and write to the memory bank without using it physical address value. LRSpec is an interface through which clients may define logical reader names instead of using physical reader names and ACSpec facilitates with the API by which access rights can be configured for various clients.

2.3. Gen2 Tag Memory Architecture

The EPCglobal specifies standards that provide full access to the functionality to the EPCglobal UHF Class 1 Gen 2 specification [2], when interacting with Gen2 tags. This includes reading and writing all memory banks, enabling users to write user memory bank which can further be used to store sensing information from any sensor device. Figure 2 shows Gen2 tag memory architecture. It consists of four memory banks: Reserved Bank (00), EPC Bank (01), TID Bank (10) and User Bank (11). The Reserved Bank is used for future use, EPC Bank stores Electronic product code, TID Bank stores Tag identification and the User Bank is used to store user related information. This bank can be used for any user purposes. Here in our MMC we used this user memory bank for storing sensing information. The memory architecture of Gen2 tag user memory consists of several individual memory blocks.

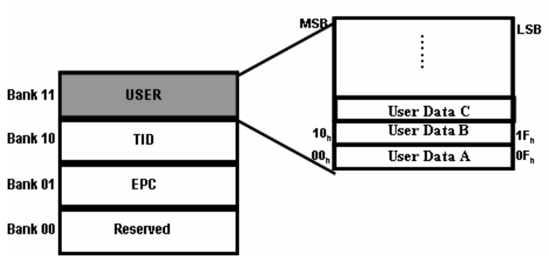


Figure 2. Gen2 Tag Memory Architecture

Each memory block in the user memory can be used to store different user related information which can further be used to store sensing information. Thus, Gen2 tag with sensor device can act as sensor tag and capable of serving dual act of identification and sensing information.

3. Design and Implementation of MMC

The architectural design and implementation of our MMC Sensor Tag Middleware comprised of several distinctive features that made it solely performed among all other existing sensor tag middlewares. In this section, we have summarized the key features of our MMC, its layered design and finally we explain about its implementation.

3.1. Features of MMC

While combining multiple application protocols into a large, adaptive and self-updating protocol is possible, our Sensor Tag Middleware has several unique features over other sensor middlewares.

3.1.1. Integration. Figure 3 depicts the integrated MMC which is an extension of EPCglobal standard Middleware. EPCglobal standard middleware collects data from passive tags and our extended integrated Middleware collects streaming sensing information both from passive and active tags. As active tag has user memory, we use its user memory for collecting sensing data. The advantage of using active tag as sensor tag is that it responds only when middleware requests for sensing information. Thus reduces both communication overhead and significant reduction in battery power. Our MMC receives sensing related user queries and performs queries on user memories and sends response to the users.

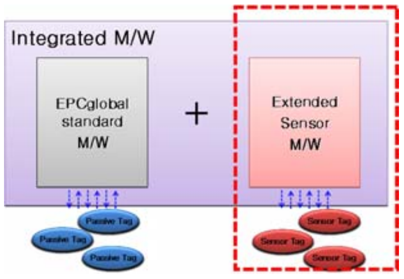


Figure 3. Integrated MMC

3.1.2. Abstraction. It is one of the key features of our middleware. We provide two types of abstractions in our middleware: Independence of reader heterogeneity and Logical naming for physical tag address. Firstly, reader heterogeneity is a big issue in designing well structured middleware. We find there are many vendors exist who provide readers to collect tag data from various tags. They are of two types: one is Reader Protocol (RP) complaint readers and other is non-RP complaint readers. RP complaint readers follow standard RP protocol whereas non-RP complaint readers do not follow RP protocol. Since non-RP complaint readers follow their own vendor specifications, middleware need to provide individual API and Adapter for each vendor specific readers. Figure 4 shows our Reader Abstraction Framework for MMC that provides an independent framework from types of readers.

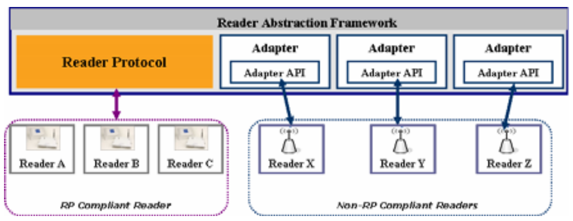


Figure 4. Reader abstraction framework for MMC

Secondly, while combining multiple reader protocols into a large, adaptive and self-updating protocol, our middleware uses abstraction for accessing physical memory block of tag memory by using logical

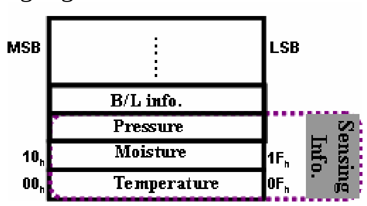


Figure 5. Storing sensor data in Gen2 User Memory

address name. Figure 5 depicts the Gen2 tag user memory architecture where user memory of Gen2 tag is used for storing various sensing information such as temperature, moisture, pressure, light etc. Recall that the key feature of TMSpec is mapping from logical name to physical address. Therefore, it is possible to provide logical name for each memory block holding sensing information. Our MMC uses key feature of TMSpec that provides logical name for accessing each physical address of the tag memory storing sensor data. Hence, hides the details of underlying memory architecture of sensor tag.

3.1.3. Extensibility. There are two ways to access sensing information from the middleware: Direct function call and Indirect function call. In case of direct function call, each sensing data is accessed by using individual method provided by the API. Therefore, middleware requires extending or modifying methods if any new sensing storage is added to the tag memory or if there is modification done on physical storage. Figure 6 shows the detailed methods for accessing sensing information in tag memory. On the other hand, indirect function call uses memory mapping technique to address to the logical name of memory address following the mapping rules specified in TMSpec. Therefore, accessing any memory address for sensing information requires only one method that uses memory address as its parameter. Thus reduces the

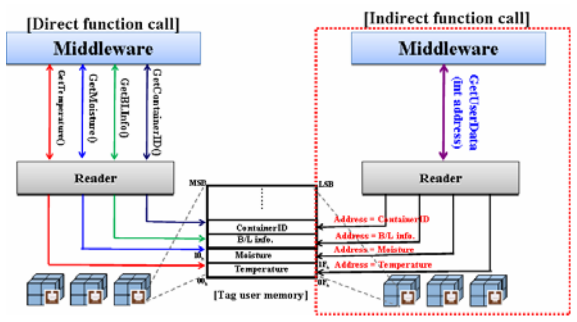


Figure 6. Direct and Indirect function call from Middleware

methods and API to access to reader from middleware. Figure7 shows method elimination technique in

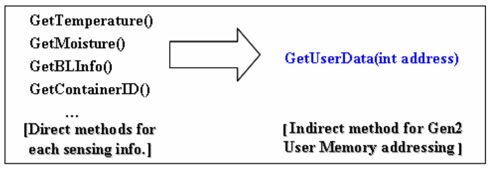

GetTemperature()
GetMoisture()
GetBLInfo()
GetContainerID()
...
[Direct methods for each sensing info.]
GetUserData(int address)
[Indirect method for Gen2 User Memory addressing]

Figure 7. Methods elimination by using Indirect function call

indirect function call. Our MMC provides extensible middleware architecture by implementing indirect function call instead of using direct function call for accessing sensing information. Therefore, no need to change system architecture even though physical storage address of tag memory is changed, extended or modified.

3.1.4. High Performance and Scalability. The question of high performance has always to be seen to the questions of response time. Incase of only few requests or queries to be processed, too much care about performance is not necessarily required. However, sensor middlewares have many applications, users and queries. Having thousands of queries to be processed (in short period of time) requires a well designed system, being able to bear the burden. We used Continuous Query Index (CQ Index), shown in Figure 8, to process enormous queries efficiently. Continuous Query Manager (CQ Manager) receives queries from user applications. After receiving queries it builds *query index* (representing query using spatial data structure that improves the speed of operation) on stored queries. The reader interface receives streaming sensor data from the sensor tag. Sensor data is send to the query index to get the query result. CQ index sends it to the CQ manager. Finally, CQ manager delivers that result in response to the query issued by the user.

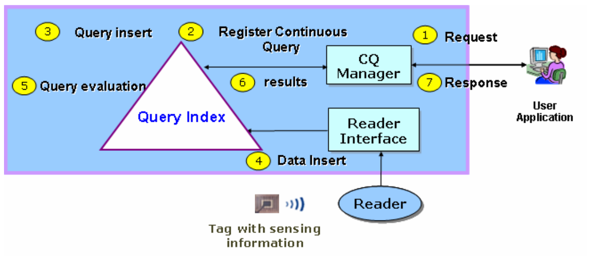


Figure 8. Continuous Query Processing in MMC

Thus, a quick and fast response is possible to generate for a significant number of user queries without degrading system performance.

3.2. Architecture of MMC

Figure9 shows the detailed MMC architecture. It consists of three layers: Application Layer (Web Service), Core Engine Layer (MMC Engine- Sensor Middleware) and Reader Abstraction Layer (Adapter Framework).

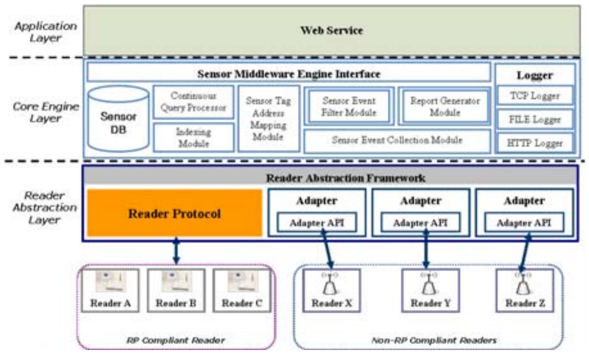


Figure 9. 3-Layer MMC Architecture

3.2.1. Application Layer. This is the upper layer of the Middleware. It contains web services and provides clients access to sensing information via Web Service API. Our system provides independence from application using *Common Application Interface* and *Common Application Query*. We used Common application interfaces provided by EPCglobal, which are Tag Memory Specification API, ALE Reading API, ALE Writing API, ALE Logical Reader API and Access Control API. Web applications use standard protocols (HTTP, TCP and File) to send queries to the MMC sensor Tag middleware. Applications can also send their queries to the middleware by using Remote Method Invocation (RMI). MMC Engine uses running thread *Listener* to receive user queries.

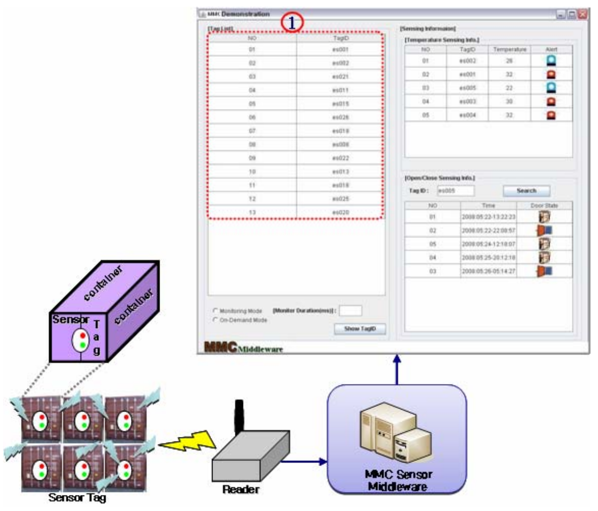
3.2.2. Core Engine Layer. This layer consists of Core MMC Engine. It contains several Modules, Sensor Database and Loggers. SensorDB stores all the sensing information stored by the sensors for certain period of time. Indexing Module uses CQ Index for fast response to the user queries. The Query Processor Module parses and processes the query received from applications. Query contains logical name of tag memory address requesting sensing data. The Sensor Tag Address Mapping Module maps the user mentioned logical name for sensing information to the physical address value. Continuous sensor events are collected from reader by the Event Collection Module and Event Filter Module filters the streaming sensor events according to the filtering condition specified by users. Finally, reports are generated by the Report Generator Module and middleware sends the generated report to the corresponding user in response to the query issued by that user. Moreover, our MMC provides three types of Loggers (TCP Logger, HTTP Logger, and File Logger) to log the users accessed asynchronously to the middleware.

3.2.3. Reader Abstraction Layer. The lowest level of MMC is *Reader Abstraction Framework*. This layer is used for providing reader independence to the middleware. It consists of two types of modules: Reader protocol and Adapters. The Reader protocol module is used to communicate and transfer data handling between RP complaint readers and the middleware whereas for each non-RP complaint readers individual adapter is used for communication between reader and middleware. Thus *Reader Abstraction Framework* provides seamless link between the Sensor Tag Middleware and various types of readers collecting sensor events. This layer allows users to configure, monitor and control reader devices without knowing details about reader types. It supports XML-based reader control methodology to control reader specific features.

3.3. Implementation of MMC

We have implemented our MMC sensor middleware for monitoring containers in the port logistics. Sensor tags are attached to each container. As each tag has its unique id, it is possible to identify and track the container along with sensing information. Our middleware is possible to provide asynchronous alarm notification to the client whenever sensing information exceeds certain threshold limit set by the user. Thus it helps safe monitoring of containers and it's inside status. Moreover, MMC can provide door status of containers by using illumination sensor. To illustrate our implementation of MMC we consider four implementation scenarios: (1) Collection of Tag Lists (2) Collection of Sensing Information (3) Door Status (open/close) and (4) Alarm Notification. All the four implementation scenarios are described in the following sections.

3.3.1. Tag List Collection. We attach sensor tag to each container. According to the clients requests reader reads the containerId by reading tagId of each container and sends to the middleware. Middleware collects all the tag lists of monitored containers. Figure 10 shows the list of tags collected by MMC.



NO	TagID
01	es001
02	es002
03	es021
04	es011
05	es015
06	es026
07	es018
08	es006
09	es022
10	es013
11	es018
12	es025
13	es020

NO	TagID	Temperature	Alert
01	es002	26	🟦
02	es001	32	🟥
03	es005	22	🟦
04	es003	30	🟥
05	es004	32	🟥

NO	Time	Door State
01	2008.05.22-13:22:23	
02	2008.05.22-22:08:57	
05	2008.05.24-12:18:07	
04	2008.05.25-20:12:18	
03	2008.05.26-05:14:27	

Figure 10. Collection of Tag lists by MMC

3.3.2. Sensing Information Collection. Figure 11 shows the collection of sensing information by our MMC. We deploy several sensor devices with each container. For example, heat lamp is attached to the container that generates heat and light. The temperature sensor collects the sensing information and sends to the sensor tag attached to the container. The Thermo Threshold meter monitors whether the temperature goes above the threshold limit. Thus, collected temperature values are stored in the tag memory. To respond to the user's query, reader read the intended memory address of the tag to get the sensing information.

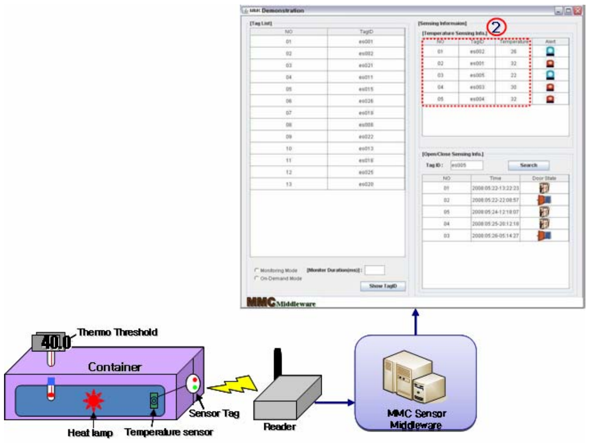


Figure 11. Collection of sensing information by MMC

3.3.3. Door Status (Open/Close). Monitoring door status is one of the most important requirements for monitoring container security in logistics systems. Light or temperature sensitive items or perishable goods inside the container may be damaged due to uncontrolled light and temperature. To fulfill these requirements we use light sensitive sensor inside the container that monitors light intensity in the container and decides whether the door is open or closed and responds to the middleware accordingly. Figure 12 shows the door status of each container in our logistics implementation.

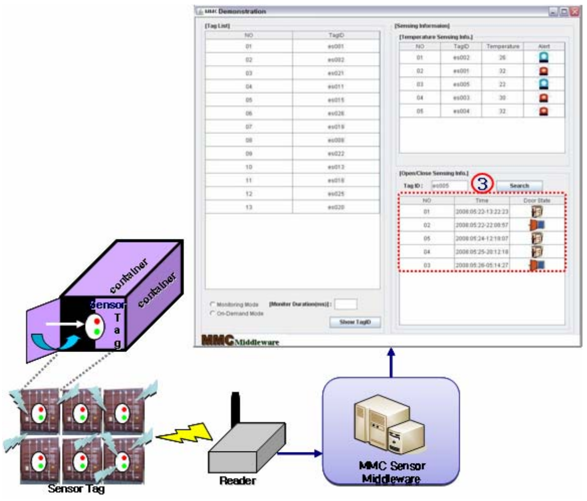
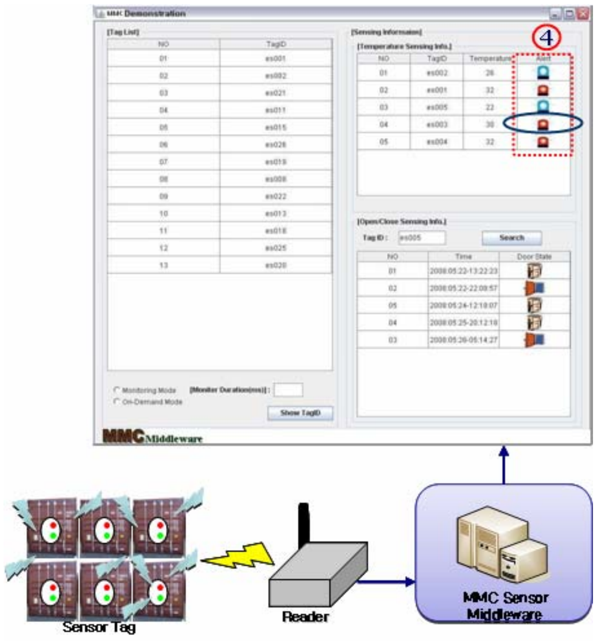


Figure 12. Containers Door status by MMC

3.3.4. Alarm Notification. Application users require sending alarm if certain unwanted situation occurs. Our MMC provides interface to clients for setting certain threshold values for sensing data. It also allows users to set some alarm conditions. When sensing information exceeds threshold limit or fulfills alarm conditions middleware sends asynchronous alarm notification to the corresponding client. Thus provides safe system monitoring. Figure 13 depicts the alarm notification generated automatically when temperature exceeds the limit 30 degree.


MMC Demonstration

[Tag List]

NO	TagID
01	es001
02	es002
03	es021
04	es011
05	es015
06	es028
07	es019
08	es008
09	es022
10	es013
11	es018
12	es025
13	es020

Monitoring Mode
On-Demand Mode
[Monitor Duration(ms)] :

[Sensing Information]

[Temperature Sensing Info.]

NO	TagID	Temperature	Alert
01	es002	26	🟦
02	es001	32	🟥
03	es005	22	🟦
04	es003	30	🟥
05	es004	32	🟥

[Open/Close Sensing Info.]

Tag ID : es005

NO	Time	Door State
01	2008.05.22-13:22:23	▯
02	2008.05.22-22:08:57	▮
05	2008.05.24-12:18:07	▯
04	2008.05.25-20:12:18	▯
03	2008.05.26-05:14:27	▮

Figure 13. Alarm notification by MMC

To illustrate our system we have implemented MMC in Intelligent Research and Develop Center (ICC) and Logistics Information Technology (LIT), Korea. We developed MMC GUI by using JAVA (JDK 6.0), We have also used several software tools such as Sun Server, Tomcat Web-server (version 6.0) and Smart UML (version 5.0.2).

4. Conclusion

In this paper, we have discussed about the detailed architecture and features of Sensor Tag Middleware. Tag memory in the sensor tag can be used for storing various sensing information. Different types of sensing information can be stored in the user memory of same sensor tag. Sensor tag can be deployed in containers in logistics system. We implemented Sensor Tag Middleware for efficient and continuous monitoring of containers in the logistics system. Our featured middleware can solve the problem of heterogeneity of various reader types such as RP complaint readers and non-RP complaint readers by using reader abstraction layer in the middleware. Mapping of name to address in tag memory is the key feature of TMSpec mentioned in EPCglobal standard specification ALE 1.1.Our middleware provides independence of architecture by incorporating this feature to our middleware whenever tag memory is extended or new sensing information is required to access from sensor tags. Furthermore, it is an integrated middleware compatible with EPCglobal standard architecture that facilitates optimized set of services at high performance to a scalable number of users. Here, we have analyzed the requirements of sensor tag middleware, derived its key features, and then we came up with designing the architecture and finally, summarize all its functional operations. Our future research work lies on the unification of RFID middleware, RTLS middleware and Sensor middleware into one so that all three types (RFID, RTLS and Sensor) of tag data can be processed by using a single middleware.

Acknowledgements

This work was fully supported by Intelligent Research and Develop Center (ICC) and Logistics Information Technology (LIT), Korea.

5. References

[1] EPCglobal Inc., "The Application Level Events (ALE) Standard Specification Version 1.1" February 27, 2008.

[2] EPCglobal, "EPC^{TM} Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 806 MHz – 960 MHz Version 1.0.9", EPCglobal Standard , January 2006.

[3] EPCglobal Inc., "The Reader Protocol Standard, Version 1.1", Ratified Standard June 21, 2006.

[4] EPCglobal Inc., "The EPCglobal Architecture Framework, EPC global Final Version 1.2", Approved 10 September 2007.

[5] EPCglobal website. www.epcglobalinc.org

[6] A. Kabir, B. Hong, W. Ryu, S. Ahn, "LIT Middleware: Design and Implementation of RFID Middleware", In proceedings of *Dynamics In Logistics*, LDIC 2007, Bremen, Germany, August 2007.

[7] C. Floerkemeier, M. Lampe "RFID middleware design – addressing application requirements and RFID constraints" , Joint sOc-EUSAI conference, Grenoble, October 2005.

[8] T. Liu, M. Martonosi, "Impala: A Middleware System for Managing Automatic Parallel Sensor Systems" *PPoPP'03*, June 11-3, 2003, San Diego, California, USA.

[9] Cold Chain Applications http://www.sensitech.com/ applications/coldstream_ccm/index.htm

[10] K. Aberer, M. Hauswirth, A. Salehi, "A Middleware for Fast and Flexible Sensor Network Deployment", *VLDB'06*, September 12-15, 2006, Seoul , Korea.