XCREAM: Collaborative Middleware Framework for RFID/USN-Enabled Applications

Kyungeun Park, Jekuk Yun,
Changhyun Byun, Yanggon Kim
Towson University
8000 York Rd
Towson, MD, 21093 U.S.A.
{kpark3, jyun4, cbyun1, ykim}
@towson.edu

Juno Chang
Sangmyung University
7 Hongji, Jongno,
Seoul, 110-743 Rep. of Korea
jchang@smu.ac.kr

ABSTRACT
The purpose of the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware) is to enable collaboration among many RFID/USN-enabled applications by providing them with flexible interface to the XCREAM through a web-based service scheme, called the Enterprise Manager, and XML infrastructure language, the XLogic script language. The XCREAM framework gathers massive events from a variety of event sources and distributes them to the appropriate service parties depending on the predefined business scenarios. The scenarios are written in the XLogic script language and registered to the XCREAM framework. The paper includes simulation result which includes the performance and collaboration validity of the XCREAM. This approach makes it possible to integrate many heterogeneous services and to present a collaborative service framework.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Constructs and Features – *frameworks.*

I.6.8 [**Simulation and Modeling**]: Types of Simulation – *discrete events.*

General Terms
Management, Performance, and Languages

Keywords
XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware), RFID (Radio Frequency Identifcation), USN (Universal Sensor Network), Collaboration, Middleware Framework.

1. INTRODUCTION
Rapid progress in wireless telecommunication and sensing facilities based on RFID/USN technology encourages us to introduce the advanced smart computing services in order to resolve many complicated issues happening across the interested parties. As a result, individual parties need to communicate with each other and try to make organic connection or integration to others.

In order to provide complex services of interconnected applications in RFID/USN environment, a seamlessly orchestrating framework, which collects a large amount of sensor data from many data sources and delivers real-time data according to a predefined scenario to each service, is necessary. Moreover, the framework must be a robust and reliable infrastructure, which is able to handle a huge amount of tag and sensor data and propagate the data to the appropriate services according to predefined scenarios.

This demand lets us develop a scenario-based collaborative framework, the XCREAM (XLogic Collaborative RFID/USN-Enabled Adaptive Middleware) framework, which seamlessly integrates numerous and heterogeneous application services and plays an organizing role in ultimate ubiquitous computing environment [8]–[9].

Various application services such as emergency rescue system, facility management system, and supply chain management system can not only provide their own services, but also perform more advanced collaborative services by combining the individual services with the other applications within the XCREAM framework.

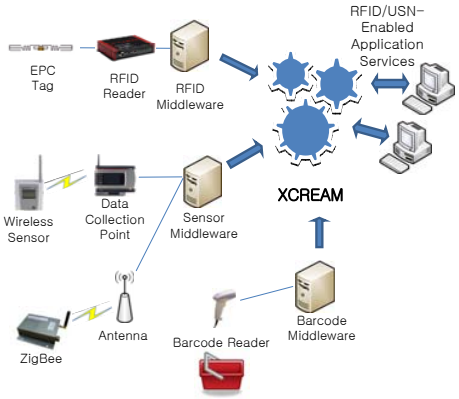

Figure 1. The XCREAM Framework Configuration.

The XCREAM is placed between RFID/USN middlewares, generally composed of RFID or sensor middlewares, and application services (see Figure 1, 2). The XCREAM framework collects the ECReports from RFID/USN middlewares [10] and

interprets and delivers them to the corresponding user-defined business application services in order to establish an integrated cooperative working environment. The highly sophisticated services, which are triggered by a specific event from a variety of sensors or other services according to a pre-defined scenario, result from the seamless integration between many service systems. The framework introduced XML infrastructure language, called the "XLogic (eXtensible Logic)", which makes the applications communicate with the XCREAM by registering XLogic scripts. The XLogic script contains specific service scenario depending on the events of automatic identification or remote measurement data from the separate RFID/USN middlewares. The XCREAM interprets the XLogic script, which contains a query request, into a Java runnable object, transforming the written script to the executable form. The runnable objects are activated by a specific event as in the event driven systems.

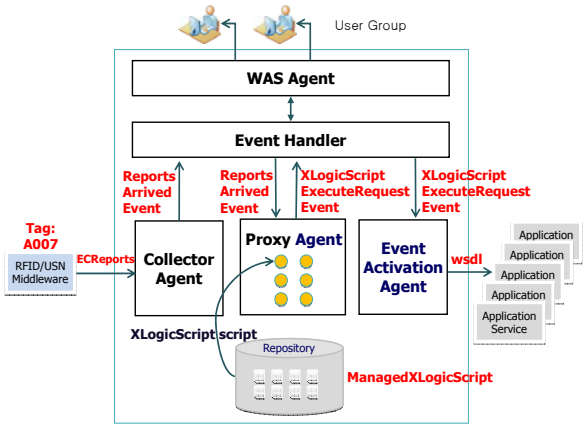

Figure 2. Overall Event Flow of the XCREAM.

The XCREAM adopts the service-oriented architecture (SOA) as an interface scheme to RFID/USN-enabled application services, not only to guarantee independence of individual services, but also to gain flexibility, when extending the framework infrastructure by enabling new collaborative services to work with the XCREAM or other application services. The SOA makes it possible to increase the reusability of the business services. By combining various services through the SOA, developers easily build composite services. The additional advantage of using the SOA consists in retaining loose coupling of the service building blocks across the whole service framework. The other important reason why the XCREAM adopts the SOA technology is that it allows a business application to request any services, which are exposed to others with SOA interfaces, regardless of the service platform environment. Moreover, the SOA has been focused as a flexible integration technology in parallel with the demands for the fast development of RFID/USN-enabled system as well as the sharp increase of the numerous device-oriented services [1]–[3].

This paper is composed of 6 chapters in total. Chapters 1 and 2 present the introduction and the related research. Chapter 3 describes the overall architecture and the internal execution mechanism of the XCREAM framework in detail. In chapter 4, the simulation results of the collaboration validity tests are presented. The conclusion and the future research direction are mentioned in chapter 5.

2. RELATED RESEARCH

There have been many research efforts in the event recognition and its processing fields: RFID-enabled auto-identification technology for the data collection and recognition of the front-end component of the framework; and USN middleware technology [5]–[7].

In conjunction with handling RFID tag data, a lot of research efforts have been made during the past decade to define the RFID tag specification. Research has also been made to formalize the protocols required for seamless communication between related components and to enhance the performance and correctness of the system itself.

Initially, RFID-enabled network infrastructure was proposed by the Auto-ID Center. Their research goal is to realize ubiquitous automated identification technologies based on the networked physical world [2]. They have developed an open architecture system which is composed of the EPC, EPC tags and readers, local networking technology, and RFID middleware [10]. Local networking technology allows readers and sensors to be connected via local databases. RFID middleware collects and filters massive tag data, aggregates and processes them into meaningful information, and delivers them to the pertinent applications. The middleware may use the Object Name Service (ONS) similar to the Domain Name Service (DNS) in the Internet for location lookups of specific items [1].

The requirements of the early USN middleware were comparatively simple due to the direct connection to a specific application service. As various kinds of ubiquitous computing applications, such as u-Healthcare, u-Transportation, u-911, and u-Eco, are rapidly introduced to our day to day life, it is expected that the increase of the information sharing from variable RFID/USN data sources will stimulate the development of the composite services [4]–[5]. Several middleware researches are having conducted on the following examples: MiLAN middleware [3], developed to guarantee QoS (Quality of Service) as its top priority; event-based DSWare middleware [6], which sends desired data to USN application systems by setting events on continuous stream of sensor data obtained, similar to RFID middleware; Impala middleware [7], which can dynamically change functions of sensor node middleware according to changes in USN application services and changes in the surrounding environment of sensor networks through wireless communication.

3. THE XCREAM FRAMEWORK

Within the XCREAM, the Event Handler plays a bridge role between the various application services and the individual RFID/USN middlewares. The Event Handler maintains the runnable objects of the XLogic scripts, which invoke the corresponding web services of the application services. Upper-level application services are to collaborate with each other, through the XCREAM, which allows pre-registered scripts to be executed depending on the event data delivered by the RFID/USN middlewares. The XLogic scripts describe specific service scenarios and interact with the XCREAM framework. Service scenarios are registered to the XCREAM through the XCREAM Enterprise Manager.

3.1 The XCREAM Components

3.1.1 The Event Handler

The Event Handler is in charge of managing the remaining four agents and delivering every event received from each agent to the appropriate parties. Internal events just pass through the Event Handler. The Event Handler propagates events to all the agents and then the agents accept only events of interest for further processing and ignore the remaining events.

3.1.2 The Collector Agent

Events originated from RFID/USN middlewares are collected by asking the XCREAM for the following queries: a *snapshot query* with which a RFID/USN middleware requests immediate real-time identification or sensor data and a *continuous query* that is kept in an active state and used to continuously request the data during a specified period.

The Collector Agent is connected to various RFID/USN middlewares and is in charge of forwarding the identification or sensor data to the Event Handler, after converting them to the corresponding Java objects, called "ReportsArrivedEven" (see Figure 2).

3.1.3 The Proxy Agent

The Proxy Agent is responsible for shortening the overall-event processing time. An XLogic script, which has been executed, is in memory as an executable Java object rather than the XML script itself saved in the repository. This buffering scheme remarkably reduces parsing time compared to the fetch-and-execution scheme from the repository and makes it possible to effectively manage the system resources throughout the whole lifetime of the XLogic script related with a specific service scenario.

The Proxy Agent usually extracts the unique identification information of "ReportsArrivedEvent," which is forwarded by the Event Handler. If the value equals to one of the XLogic script resident in memory, then the XLogic is converted into "XLogicScriptExecuteRequestEvent" and sent back to the Event Handler. If it is not found in memory, then the XLogic in XML form is queried and then parsed to be placed in memory as an executable Java object. Accordingly, the event is delivered to the Event Handler.

3.1.4 The Event Activation Agent

The Event Activation Agent not only executes the XLogic script but also keeps the statistics of the active XLogic such as the success and failure ratio, the last completion time, and the current status. This information is presented online in real-time through a web interface, the Enterprise Manager (hereafter known as the EM), and helps the user establish an execution strategy based on the acquired statistical data.

3.1.5 The Web Application Service (WAS) Agent

The WAS Agent exposes the useful functions to the outside by providing users with web interfaces, and acts as the web server of the EM.

3.2 XLogic Script Language

The external application systems register XLogic scripts of their own service scenarios to the repository of the XCREAM through the EM of the XCREAM framework, which correspond to the RFID identification or the USN sensor data.

The XLogic script language follows the XML-based scheme and provides the application services with the statements in the form of the XML tags including *set*, *wait*, *print*, *if*, *while*, *foreach*, *invokeWebService, break*, and *continue*.

4. COLLABORATION VALIDITY TESTS

As the main idea of developing the XCREAM is to establish a collaborating infrastructure for the numerous application services, the tests verifying collaboration validity have been performed. The tests include three cases with or without the XCREAM by varying the number of event generators and application services of the simulation environment: 1 event generator versus N applications; N event generators versus 1 application; and M event generators versus N applications.

The following sections show direct interface versus the XCREAM test results with the elapsed time to send the tag event from the event generator(s) to the application service(s) and receive the acknowledgements from the application(s) to the event generator(s).

4.1 1 Event Generator vs. N Applications

The first model requires one event generator, which generates 1,000s events, and increasing number of application services. The test result indicates that the test with direct interface is faster than the one through the XCREAM, because direct interface requires less transaction for the event transmission than through the XCREAM model. Nevertheless, we are encouraged with the test result, which shows the XCREAM provides gradually increasing performance as well as the reliable communication interface to the individual applications. As the XCREAM allows application developers to easily connect their own applications to the event sources with the XLogic scripts through the Web-based EM, this scheme is well worth considering as the infrastructure framework for our target environment.

Table 1. 1 EG vs. N Apps.

Test Type	1 EG/ 1 App	1 EG/ 2 Apps	1 EG/ 5 Apps	1 EG/ 10 Apps
1) Direct Interface				
No. of Transaction	1,000	1,000	1,000	1,000
Average time (sec)	1.7	1.52	1.38	1.41
2) Through XCREAM				
No. of Transaction	2,000	1,500	1,200	1,100
Average time (sec)	3.17	2.59	2.3	2.29

4.2 N Event Generators vs. 1 Application

The second experiment increases the number of event generators and lets the individual event generators generate events by 1,000 divided by the number of event generators. The following test result shows that the XCREAM works reliably even in the numerous RFID readers and multi-sensor environment such as

smart environment. The constant number of communication transactions results in this result.

Table 2. N EGs vs. 1 App.

Test Type	1 EG/ 1 App	2 EGs/ 1 App	5 EGs/ 1 App	10 EGs/ 1 App
1) Direct Interface				
No. of Transaction	1,000	1,000	1,000	1,000
Average time (sec)	1.7	1.45	1.24	1.17
2) Through XCREAM				
No. of Transaction	2,000	2,000	2,000	2,000
Average time (sec)	3.17	3.01	3.05	2.98

4.3 M Event Generators vs. N Applications

The final tests are designed to check the validity of the collaboration under the XCREAM framework in the real-world with varying number of the event generators and various business applications. As shown in the previous experiments, the XCREAM works better with multiple applications. Whereas, the number of event generators doesn't influence significantly on the overall performance. According to the test results obtained from the above tests, the number of applications is fixed to ten at this time and only the number of event generators is increased. An event generator generates events by 1,000 divided by the product of the number of the event generators and the number of applications.

The test result shows that the XCREAM works well in the environment where multiple applications are considered. This means the XCREAM can be applied to the complex smart environment, where the orchestration among various business applications is required. Moreover, the XCREAM provides those applications with scenario-based interfaces and makes it possible to build a complex service with the combination of multiple services.

Table 3. M EGs vs. N Apps.

Test Type	1 EG/ 10 Apps	2 EGs/ 10 Apps	5 EGs/ 10 Apps	10 EGs/ 10 Apps
1) Direct Interface				
No. of Transaction	1,000	1,000	1,000	1,000
Average time (sec)	1.41	1.38	1.32	1.41
2) XCREAM				
No. of Transaction	1,100	1,100	1,100	1,100
Average time (sec)	2.29	2.24	2.27	2.19

5. CONCLUSION

This paper focuses on the introduction of the collaborative XCREAM framework including its internal components and external interfaces as well as the verification of the framework with the performance and collaboration validity test results.

The wide adoption of the RFID/USN-enabled application services with the proposed framework accelerates rapid prototyping and deployment of the new services with the effective customization methods.

The research team is building reasoning process to the XCREAM to help the framework recognize a special situation with the combination of the context, i.e. the current status of an environment. The framework presented in this paper is the revised version of the previously published one in [8] to accommodate the new functionalities.

Improved services with a variety of combination of individual services would remarkably change our day-to-day life a lot more comfortable and safer than before.

6. REFERENCES

[1] N.D. Evans, *Middleware is the key to RFID*, 2004, DOI= http://www.rfidjournal.com/article/view/858/1/82.

[2] C. Floerkemeie, M. Lampe, RFID mddleware design – addressing application requirements and RFDI constraints, In *Joint sOc-EUSAI conference*, 2005, pp. 219-224.

[3] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo, Middleware to Support Sensor Network Applications, *IEEE Network Magazine Special Issue* Jan. 2004.

[4] M. S. Kim, Y. J. Lee, J. H. Park, USN Middleware Technology Trend, *Journal of ETRI*, Vol. 22, No. 3, Jun. 2007.

[5] Y. B. Kim, C. S. Kim, J. W. Lee, A Middleware Platform Based on Multi-Agents for u-Healthcare Services with Sensor Networks, In *Symposium on Applied Computing Proceedings of the 2008 ACM symposium on Applied computing*, 2008.

[6] S. Li, S. H. Son, and J. A. Stankovic, Event Detection Services Using Data Service Middleware in Distributed Sensor Networks, In *Information Proceedings of the In Sensor Networks*, Apr. 2003, LNCS 2634, pp.502-517.

[7] T. Liu and M. Martonosi, Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems, In *Proceedings of the ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, 2003, pp. 107-118.

[8] K. Park, J. Yun, Y. Kim, and J. Chang, Design and Implementation of Scenario-Based Collaborative Framework: XCREAM, In *Proceedings of the Intl. Conf. on Information Science and Application (ICISA 2010)*, Seoul, Korea, 2010

[9] K. Park, Y. Kim, J. Chang, D. Rhee, and J. Lee, The Prototype of the Massive Events Streams Service Architecture and its Application, In *Proceedings of the Intl. Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2008)*, Thailand, 2008.

[10] EPCglobal , *The Application Level Events (ALE) Specification*, ver. 1.0, 2005, DOI= http://www.epcglobalinc.org/standards/ale/ale_1_0-standard-20050915.pdf