2005－UBI－8（13）
2005／6／9

Ontology-based High Level Event Subscription Method for RFID Middleware

Jong-Yun Jung*, Ki-Yeol Ryu*, Jung-Tae Lee*

*Graduate School of Information and Communication,
Ajou University, Korea
{jongyun, kryu, jungtae}@ajou.ac.kr

ABSTRACT

The content-based publish/subscribe system provides an efficient method for the event subscription in RFID environment. Applications should inform the event broker of their demands for receiving the interested events. Therefore this way is well suited to the RFID system and also provides an efficient way for processing tremendous events generated between applications and event brokers. But the procedure of event subscription is a difficult and complex work for application developers or users to understand and utilize. To address this problem, we proposed the ontology-based high-level event subscription method, which adopted ontology technique to define the entities and their relations in RFID application environment.

Keywords: Ontology, Publish/Subscribe, Ubiquitous, RFID middleware, Event Subscription,

1. INTRODUCTION

The barcode system has been widely applied to various industry fields, such as manufacturing, supply chain management, and warehouse management, etc. The barcode scanner is directly linked to application, so a bridge-software so called middleware does not have been highlighted generally. However the barcode will be gradually replaced by the electric tag like RFID(Radio Frequency Identification) tag in ubiquitous environment. Unlike the barcode scanner, the RFID reader should have an ability to identify lots of tags simultaneously and also quickly transmit those data to the connected application. These works increases the communication cost in RFID network, and needs more computing power for application to process those received data from readers. It is also not easy for application developers to understand the new technique well. This is the new burden for application developers which are only concerned with business logic.

To solve above problems, the concern with RFID middleware has been growing for the last several years. The middleware in RFID environment sits between RFID readers and enterprise applications. The RFID middleware performs many operations on behalf of applications. The main works of RFID middleware is to collect tag IDs from readers and decide the destination of those data [8]. The decision process means that RFID middleware selects tag IDs that application wants to receive. So applications inform RFID middleware of their interest in advance. The description method for application's demand is similar to the event subscription in content-based publish/subscribe system. The problem is how to describe the subscription which can express application's requirements well. The event subscription methods should be simple and easy for application developer or user to use. Therefore we adopt the ontology to define the entities and their relations in RFID application environment [7]. Ontology enables the high-level event subscription. The ontology-based subscription is more understandable and familiar to human because the subscription can contain high-level semantics.

This paper proposed the ontology-based high-level event subscription method for RFID middleware. To define all entities and their relations in RFID application domain, we utilized the OWL Web Ontology Language developed by the World Wide Web Consortium [9]. In chapter 2 we present a discussion of related works. We then describe our approach to the ontology-based high-level event subscription method in chapter 3. In chapter 4 we present the implementation strategy and conclude in chapter 5.

2. RELATED WORKS

2.1 RFID Middleware

The RFID middleware collects tag identifiers(IDs) which is the 64bit or 94bit binary value(may be other format) and sends those collected data to applications. Besides tag IDs, applications receive the additional information such as reading time and readers' location etc. Auto-ID Center is the leader of RFID technology including middleware. Auto-ID center published the specification of Savant, which was the first trial to define the function of RFID middleware [8]. Savant is a data router that performs operations such as data capturing, data monitoring, data filtering, and data transmission etc. The savants basically are organized in a hierarchical tree structure. It is not appropriate to apply Savant to various applications in wide-area network like internet. The problem results from the structure of Savant. The RFID middleware can select tag IDs what application wants to receive before application informs their requirement. But Savant does not satisfy various applications' needs because of the absence of subscription methods that express application interest.

*This research is supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication 21 Century Frontier Project R&D Program in Korea

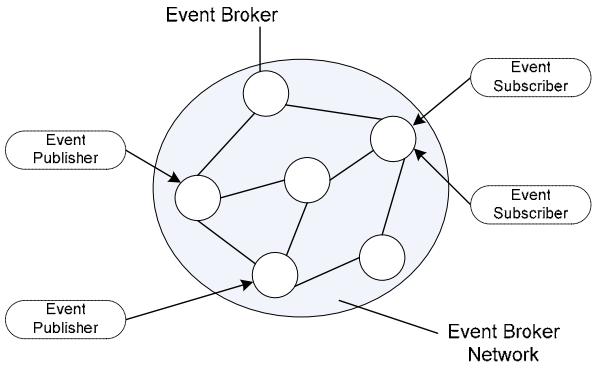

Figure 1: Publish/Subscribe Architecture

2.2 Content-based Publish/Subscribe

A content-based publish/subscribe service is considered as a useful mechanism to support integrating distributed components or applications on a wide area network. It can be implemented as a network of servers that route and deliver event notifications to those clients that are interested. Event subscribers subscribe a category of events to the event servers and publishers publish events to them. The event servers carry out selection process (determining which events match which subscriptions) and event delivery (routing matched events to interested subscribers). A content-based publish/subscribe systems provide a highly flexible selection process. The selection means that published events can be compared with event subscriptions based on their contents rather than predetermined addresses or channels.

These features facilitate for content-based systems to support the integration of highly loosely-coupled, heterogeneous, and asynchronous distributed components. For this reason, content-based subscribe/publish method provides a foundation of communication between RFID middleware and applications. Figure 2 shows the general architecture of publish/subscribe system. The content-based publish/subscribe system has already been implemented and experimented in various systems such as SIENA [1][2], and Gryphon [3]. Especially, our approach is highly influenced by SIENA which can be applied to general peer-to-peer networks.

2.3 Ontology

Ontology is one of emerging technologies and many researchers in computer science would introduce this technique into their studies. Ontology is typically a hierarchical data structure containing all the relevant entities and their relationships and rules within a certain domain. Ontology is the term used to refer to the shared understanding of some domain of interest which may be used as a unifying framework [5][6]. Our approach adopts the ontology to define the entities and their relations in RFID application environment. The World Wide Web

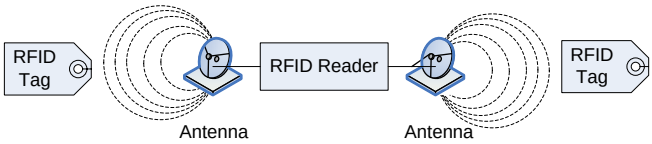

Figure 2: Reader, Antenna and RFID Tag

Consortium(W3C) developed the OWL Web Ontology Language for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF etc [9]. The OWL is one of best tools to build the shared data structure for RFID system.

3. HIGH-LEVEL EVENT SUBSCRIPTION

3.1 Event Broker Network

We can consider two approaches to process events from in RFID environment. One is the application-oriented and another is the middleware-oriented. The former is the conventional approach that has been generally applied to barcode system. But this way is not appropriate to RFID applications because tremendous data are delivered from readers. As we said earlier, this causes increase of communication cost and burdens applications. Therefore, it is essential to locate a middleware between RFID readers and applications.

After a RFID tag is detected by a RFID antenna is connected to a RFID reader in figure 2, the antenna transmits an identified tag ID to RFID reader. The reader transforms the received analog signals to the digital signals and creates message including additional data. Events are originated from situation where RFID antenna detects RFID tags. We consider these events are the basic event. That is, one basic event is correspondent to one detected RFID tag with the unique ID value(64bit or 96bit binary). After detecting tag, a RFID reader creates one event message and sends it to RFID middleware. This event message includes additional data such as the reading time and the reader identifier which reports the identified object's location.

Receiving events from reader, the RFID middleware should select events according to each application's requirements and forwarding selected data(tag IDs) to applications. The selection process is similar to the event matching in content-based event publish/subscribe system. The event publisher, event subscriber, and event broker are the main elements of publish/subscribe architecture. Above three components are correspond to RFID reader, application, and RFID middleware in RFID system. RFID middleware should support functions of an event broker. To receive interested events, applications must register their interests to middleware. This step is called event subscription in publish/subscribe system. The subscription message should include application's demands. It is almost similar to an event filter in the publish/subscribe system. We use the term of event broker instead of RFID middleware.

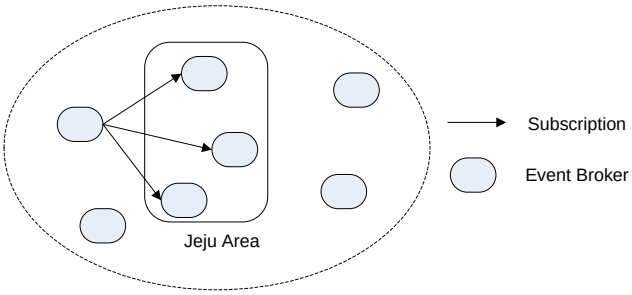

Figure 3: Low-Level Event Subscription

Many Event brokers are usually distributed in wide area, so they organize a kind of network in RFID Environment. Each event broker is logically linked to one or more event brokers, and some of them may be connected to one or more RFID readers directly. We call this network as event broker network. There are various ways to support communication among event brokers. It is reasonable to employ the event broker architecture to the RFID middleware.

An event subscription has predefined attributes which is name-value pair in the publish/subscribe system. When applications subscribe a subscription message to the event broker network, they must know the address of the event brokers in certain area, the identifier of reader, and tag IDs etc. For examples if application wants to receive events about 'certain goods stored in Jeju area', it should be aware of all event brokers in Jeju area, all readers connected event brokers, and tag IDs related to the some goods. If an application wants to receive events about 'Trucks arrived at Jeju area today', it should create a subscription message containing '(location=EB1 or location EB2 or location EB3) and (tag => 1001 ^ tag <=1999)'. Figure 1 shows the low-level event subscription on the event broker networks. The low-level event subscription means that subscription is described in detail. This is very inconvenient and difficult way for application developer to understand. This may be suited to applications for small-area, or applications in limited domain, but not to applications for wide-area, like nation. To help developer or user work easily, it is necessary for to provide new methods for event subscription. We designed high-level event subscription methods combined with ontology technique. We will discuss the high-level description method in the next chapter.

3.2 High-Level Event Subscription

To subscribe a subscription message to RFID middleware, application must describe the low-level subscription using tag ID, subfields in tag ID, and reader ID etc. It is difficult for application developers or users to know above all matters when they implement or use applications. The low-level event subscription is generally described by combination of primitive elements, which are attributes, values and operators. Attributes are specified by reader IDs, manufacturer IDs, object class IDs etc. Operators are specified by equality(=), inequality(!=), greater than(>) etc. If application wants to receive events about 'the exact time that truck loaded with computers pass the main gate', the subscription message is described containing a content like 'Tag id is 2, event broker's id is a, and reader's id is 100 and 200'.

However if the event broker supports high-level event subscription, we can describe the subscription message containing 'notify someone(e.g. application) when truck loaded with computers pass the main gate'. The high-level event subscriptions have a common pattern in respect of the content of subscription. The content of most subscriptions can be described by 'any object with any tag arrives at any areas' or 'any object with any tag is identified by any reader at any areas. These subscriptions consist of three main parts; 'area or region', an 'action', and 'object' with RFID tag. The Action is highly relevant to the objects identified by readers.

1. area : actually represents the event brokers in the area
2. action : represents 'storage in warehouse', 'delivery', in domain like physical distribution and logistics. This means the RFID readers
3. object : represents real objects like truck, pallet, and box etc.

The shared data is the data structures contain above entities, sub-entities, and their relations. It is reasonable that applications require more data in addition to Tag ID in ubiquitous environment. Applications would need temperature, humidity, or other environmental data in any area when RFID tags are identified by readers. Therefore, it is necessary for the event broker to provide more expressive method for the event subscription. The ontology definitions for above entities and the subscription message will be discussed in next chapter.

3.3 Processing Subscription Message

The process of event subscription begins when application sends a subscription message to an adjacent event broker. The subscription messages actually delivered to the event subscription helper. The subscription helper offers common functions used by all event brokers. Subscription helper can be implemented as an independent service and be the part of event broker as a common module. The subscription helper analyzes the subscription message and propagates it to several event brokers which are relevant to the content of message. To analyze the subscription message, subscription helper refers to the shared data which is a hierarchical data structure containing all predefined entities in application domain. We will discuss the shared data in the following chapter. Using subscription helper, applications transparently sends a subscription message to even brokers though they do not know which event broker receives the subscription message. Figure 5 shows the message flow of processing event subscription in the event broker network.

As we mentioned above, the high-level event subscription provides the power of expressiveness and the better readability.

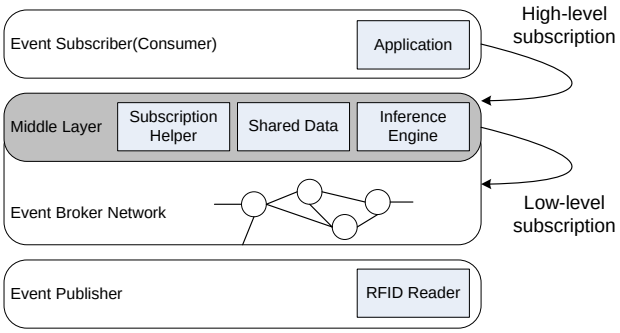

Figure 4: Architecture for Ontology-based High-level Event Subscription

Application developer or user can easily understand how to describe the contents of subscription and create a subscription message. However, the question is that the high-level subscription should provide same semantics and expressiveness as low-level subscription does. The expressiveness of event subscription message is highly dependent on application domain.

4. IMPLEMENTATION

4.1 Proposed Architecture

Figure 4 shows our proposed architecture for ontology-based high-level event subscription method. The event broker network holds the middle layer, which includes the subscription helper and the shared data. The subscription helper provides the transparent service to applications. This means that applications do not need to know where to send a subscription messages. The middle layer receives the high-level subscription messages and also propagates the low-level subscription messages to event brokers.

If an application wants to receive the events from Jeju area and Korea, it must send two subscription messages to the event broker. This is inefficient way. To remove the unnecessary subscription, we choose the ontology technique. The shared data is a hierarchical data structure containing all relevant entities and their relations. It is appropriate to include the function of the extendible inference for high-level event subscription. The subscription helper makes use of the shared data to analyze and reason the subscription. Subscription helper also converts high-level subscription to low-level subscription which event broker can understand. Using high-level event subscription methods decreases the number of unnecessary and redundant subscription messages in the event broker network. This decreases total cost of communication between event brokers and applications and eliminate the burden of event broker.

4.2 Process of High-Level Event Subscription

After subscription helper analyzes and reasons the subscription message using ontology for high-level subscription, it forward the message

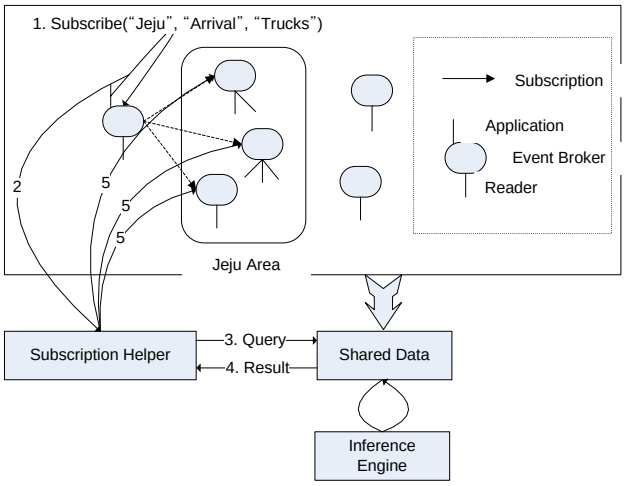

Figure 5: Message Flow of High-Level Event Subscription

to the actual event brokers collecting events from RFID readers. Table 1 shows the sequence of action performed by subscription helper. This is the internal logic of the subscription helper just receiving the subscription message.

Table 1: Pseudo-Code about subscription process

1) Application(A1) subscribe an adjacent event broker(EB1)
   ex) Subscribe("area", "arrival", "all truck")
2) EB1 send subscription message(SM) to the subscription helper(SH)
3) SH analyze SM and obtain event brokers using the shared data, SM's receiver.
   ex) query name : area
   query : EventBroker.location is(contains, begins with, ends with) "Jeju area"
   search results : EB4 v EB5 v EB6
4) To find all readers connected to obtained event brokers, send query message.
   ex) query : (Reader.alias is "arrival") ∧ (Reader.EventBroker is "EB4" ∨ Reader.EventBroker is "EB5" ∨ Reader.EventBroker is "EB6")
   Search results : R2.EB4 ∨ R1.EB5 ∨ R2.EB5 ∨ R5.EB6
5) Convert the contents of subscription to actual sentence..
   ex) query name : all truck
   query : tag ≥1001 ∧ tag ≤ 1999
6) Create a subscription message for event borker.
   ex) "If tag id's value ranges from 1001 to 1099 in [EB4:R2] [EB5:R1,R2] [EB6:R5], then publish events to EB1" (this sentence can be expressed by XML)
7) Send the message to event broker.

If one event broker is located in JeJu, it also is located in Korea. Therefore, if both 'Event broker(EB1) is in Jeju' and 'Event broker(EB1) is in Korea' are described in the subscription message, the inference engine can find out the relation that Jeju is a local area of Korea. This way removes

unnecessary search and comparison work about whether an event broker is in Jeju or Korea.

4.3 Subscription Message

There are four message types for subscription. They are closely related to the area of event creation and the way of describing events. The message has three parameters, area, object, and action.

1) 0 – parameter subscription message
This message has no parameter. If application subscribes this message, all events are delivered to the subscriber. This way will cost more computing power, network resource, and communication cost etc.
ex) subscribe()

2) 1 – parameter subscription message
1-parameter message has one parameter about area information. The area indicates event brokers located in the area. Those event brokers send all received event to the subscriber.
ex) subscribe('area')

3) 2 – parameters subscription message
The action is added to 1). The action indicates all readers connected to event brokers in the registered area. The action parameter is dependent on the application domain.
ex) subscribe('area', 'action')

4) 3 – parameters subscription message
The Object identified by readers is added to 3). This parameter indicates tag IDs.
sx) subscribe('area', 'action', 'object')

4.4 Ontology for Physical Distribution & Logistics

We designed ontology definitions for the application domain of physical distribution and logistics because it is difficult to define general-purpose ontology. The ontology definitions are showed in the following tables. The targets are main entities in RFID environment. They are the event broker, the reader, the area, and the action. Firstly, the area class can have one or more aliases and be included to other wide area. The event broker class is included to one or more areas and has one or more readers. This has communication channels for subscriber. The reader class has one name and one or more actions. The action class has one or more aliases. For writing ontology definitions, we use the ontology editor, Protégé-2000, which is a graphic-based tool [7].

The proposed approach needs the further investigation about the shared data. A large of data for reasoning the subscription message should be pre-constructed in advance. The shared data is a complex and large ontology structure. Therefore the efficient storage method and the fast search ability to process queries from the subscription helper are critical issues.

Table 2: Ontology Definition about Area(Region) Class

```
<owl:Class rdf:ID="Region">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasName" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasSuperSet" />
      <owl:minCardinality>0</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
</owl:Class>
```

Table 3: Ontology Definition about Event Broker Class

```
<owl:Class rdf:ID="EventBroker">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#belongRegion" />
      <owl:minCardinality>0</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasReader" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty
      rdf:resource="#hasSubscribeChannel" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
</owl:Class>
```

Table 4: Ontology Definition about Reader Class

```
<owl:Class rdf:ID="Reader">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasName" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasAction" />
      <owl:minCardinality>0</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
</owl:Class>
```

Table 5: Ontology Definition about Action Class

```
<owl:Class rdf:ID=”Action”>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=”#hasName” />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restiction>
  </rdfs:subClassOf>
</owl:Class>
```

5. CONCLUSION

The content-based publish/subscribe system provides an efficient solution for event subscription in RFID environment. Applications should inform the RFID middleware of their demand for receiving the interested events. This way is well suited to the RFID system and provides an efficient way for processing tremendous events generated between readers and event brokers. The approach decreases total cost of communication between event brokers and applications and eliminate the burden of event broker. For this reason, we applied the content-based publish/subscribe method to the RFID middleware and design event broker architecture for the high-level event subscription. But the process of event subscription is difficult and complex work for application developer or user. To address this problem, we proposed the ontology-based high-level event subscription method. Using high-level subscription methods decreases the number of unnecessary and redundant subscription messages in the event broker network. Our approach also adopts the ontology to define the entities and their relations in RFID application environment. We designed ontology definitions for the application domain of physical distribution and logistics because it is difficult to define general-purpose ontology. The ontology definitions define the event broker, the reader, the area, and the action. We will study the forwarding method for event broker and data structure for storing forwarding information. A continuous examination of the subscription method is needed to clear the structure of the subscription message. Another direction of this study will be about the high-level event model.

REFERENCE

[1] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," ACM Transactions on Computer Systems, 19(3), Aug. 2001, 332-383.

[2] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," ACM Transactions on Computer Systems, 19(3), Aug. 2001, 332-383.

[3] M. K. Aguilera, R.E. Strom, D.C. Struman, M. Astley, T.D. Chandra, "Matching Events in a Content-Based Subscription System," 18th ACM Symposium on Principles of Distributed Computing(PODC1999), Atlanta, GA May 1999, pp. 53-61.

[4] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R.E. Strom, D.C. Sturman, "An Efficient Multicast Protocol for Content-Based Publish-Subscribe system," IBM T.J. Watson Research Center.

[5] M. Uschold and M. Gruninger, “Ontologies:principles, methods and applications,” The KnowledgeEngineering Review, 11(2)93-136, 1996.

[6] T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing,” Int. J.Humna-Computer Studies, 43:907-928, 1995.

[7] N. Noy, M. Sintek et al., “Creating Semantic Web Contents with Protege 2000", IEEE Intelligent Systems vol 16 No.2, 2001.

[8] Auto-ID Center, Auto-ID Savant Specification 1.0 – Version of 1 September 2003, http://www.epcglobalinc.org/standards_technology/Secure/v1.0/WD-savant-1_0-20030911.doc

[9] W3C, Web Ontology Language (OWL), http://www.w3.org/2004/OWL/