AN EVENT-BASED MIDDLEWARE FOR ANALYZING CONTEXT INFORMATION UNDER USN ENVIRONMENT

Yongmi Lee*, Kwang Woo Nam**, Hi-Seok Kim***, Keun Ho Ryu*

* Dept. of Computer Science, Chungbuk National University, Korea
410 Sungbong-Ro, Heungduk-gu, Chungbuk, Korea, 361-763
{ymlee, khryu}@dblab.chungbuk.ac.kr
** Dept. of Computer Information Science, Kunsan National University, Korea
kwnam@kunsan.ac.kr
*** School of Electronics and Information Engineering, Cheongju University, Korea
khs8391@cju.ac.kr

ABSTRACT: With the proliferation of advanced wireless network and sensor technologies, smart devices under USN(ubiquitous sensor network) environment are capable of collecting context information such as temperature, humidity, weight, and location about objects at real time. Therefore, applications must be able to analyze collected information and notify useful information to wanted users timely. This service can be realized by implementing an event-based middleware. In the middleware, event messages collected from physical environment will be filtered according to profiles that users define in advance and the result will be sent to the interested users. In this paper, we present XML-based event model, ECA-based profile model, and the architecture of an event-based middleware suitable to USN environment. We will also model and describe them using the examples of logistics area. By implementing the system based on the design above, the middleware enable applications or users to easily access to physical sources. The proposed middleware can also apply to not only logistics area but also other various areas under USN environment such as intelligent traffic control system, national disaster management system and u-medical system.

KEY WORDS: Event-based middleware, Context Information, Event, Profile definition, u-Logistics

1. INTRODUCTION

Advance in wireless network and sensor technology allows applications to collect context information such as temperature, humidity, weight, location, and so on from intelligent objects at real time. Users under such USN environment will expect to know context information about interest of objects at real time.

We can support these services by implementing an event-based middleware that analyzes newly collected context information and notifies interest of information to wanted users timely. In that service, context information about objects corresponds to event messages and interest of information corresponds to user profile, respectively. Therefore, the systems must support data models that can accurately express various event sources and profiles, and present algorithms that can reduce the cost in event filtering.

In this paper, we focus on implementing an event-based middleware suitable to USN-based environment. The central issues of event-based system are how the data are exchanged and which filtering techniques are used. To do this, we suppose a XML-based event model and an ECA-based profile model, and explain the models by giving an illustration in u-logistics area. We also present the architecture of the middleware based on the models and a filtering algorithm that divides into two steps by separating selection predicates and function predicates. This technique intends to reduce filtering costs by separating complex computations from relatively simple selection predicates.

The proposed middleware can apply to not only logistics area but also other various areas under USN environment such as intelligent traffic control system, national disaster management system and u-medical system.

The rest of this paper is organized as follows. Section 2 gives a brief overview of works related to data modelling and filtering techniques. Section 3 models event and profile and Section 4 designs the architecture of event-based middleware and its filtering technique. In section 5, we implements the middleware based on the data models and architecture. Finally, we conclude in Section 6.

2. RELATED WORK

An event denotes the state transition of objects[Hinze, 2003]. Such changes will be delivered to any system after transforming the events into a specific format. At this time, the most simple method is to enumerate predicates that consist of a pair of (attribute, value) like Siena [Carzaniga, 1998], LeSubscribe[Pereira et al., 2001], and Yeast[Krishnamurthy et al., 1995] or express the event into XML format that has hierarchical structure like YFilter[Diao et al., 2003]. On the other hand, a profile denotes interest about objects pre-defined by users. After receiving an event, a system tries to match it with all the profiles stored in system[Hinze, 2003]. Therefore, the

volume and format of the profiles affect the capacity of event filtering in the system. LeSubscribe[Pereira et al., 2001], Siena[Carzaniga, 1998], and Elvin[Segall et al., 1997] support a profile model that consists of a pair of (attribute, value). Yeast[Krishnamurthy et al., 1995] and NiagaraCQ[Chen et al., 2000] support a profile definition language based on ECA(Event Condition Action) paradigm. Besides, some XML-based approaches like [Diao et al., 2003; Tian et al., 2004] define profiles using XPath expression. Expressing event and profile by a pair of (attribute, value) is simple but supports limited expression, so we consider a XML-based event model and an ECA-based profile model to support more power of expression.

Event filtering is based on the semantics of the filtering engine of the system and profile definition is also limited by the capacity of the filtering engine. Therefore, there exists an unavoidable trade-off between the profile definition language and the semantics of filtering engine. Database-based approaches like NiagaraCQ[Chen et al., 2000] process profiles as data, so their matchings are evaluated by join query in DBMS. On the other hand, because memory-based approaches process a profile as filter about message, the profile is maintained in main memory. After generating an in-memory decision tree[Aguilera et al., 1999] or a FSA(finite state automaton) per profile to store profiles, the approaches examine whether an event matches profiles. Database-based approaches must pay expensive costs to perform join operation, but the volume of memory in terms of the number of profile wasn't largely limited. In memory-based approaches, the number of profile affects the volume of memory, so this isn't suitable to applications that have a great number of profiles. Considering the filtering is conducted in middleware-level, this paper employ a database-based filtering approach as well.

3. EVENT & PROFILE MODELLING

Users are interested in changes occurring to object, so an event-based middleware must support data models that express context information about objects to satisfy such demands.

3.1 XML-based Event

An event is context information about an object occurring in physical environment. Therefore, when modelling them, it is common to model an event using observers such as RFID reader, temperature sensor, humidity sensor, and GPS receiver that are capable of observing environmental attributes or objects. Modelling focused on objects may limit common and flexible framework in the power of its expression.

Now, we consider XML syntax as a structure to present event model and explain it using a simple example in u-logistics area. That is why XML syntax has flexible structure to express events as messages having hierarchy and can guarantee interoperability between heterogeneous platforms. Figure 1 is the example of a XML-based event model expressed by BNF(Backus Nour Form) syntax. BNF syntax is convenient to explain XML-based models because it describes symbols in "symbol ::= expression".

```
Sensor ::= <Id><Observation>
Id ::= <String>
Observation ::= <Type><Datetime>(<Status><Tag>|<Position>)
Type ::= 'Detect' | 'Location'
State ::= 'Loading' | 'Unloading' | 'Departure' | 'Arrival'
Datetime ::= <StringLiteral>
Tag ::= <EPC> {<EPC>}
EPC ::= <StringLiteral>
Position ::= <Tag><Latitude><Longitude><Velocity><Course>
Latitude ::= <Degree><Minute><Second>
Longitude ::= <Degree><Minute><Second>
Velocity ::= <StringLiteral>
Course ::= <StringLiteral>
Degree ::= <StringLiteral>
Minute ::= <StringLiteral>
Second ::= <StringLiteral>
```

Figure 1. XML-based event model

This model assumes that a system collects its location from trucks fitted with a GPS receiver and also collects context information about each item from trucks and warehouses fitted with a RFID reader. Table 1 is listing the source and type of events that can occur at this time.

Table 1. The source and type of event

Event Source	Object	Type	State
RFID reader	An item	Detect	Loading
			Unloading
	A truck		Departure
			Arrival
GPS receiver		Location	-

"Sensor::=<ID><Observation>", the first line of figure 1, denotes that a sensor has an unique identifier and observes something from physical environment. The value of an attribute 'Type' of the third line will be 'Detect' or 'Location' as listed in table 1. If the value of an attribute 'Type' is 'Detect', it denotes that the event is observed by a RFID reader and has one of 'Loading', 'Unloading', 'Departure', or 'Arrival' as the value of an attribute 'State'. On the other hand, an event observed by a GPS receiver can have coordinates such as longitude and attitude as the lower elements.

3.2 ECA-based Profile

A profile pre-defined by users is stored in system and filtered to newly occurring events at real time. Therefore, profile must be transformed into specific format like events as well.

This paper consider ECA paradigm corresponding to a rule in active database to present profile model. That is why the paradigm is easy to describe conditions(C) that must be evaluated and actions(A) that must be performed as its result, when an event(E) occurs. Figure 2 shows an ECA-based profile model adaptable to u-logistics area expressed by BNF syntax.

The first line of this model denotes that each profile is based on ECA paradigm and when events that correspond to 'Detect' or 'Location' occurs, conditions described in symbol 'ConditionExpr' is evaluated. Our profile model can describes protocol type and destination of messages to be sent as its result in 'ActionList'.

```
Profile ::= 'Event' <EventExpr>
            'Condition' < ConditionExpr>
            'Action' <ActionList>
EventExpr ::= 'Location' | 'Detect'
ConditionExpr ::= <PredicateList> ['AND' <Operation>]
PredicateList ::= <Predicate> {<LogicalOP> <Predicate>}
Predicate ::= <Literal><ArithmeticOP><Literal>
LogicalOP ::= 'AND' | 'OR'
ArithmeticOP ::= '>' | '>=' | '=' | '<>' | '<' | '<='
Operation ::= <DistanceExpr> | <IntervalExpr>
DistanceExpr ::= 'Distance('<NumLiteral> ',' <NumLiteral>
                 ',' <NumLiteral>')'
IntervalExpr ::= 'Interval(' <NumLiteral> ')'
ActionList ::= 'Send('<Protocol>,<StringLiteral>')'
               {', Send('<Protocol>,<StringLiteral>')'}
Protocol ::= 'SMS' | 'E-mail'
Literal ::= <StringLiteral> | <NumLiteral> | <DateLiteral>
NumLiteral ::= <FloatLiteral> | <IntegerLiteral>
```

Figure 2. ECA-based profile model

"ConditionExpr::=<SelectionList>[<LogicalOp><Function>]", the fifth line of figure 2, denotes that the profile comply with function predicates that need more complex computation as well as selection predicates such as relative operators. In this model, there are function predicates that compute time interval and Euclidean distance.

4. DESING OF EVENT-BASED MIDDLEWARE

An event-based middleware serves as an engine that performs filtering between events and profiles. Therefore, we propose the architecture of the middleware and a filtering algorithm.

4.1 The architecture of middleware

Generally, most of event-based systems consist of three modules such as Observer, Filter, and Notifier to connect suppliers and users of context information[Hinze, 2003]. In addition to three modules, we organize additional modules, EParser and Profile Manager. Figure 3 shows the architecture of the middleware and data flow between each module.

Observer observes events about objects from physical USN environment and sends the messages to the system using TCP/IP socket connection after transforming the events into XML-based messages.

Filter is the core module of an event-based middleware that performs filtering between newly occurring events and pre-defined profiles.

Notifier generates notifications to be sent as the result of filtering. In our proposed middleware, the results are dispatched using SMS(Short Message Service) or e-mail as well as system and are also maintained in Notification Repository.

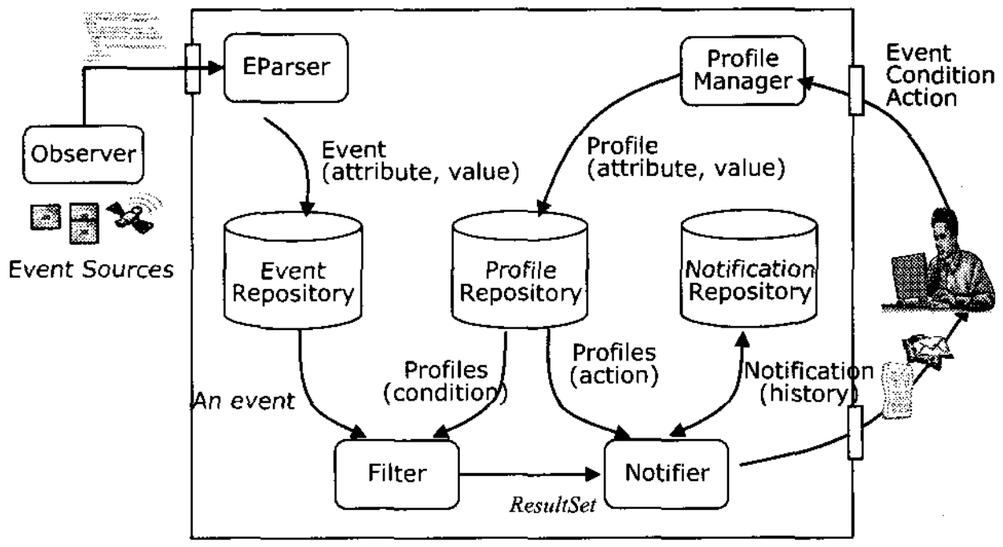

Figure 3. The architecture of event-based middleware

EParser parses XML-based event messages to a set of elements and stores them in Event Repository. At this time, each event is stored in corresponding repository according to its event type to increase the efficiency of event filtering.

Profile Manager helps that users easily define profiles and manages profile. After extracting a set of elements from profile syntax, this module divides them into action part and condition part. The condition part is divided into parts for selection predicate and function predicate again. Likewise, the predicates of condition part are stored in corresponding Profile Repository according to its event type to increase the efficiency of event filtering.

4.2 Two-step event filtering

As mentioned in section 2, event filtering is based on the semantics of the filtering engine of the system and profile definition is also limited by the capacity of the filtering engine. Therefore, a middleware system must support filtering algorithms that can minimize a trade-off between the profile definition language and the semantics of filtering engine. To do this, we propose a filtering algorithm divided into two steps by separating selection predicates and function predicates. This technique intends to reduce filtering costs by separating complex computations from relatively simple selection predicates.

Moreover, our proposed technique stores the whole selection predicates as it is. This denotes that because our proposed technique is based on database operation, it can use operation supported by DBMS without additional operation like PSoup[Chandrasekaran et al., 2003]. Consequently, the technique is able to support non-equality and comparison operator as well as equality operator. The technique gives flexibility in environment that needs additional operation by considering function predicates,.

5. IMPLEMENTATION

The proposed middleware was implemented in JDK 1.4 and Oracle 9i under system environment supporting Pentium 2.0GHz, RAM 1 GBytes, and Windows XP Professional.

As showed in Figure 4, Profile Manager consists of an attribute panel that can search the attributes of all the profiles defined by a user in tree format and a profile

panel that can describe a selected profile in detail. The value of each attribute can define using attribute window showed at the foot of figure 4. Moreover, we maintain XML format based on DOM tree so that users can easily search and update profiles.

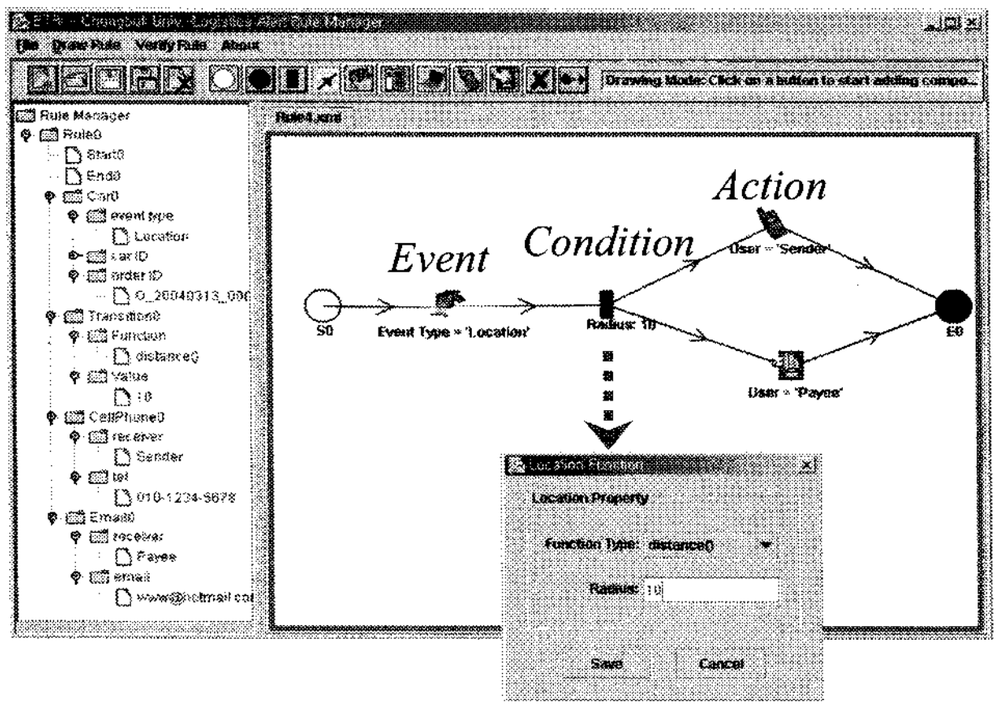

Figure 4. Profile Manager

In paring XML-based messages, we used both SAX parser and DOM. The former was used to parse event messages because it is lighter than a tree-based parser and the latter was used to help loading profiles on the profile panel based on GUI.

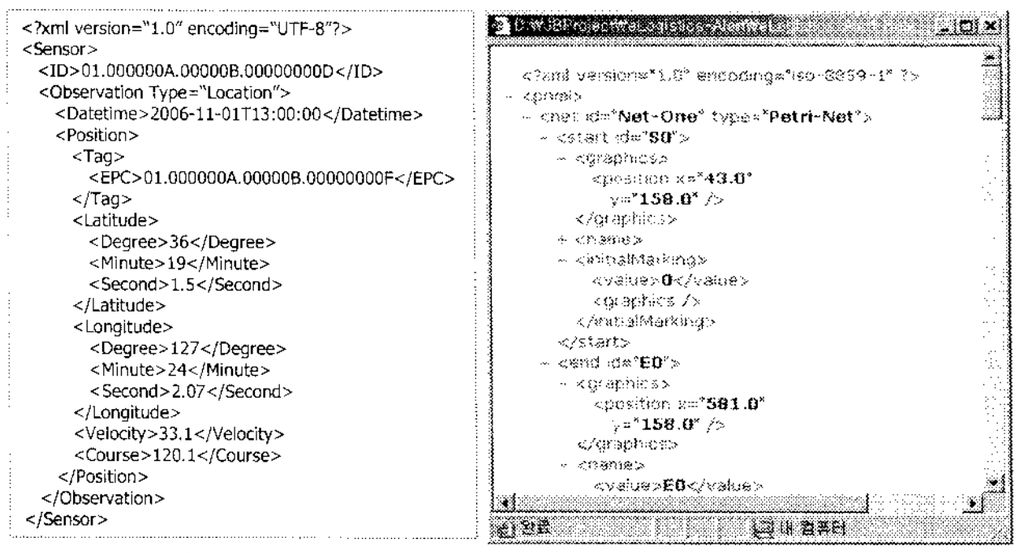

Figure 5. Messages using XML structure

Figure 5 shows the examples of a XML-based event message and a profile stored in DOM tree format.

6. CONCLUSIONS

With Advance in wireless network and sensor technology, applications are capable of collecting context information from intelligent objects at real time. Therefore, this paper proposed the event-based middleware that can adapt to varying environment and that can analyze and support the context information at real time. First, we presented a XML-based event model and an ECA-based profile model under USN environment. Second, we presented the architecture of the middleware and the filtering algorithm. In the algorithm, we intended to reduce filtering costs by separating complex computations from relatively simple selection predicates. Consequently, by implementing the proposed middleware, we are able to track and trace objects at real time.

Currently, works to extend even observers such as temperature sensor, humidity sensor, and so on and to include data mining modules for outlier detection and prediction are ongoing.

REFERENCES

Aguilera, M. K., Strom, R. E., Sturman, D. C., and et al, 1999. Matching Events in a Content-based Subscription System. In Proc. of PODC.

Carzaniga, A., 1998. Architectures for an Event Notification Service Scalable to Wide-area Networks. Ph.D. Thesis, Politecnico di Milano, Dipartimento di Elettronica e Informazione.

Chandrasekaran, S. and Franklin, M. J., 2003. PSoup: A System for Streaming Queries over Stream Data. *The VLDB Journal*, 12(2), pp.140-156.

Chen, J., 1DeWitt, D., Tian, F., and Wang, Y., 2000. NiagaraCQ: A scalable continuous query system for internet databases. In Proc. of ACM SIGMOD, Dallas, TX.

Diao, Y., Altinel, M., Franklin, M. J., Zhang, H., and Fischer, P. M., 2003. Path Sharing and Predicate Evaluation for High-Performance XML Filtering. In Proc. of ACM TODS.

Hinze, A., 2003. A-MEDIAS: Concept and Design of an Adaptive Integrating Event Notification Service. Ph.D. Thesis, Freie Universitt Berlin.

Krishnamurthy, B. and Rosenblum, D. S., 1995. Yeast: A General Purpose Event-Action System. *IEEE Transactions on Software Engineering*, 21(10), pp.845-857.

Pereira, J., Fabret, F., Jacobsen, H., and et. al, 2001. LeSubscribe: Publish and subscribe on the web at extreme speed. In Proc. of ACM SIGMOD, Santa Barbara, CA.

Segall, B. and Arnold, D., 1997. Elvin has left the building: A publish/subscribe notification serivce with quenching. In Proc. of AUUG, 1997.

Tian, F., Reinwald, B., Pirahesh, H., Mayr, and T., Myllymaki, J., 2004. Implementing A Scalable XML Publish/Subscribe System Using Relational Database Systems. In Proc. of SIGMOD.

ACKNOWLEDGEMENTS

This research was financially supported by the Ministry of Commerce, Industry and Energy (MOCIE) and Korea Industrial Technology Foundation (KOTEF)

through the Human Resource Training Project for Regional Innovation.