XML-based Event Notification System for Large Scale Distributed Virtual Environment

JinHyun Tak *, Seihoon Lee **, Changjong Wang*
*Dept. of Computer Science & Engineering, Inha University, KOREA
**Dept. of Computer Engineering, Inha Technical College, KOREA
tak2023@hanmail.net, seihoon@true.inhatc.ac.kr, cjwangse@inha.ac.kr

Abstract: In this paper, we propose XML based event notification system in distributed virtual environments. The proposed system imports a notifier model of which event receivers can add and remove dynamically without affecting system. The system defines basic event types, and then those events express XML, which is standard of information transmission in the WWW to be operating independently specific application on other environments. Because the system expresses event using XML, it can store occurred event log data and can use filtering and retrieval of event using log data to improve effectiveness. Also, filtering is improved with which system allow participant to register their interested event. Finally, our system can utilize interoperability between applications of other virtual reality as well as XML based applications

.

1. Introduction

Recently, it is increased interest about virtual reality according to developing Internet technology including VRML rapidly, due to development of computing ability and advancement of network (VRML 1997). Especially, it becomes to need event notification model for interoperability between virtual reality systems (Hagsand 1996, Funkhouser 1995).
Nowadays, most virtual reality systems use their own system and event notification model. There are problems which systems of different environment have difficulty of which apply to each event and which they are difficult to interoperability with occurred events. Therefore, it needs system which can accept events occurred in other virtual environments and which can filter through events occurred between virtual applications. Especially, filtering is a need part because events are occurred too much in virtual environments. Also, in case of collaborative work, events should be managed for event filtering and retrieval (Funkhouser 1995, Brandt & Kristensen 1997).
Therefore, we propose XML based event notification system in distributed virtual environments. The proposed system is a notification model, and we design notifier which implement notification model provided dynamic event registration of event receiver and which has property of separating event senders with receivers. This notifier is appropriate with virtual environments, which make a change between event senders and receivers frequently. Also, this system is independent of application domain because it defines event format using XML, which is content-based structured data and self-describing standard of information transmission. The system filters events using XML and makes easy retrieval of event through database of occurred event log data using XML. Finally, this system can utilize interoperability between applications of other virtual reality as well as XML based applications

2. Related Works

We will survey about event distribution model and XML (eXtensible Markup Language), which is standard of data transmission type.

2.1 Event Distribution Model

Classic model

The simplest event distribution model is that event sender pass events to receiver directly. This way has a disadvantage, which is event receiver should have reference of all event senders. The followings are approaches to overcome this disadvantage.

Mediator model

This approach is that Mediator is existed between event receivers and senders. That is, event senders send events only Mediator and then Mediator pass events to event receivers. The disadvantage of this model is that it should be modify Mediator' s list, which is managed to event receivers if event receivers increase (Gupta et al. 1998, Gamma et al. 1997).

Observer model

This approach allows event receivers to register their interest events, and event receivers receive only their registered events when events occur. This Observer approach has an advantage, which can extend system without affecting other participants although event receivers increase. However, event receivers have to know about information of event senders (Gupta et al. 1998, Gamma et al. 1997).

Event Notifier model

This approach combines the benefits of both the Mediator and Observer approaches, as shown in Figure 1. Like in the Mediator approach, we have a central event service that mediates notification, so that event senders and event receivers do not need to know about each other. Like in the Observer approach, a registration system allows us to add and remove observers (called subscribers) dynamically (Gupta et al. 1998, Gamma et al. 1997).

Figure 1 is illustrated UML notation for event notifier model (Gamma et al. 1997).

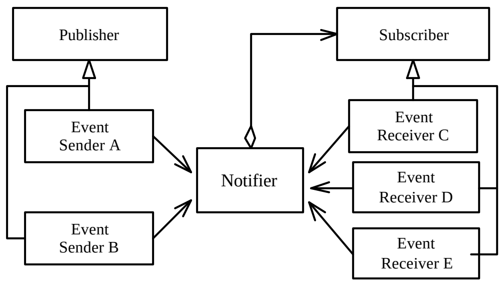


Figure 1. UML notation for Event Notifier Model

2.2 XML

XML is defined as subset of SGML (XML 1998). It formalize data in simple and consistent way, and provides structured data type for transmission. On heterogeneous platforms, it support to define contents clearly, and to gather meaningful search results.

XML documents are combined with DTD (Document Type Definition), and is regarded valid when based rules defined in DTD. DTD defines tag used in XML documents, processing sequences, and specific tags can include others. XML can provide easily extensible and structured form of data.

Using XML, we can accept requirements of content based event format to standard of content-based structured data and self-describing information transmission. Also, we can offer event filtering and QoS through filtering and then it is very useful database of event log information.

3. Design of System

The event-processing scheme in legacy VR environment is dependent to the implementing system, and the definition of events is dependent to applications of system. Therefore, we propose a generic event notification system to eliminate these dependencies.

The proposed system designs Notifier to interface with external services having different VR environments, and performs event filtering to reduce the amount of events. Additionally, we describe events based on XML, so it increases the efficiency of interaction with external services and makes searching convenient through event logs management.

Figure 2 shows the architecture of designed event notification system.

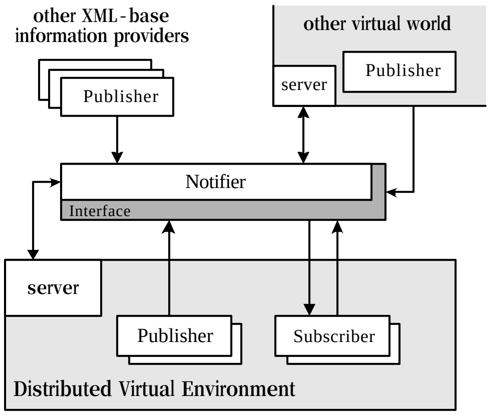


Figure 2. Event Notification Architecture

The designed notifier has following features.

- It simplifies complex event propagation routes by providing transparency between publisher and subscriber.
- It makes an easy to insert or delete the receiving events of shared objects or specific information dynamically regardless of system when it uses different interaction ranges according to avatar. The definition of method about these interaction ranges is dependent on the management of VR environments, but it is independent of notification system.
- It provides dynamic filtering facilities. Subscribers can personalize events propagated through the registration of filter about specific events to notifier. Events described in XML are also possible to filtering of event contents using searching and conversion of XML tags because XML based events are hierarchical and self-describing.
- It can interact with legacy applications dependent on domain. And it enables to convert XML formed events with the registration of applied application interfaces to notifier.

3.1 event specification

The basic event types for accepting events in VR environment are defined as follows.

We use XML to describe events because it is possible content based filtering and is convenient to build database of log information. In addition, these information are useful to retrieve events generated, and write working information or behavior patterns of participants in VR environment.

- ***Action Event***

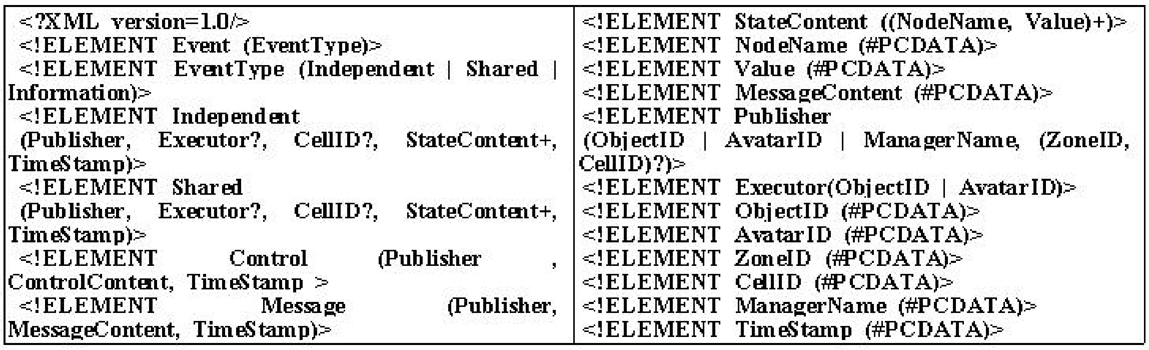
Action events are generated when the object status is changed, or when active behaviors are happened. These events are related with persistency maintenance of VR environment. They are defined as follows according to the degree of synchronization.

- Independent interaction event: This event is generated independently regardless of participants such as movement of custom objects or changes of avatar location. Recent events can be substituted for previous events because it does not effect other participants immediately. However, this should be reflected upon other clients for consistency.
- Shared interaction event: This kind of event effects other clients immediately, so it has to be reflected firstly of all than independent events. For example, events generated in cooperative work should be reflected on participants firstly.

- ***Information event***

This event has no connection with the synchronization of virtual environment. It is not notified for consistency immediately, but should be transmitted. This event is generated when mail type data is propagated to internal/external virtual environment, or when the previous log data is requested from system.

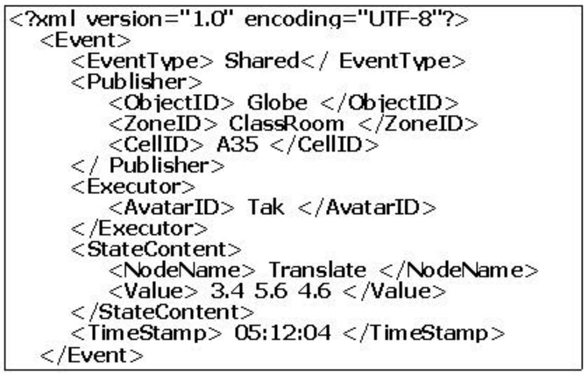
Figure 3 is a DTD for data structure of events, and Figure 4 is a sample event that describes the data structure of shared Interaction event using XML Syntax. This is event that avatar named “Ellio” makes a movement the target object of cooperative work, “CoWork”, according to the value of attribute <vec3fx>.



```
<?XML version=1.0/>
<!ELEMENT Event (EventType)>
<!ELEMENT EventType (Independent | Shared |
Information)>
<!ELEMENT Independent
(Publisher, Executor?, CellID?, StateContent+,
TimeStamp)>
<!ELEMENT Shared
(Publisher, Executor?, CellID?, StateContent+,
TimeStamp)>
<!ELEMENT Control (Publisher ,
ControlContent, TimeStamp >
<!ELEMENT Message (Publisher,
MessageContent, TimeStamp)>
```

```
<!ELEMENT StateContent ((NodeName, Value)+)>
<!ELEMENT NodeName (#PCDATA)>
<!ELEMENT Value (#PCDATA)>
<!ELEMENT MessageContent (#PCDATA)>
<!ELEMENT Publisher
(ObjectID | AvatarID | ManagerName, (ZoneID,
CellID)?)>
<!ELEMENT Executor(ObjectID | AvatarID)>
<!ELEMENT ObjectID (#PCDATA)>
<!ELEMENT AvatarID (#PCDATA)>
<!ELEMENT ZoneID (#PCDATA)>
<!ELEMENT CellID (#PCDATA)>
<!ELEMENT ManagerName (#PCDATA)>
<!ELEMENT TimeStamp (#PCDATA)>
```

Figure 3. DTD for Data structure of Events



```
<?xml version="1.0" encoding="UTF-8"?>
  <Event>
    <EventType> Shared</ EventType>
    <Publisher>
      <ObjectID> Globe </ObjectID>
      <ZoneID> ClassRoom </ZoneID>
      <CellID> A35 </CellID>
    </ Publisher>
    <Executor>
      <AvatarID> Tak </AvatarID>
    </Executor>
    <StateContent>
      <NodeName> Translate </NodeName>
      <Value> 3.4 5.6 4.6 </Value>
    </StateContent>
    <TimeStamp> 05:12:04 </TimeStamp>
  </Event>
```

Figure 4. Event Format using XML syntax

The sub-element of <Content> tag is used to describe the node name and value of VRML, which should be changed by this event. Events using XML syntax can be extended through adding event tag according to each application applied notification system, and enable event transmission between applications for interoperability of result in cooperative work. And building database of log data described in XML can be applied for analysis of the propensity to consume and the shopping route pattern in virtual shopping mall, analysis of study attitude of students in virtual class, the maintenance of a course of working in cooperative work, and so on.

3.2 Notifer

The designed Notifier consists of Converter and Filter. Converter changes events, which are created in external services of other VR environment, to XML. Filter reduces the amount of events by filtering.

If Notifier receives events from publisher, it doesn’ t transmit an event to subscribers at once. Events are transmitted to subscribers after these events pass through Converter and Filter. Figure 5 shows the operation of XML Converter, the registration of filtering and the applying point of Converter and Filter during processing of event notification.

If events are created in other VR environment, Converter must change events to XML-form to interact with the system of other VR environment. Filter performs events filtering though examining the events that are registered by subscribers, and then transmits the filtered events to subscribers. At this time, Notifier knows the propagation range of event through accessing to VR environment server that events were generated in advance.

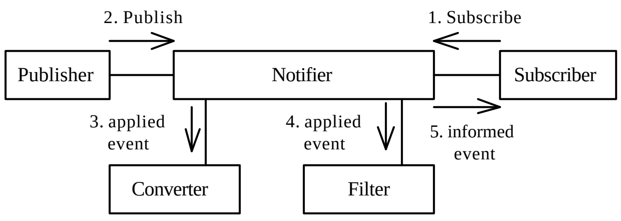


Figure 5. Event Notifier Collaboration Diagram

The registered Filter separates the default filter and the personalized filter. The default filter is applied by server (or controller) which manages application. The personalized filter can be registered by User Interface, which are supplied in each application.

The Default Filter performs the presentation event filtering generated in virtual environment about Action Event. It can apply working rules or policies about Information Event in the case of CSCW on virtual environment. Event filtering performs not only simple filtering by user request but also presentation event filtering about massive events for presentation in virtual environment. So event filtering has the benefit of reducing the amount of events for propagation in virtual environment.

The personalized filter is used to apply Human-readable event and user' s interest to event receipt. In general, the registration of behavior event is automatically filtering by system, and information event is applied for private filter based on the degree of participant' s interest.

4. Experimentation

We composed VR environment in which 50 participants interact with each other in 2 sessions, to evaluate usability of the proposed notification system. We apply 3 filters, presentation filter, access right filter, and receipt Information filter, to the composed VR environment.

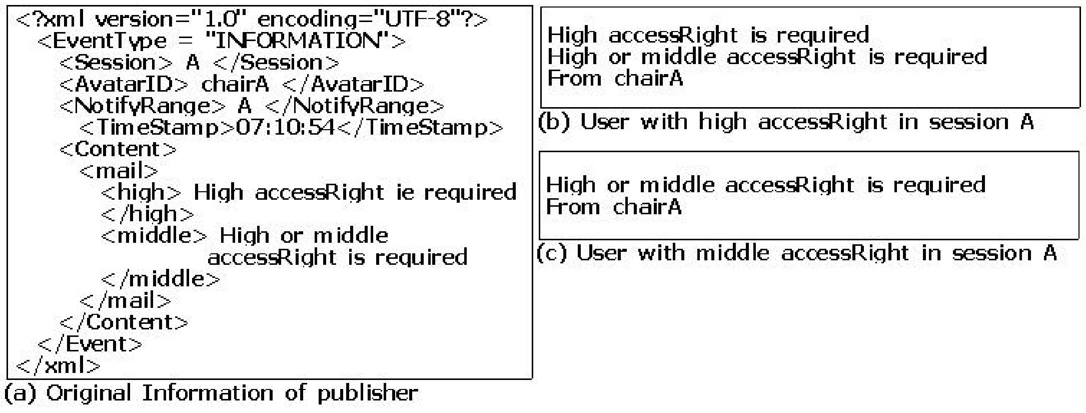
The presentation filter makes different the count of event propagation per a unit time according to distance between avatars. Notifier has a reference the value of distance between avatar from server. We assume the access right filter for access right in CSCW. The access rights of the participants in session are divided into high, middle, and low. These are applied for <tag> of XML document propagated. Notifier propagates XML document of information events propagated from server according to access rights through filtering. Finally, we applied receipt Information being able to receive events that specific avatar participating to different session generates. We assume that this filter is to support interaction with specific person in VR environment.

Table 1 shows environment applied in this experimentation.

Table 1. Parameters for VR configuration

parameter	value
size of CVE	500(m) * 500(m)
session	A,B
No. of avatars	50 per session
movement event frequency	3 (event/sec)
information event frequency	1(event/sec)
distance between avatars	A(max 30m), B(max 60m)

The access right filter propagates <high> and <middle> to “HIGH” subscriber participating to the same session, and only <middle> to “MIDDLE” subscriber about following Information events. And in the case of the participants having “LOW” rights, the filter discards the event. Figure 6(a) shows original event format, (b) and (c) is view showing according to each access right. Figure 7(a) shows the count of propagated events per each subscriber filtered by each filter among all events. Figure 7(b) shows the amount of event per each subscriber in session A filtered by each filter among all event propagation amounts.



```
<?xml version="1.0" encoding="UTF-8"?>
  <EventType = "INFORMATION">
    <Session> A </Session>
    <AvatarID> chairA </AvatarID>
    <NotifyRange> A </NotifyRange>
      <TimeStamp>07:10:54</TimeStamp>
    <Content>
      <mail>
        <high> High accessRight ie required
        </high>
        <middle> High or middle
                 accessRight is required
        </middle>
      </mail>
    </Content>
  </Event>
</xml>
```

(a) Original Information of publisher

```
High accessRight is required
High or middle accessRight is required
From chairA
```

(b) User with high accessRight in session A

```
High or middle accessRight is required
From chairA
```

(c) User with middle accessRight in session A

Figure 6. The operation result of access right filter

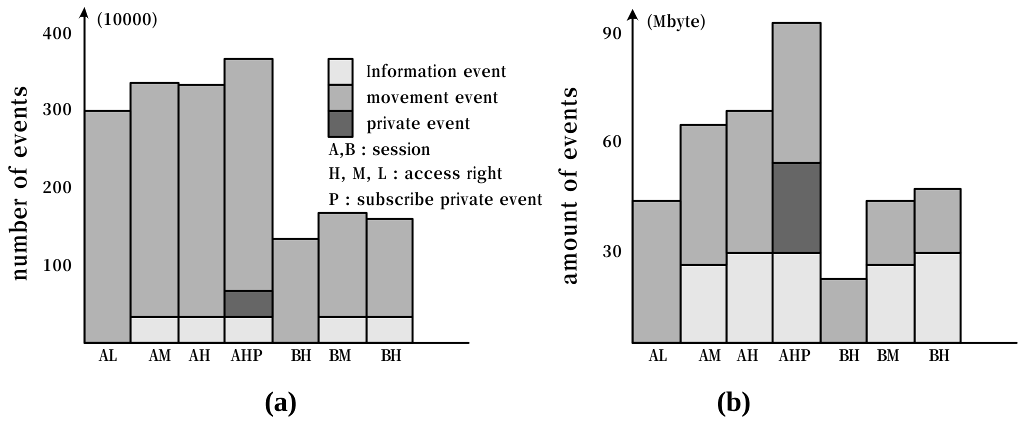


Figure 7. The result graph of event filtering

5. Conclusion

In this paper, we propose XML based event notification system in distributed virtual environments. The proposed system imports a notifier model of which event receivers can add and remove dynamically without affecting system. The system defines basic event types, and then those events express XML, which is standard of information transmission in the WWW. So, system can operate independently specific application on other environments. Therefore, it send events to receivers in accordance with registered information of participants in the filter after system convert events occurred in the every environment to XML using converter and filter. By this, we solve problem of event notification between different environments, and we also can decrease amount of transmitted events through filtering. In addition to, we increase effectiveness because system stores occurred event log data and operates filtering and retrieval of event using XML.

Reference

VRML(1997). The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1

Olof Hagsand (1996). Interactive multiuser Ves in the DIVE system, IEEE Multimedia Magazine, 3(1):30-39.

Funkhouser, Thomas(1995). A. RING: A Client-Server System for Multi-User Virtual Environments, Computer Graphics(1995 SIGGRAPH Symposium on Interactive 3D Graphics), pp 85-92.

Brandt & Kristensen (1997). “Web Push as an Internet Notification Service,” online White paper, http://keryxsoft.hpl.hp.com/doc/ins.html

Gupta et al. (1998). Event Notifier: A Pattern for Event Notification, in Java Report Magazine.

XML (1998). W3C Recommendation 10-Feburary-1998, eXtensible Markup Language(XML) 1.0, http://www.w3.org/TR/1998/REC-xml-1998210

Gamma et al. (1997). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley