공학 연구논문집 제30권 1호 pp. 325~336, 울산대학교.
Journal of Engineering Research Vol. 30, No. 1, pp. 325~336, 1999
University of Ulsan.

JavaRmi 분산객체를 위한 이벤트 서비스의 설계*

이명준 · 김현규 · 김규완
울산대학교 공과대학 컴퓨터정보통신공학부

<요 약>

대규모의 분산응용시스템은 일반적으로 복잡한 이벤트 전달과 함께 빈번한 상호작용을 필요로 하는 구성요소들로 이루어져 있다. 다중 이벤트 전달을 지원하는 상호작용 기법은 오류없이 구현하기 힘드므로 보다 안정적인 분산응용시스템을 효율적으로 지원하기 위해서는 이벤트의 다중처리를 체계적으로 처리하기 위한 서비스를 개발하는 것이 바람직하다.

본 논문에서는 Sun의 JavaRmi 분산객체 환경에서 다중 이벤트 전달을 체계적으로 지원하기 위한 JavaRmi 이벤트 서비스의 설계에 대하여 기술한다. 설계된 JavaRmi 이벤트 서비스는 체계적인 인터페이스와 병행성 제어를 제공하면서 이벤트 채널 팩토리와 이벤트 필터의 개념을 지원한다.

A Design of Event Service for JavaRmi Distributed Objects

Myung-Joon Lee · Hyun-Gyu Kim · Gyu-Wan Kim
School of Computer Engineering & Information Technology, University of Ulsan

A large distributed application generally consists of multiple components which require frequent interactions with complex event transfers. Since the interaction mechanism supporitng multiple event transfers is difficult to implement correctly, it is desirable to develop a service for the systematic multiple event transfers to support a stable distributed application in an effective manner.

In this paper, we describe a design of JavaRmi Event Service for systematic multiple event transfers on Suns JavaRmi distributed object environment. The JavaRmi Event

* 이 논문은 1998년 울산대학교의 연구비에 의하여 연구되었음.

Service includes the concept of event service factory and event filter with well-defined interfaces and concurrency control.

1. 서 론

90년대에 이르러 대두된 분산 환경은 네트워크로 연결된 이기종 시스템들간의 자원 공유를 가능하게 하여 정보 이용의 효율을 극대화시키며, 다양한 분산 환경 서비스의 투명성을 제공하고, 일부 시스템의 결함에 대처하여 높은 신뢰도를 제공할 수 있는 장점을 지니고 있다[1]. 그러나 분산 환경을 구성하는 각 시스템간의 이질성과 원거리 통신에 대한 세부적인 구현 사항은 분산 응용 프로그램을 구축하는 개발자들에게 상당한 어려움을 주게 되었다. 이를 극복하기 위하여 소개된 분산 객체 컴퓨팅(Distributed Object-Oriented Computing) 기법은 클라이언트/서버 기법과 객체지향 개념을 도입하여 분산 환경이 지닌 세부 구현 사항으로부터 독립적으로 시스템을 설계하기 위한 방법론으로서, 이 개념을 수용한 시스템의 대표적인 예로는 Sun사의 JavaRmi와 OMG의 CORBA를 들 수 있다.

Sun사의 JavaRmi는 Java언어를 기반으로 하여 대중성과 효율성의 측면에서 CORBA에 비해 우위를 가질 수 있으며, 이외에 기존의 다른 언어로 작성된 코드를 재사용하기 위한 JNI 기능이나 기존의 데이터베이스와 연결하기 위한 JDBC 기능 등 기존의 Java능력을 크게 향상시킴으로써 Java 객체 지향 프로그래밍을 위한 필수적인 도구로서 소개되고 있다. 그러나 JavaRmi에서는 JavaRmi 분산객체간에 이벤트의 복잡한 상호작용을 처리하기 위한 기능을 제공하지 않으며, JavaRmi를 이용하여 분산 응용 시스템을 보다 정교하고 안정성 있게 작성하기 위해서는 이러한 이벤트의 다중처리를 위한 기반 시스템이 제공되어야 할 것으로 예측된다.

이에 반해 분산 객체 컴퓨팅의 표준으로 OMG에 의해 제안된 CORBA에서는 이러한 메세지의 다중처리 문제를 체계적으로 지원하기 위해 이벤트 서비스(Event Service)[5,7]를 정의하고 있다. 그러나 CORBA 이벤트 서비스는 그룹 프로그래밍에 있어서 가장 기본적인 이벤트의 다중처리 기능에 대한 추상적인 명세만을 제공하고 있으므로, 이벤트 서비스를 사용하더라도 실제 분산 응용 시스템의 구현시에는 개발자의 별도의 노력을 필요로 하게 된다[6]. 이벤트 서비스가 분산 응용 시스템의 하부 구조로서 효과적으로 사용되기 위해 보완되어야 할 점은 소비자 측에서 선별적으로 이벤트를 구분할 수 있는 이벤트 필터(Event Filter) 기능[12], 다른 소비자와 상호작용을 위한 기능, 그룹별로 이벤트 서비스를 생성할 수 있는 Factory 기능[9], 오류가 발생했을 때 신뢰성 있는 복구를 위한 지속성(Persistency) 기능[1,8,9], 이벤트의 병행 처리와 그에 필요한 동기화 기능[12,13] 등을 들 수 있다.

따라서 본 연구에서는 CORBA 이벤트 서비스의 개념을 보완하여 JavaRmi[8] 분산객체를 위한 이벤트 서비스의 설계에 도입함으로써, 분산 응용 프로그램의 작성시에 그룹 프로그래밍을 위한 Java 개발자의 노력을 최소할 수 있도록 설계하였다. 이벤트 서비스의 기능 지원에 있어서 JavaRmi를 하부 구조(infrastructure)로 사용함으로써 가질 수 있는 잇점은 CORBA 이벤트 서비스에 비해 이벤트의 처리 과정이 효율적이며, 분산 환경의 세부 구현 사항에 독립적으로 구현할 수 있고, 또한 Java가 지니는 범용성으로 인해 대중화를 보다

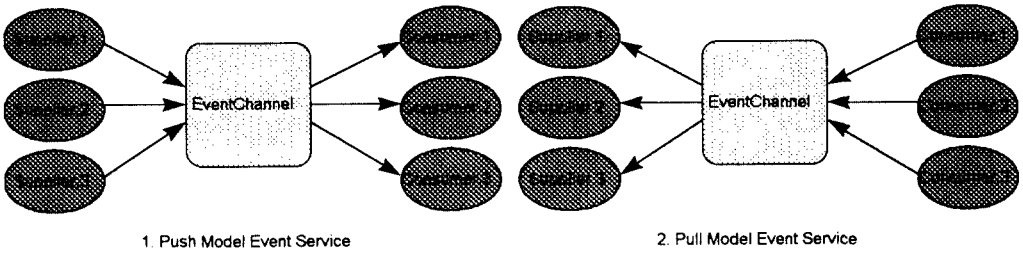
쉽게 기대할 수 있다는 것이다. 이러한 기본적인 잇점 이외에도 위에서 제시한 CORBA Event Service의 기능을 보완하여 직관적인 인터페이스, 이벤트 필터, 그룹별 이벤트 서비스의 생성과 관리 기능, 병행성과 동기화 도구 등에 관한 기능을 추가하여 Event Service의 유연성(flexibility)을 증가시킴으로써 그룹 프로그래밍을 위한 개발자의 별도의 노력을 줄일 수 있도록 설계한다.

2. 분산객체를 위한 이벤트 서비스의 바람직한 기능

이 장에서는 CORBA 이벤트 서비스의 개념에 대한 소개와 이러한 이벤트 서비스가 보다 효과적으로 활용되기 위해 CORBA에서 보완되어야 할 기능에 대해 언급한다. 이 장에서 제시한 이벤트 서비스의 기능들은 JavaRmi 분산객체를 위한 이벤트 서비스의 설계에 사용되며 이는 3장에서 다루게 된다.

2.1. CORBA 이벤트 서비스

CORBA 이벤트 서비스는 CORBA객체 간의 이벤트의 다중 전달 문제를 체계적으로 지원하기 위한 명세로서, CORBA 객체 간의 서비스를 확장하기 위한 명세서인 COSS(CORBA Object Service Specification)내에서 정의하고 있다. CORBA 이벤트 서비스는 이벤트를 생성하는 공급자(Supplier)와 이벤트를 처리하는 소비자(Consumer), 그리고 다수의 공급자와 다수의 소비자간의 중재자 역할을 수행해 주는 이벤트 채널(Event Channel)로 구성된다. CORBA 이벤트 서비스에서 공급자들은 소비자들에게 데이터를 보내주기 위해 이벤트 채널을 사용하며, 각 소비자는 이벤트 채널을 통해 공급자로부터 전달된 이벤트를 가져온다. 이벤트 데이터를 전달하고 가져올 때에는 공급자나 소비자가 서로에 대한 정보를 알 필요가 없으며, 이를 통해 CORBA 이벤트 서비스는 공급자와 소비자 간에 다중적이고 결합도가 낮은 통신 원리를 제공하게 된다.



[그림.1] 이벤트 서비스 모델

CORBA 이벤트 서비스는 이벤트의 전달 방법에 따라 Push 모델과 Pull 모델로 분류된다[그림.1]. Push 모델에서는 공급자가 능동적으로 소비자에게 이벤트를 전달하는 경우이며, Pull 모델에서는 소비자가 능동적으로 공급자에게 이벤트를 요청하는 경우이다. 또한

두 모델을 병행하여 사용할 수도 있다.

그러나 CORBA 이벤트 서비스는 이질적이고 다양한 분산 객체에 광범위하게 적용될 수 있도록 명세를 지나치게 추상화함으로써, 그룹 프로그래밍에 이벤트 서비스를 사용하더라도 별도의 이벤트 처리기법을 필요로 하게 되며, 사용에도 어려운 단점이 있다. 따라서 이벤트 서비스 시스템이 보다 다양한 영역에서 사용되기 위해서는 이러한 단점에 대한 기능이 보완되어야 할 것으로 예측된다.

2.2. 이벤트 서비스의 바람직한 기능

다음에서는 분산객체를 위한 이벤트 서비스가 보다 다양한 영역에서 효과적으로 적용되기 위해 제공되어야 할 기능에 대해 소개하고 있다.

가. 사용 절차

CORBA 이벤트 서비스는 자신이 공급자나 또는 소비자로서 EventChannel에 연결하기 위한 과정이 다소 복잡하다. 따라서 이벤트 서비스에서는 CORBA 이벤트 서비스의 대리자(proxy)에 연결하기 위한 복잡한 상호작용에 대한 구현 부분을 이벤트 서비스 내부적으로 처리하고 사용자들에게 보여지지 않도록 함으로써, 보다 직관적이고 사용하기 쉽도록 기능을 제공하는 것이 바람직하다.

나. 그룹별 이벤트 서비스의 생성과 관리

CORBA 이벤트 서비스에서는 그룹별로 이벤트 서비스를 생성하여 사용하기 위한 명세를 정의하지 않으므로 이벤트 서비스를 사용하더라도 그룹 프로그래밍을 위해서는 개발자가 별도로 구현을 해야 했다. 따라서 이벤트 서비스에서 이러한 문제를 해결하기 위해서는 이벤트 서비스내에 기본적으로 Factory 개념을 도입하여 그룹 단위의 이벤트 서비스 생성과 관리가 가능하도록 지원하는 것이 바람직하다.

다. 이벤트의 선별적인 수용(filter)과 상호작용(correlation)

CORBA 이벤트 서비스에서는 공급자가 전달한 이벤트가 이벤트 채널에 연결된 모든 소비자에게 전달되므로, 개발자가 보다 정교하게 이벤트를 다루기 위해서는 부가적인 구현을 필요로 하였다. 따라서 이벤트의 수용에 대한 정교한 제어를 지원하기 위해서는 일반적으로 이벤트 서비스에서 이벤트의 선별적인 수용과 이벤트의 수용에 대한 소비자간의 상호작용을 위한 방법을 제공하는 것이 바람직하다.

라. 이벤트의 병행 처리와 그에 따른 동기화 기능

일반적으로 분산환경에서 작동되는 응용시스템은 효율성을 위해 병행성(concurrency)과 그에 따른 동기화(synchronization) 기능을 지원하는 것이 바람직하다.

3. JavaRmi 이벤트 서비스의 설계

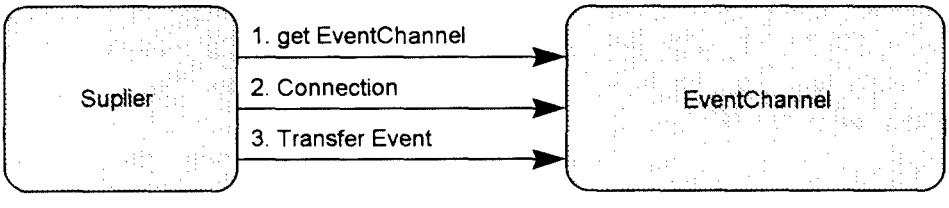
본 연구에서는 이벤트 서비스의 기능 지원에 있어서 Java언어에서 분산객체 개념을 지원하기 위한 JavaRmi를 하부 구조(infrastructure)로 사용함으로써, CORBA 이벤트 서비스에 비해 효율적인 이벤트의 처리 과정과, 분산 환경의 세부 구현 사항에 독립적인 구현, 또한 Java가 지니는 범용성으로 인한 쉬운 대중화 등에 대한 잇점을 충분히 활용하고자 하였다. 이 장에서는 JavaRmi를 하부구조로 사용함으로써 얻어지는 기본적인 잇점 이외에 이벤트 서비스에서 기능적으로 지원해야 하는 부분들에 대한 방법을 지적하고, 또한 이를 효과적으로 구현하기 위한 시스템의 구조와 관련된 인터페이스 설계를 제시한다.

3.1. JavaRmi 이벤트 서비스의 기능 설계

JavaRmi 이벤트 서비스는 위에서 지적한 CORBA 이벤트 서비스의 단점을 보완하여 이벤트 서비스의 유연성을 높임으로써 그룹 프로그래밍에 필요한 개발자의 노력을 최소화하고 또한 여러 분야에 광범위하게 적용될 수 있도록 설계하고자 한다. JavaRmi 이벤트 서비스는 다음과 같은 기능을 가지고 동작하도록 설계되었다.

가. 사용 절차

JavaRmi 이벤트 서비스에서는 CORBA 이벤트 서비스의 대리자(proxy)에 연결하기 위한 복잡한 상호작용에 대한 구현 부분을 이벤트 서비스 내부적으로 처리하고 사용자들에게 보여지지 않도록 함으로써, 보다 직관적이고 사용하기 쉽도록 하였다. 다음은 JavaRmi 이벤트 서비스에서 공급자로서 EventChannel에 연결하기 위해 필요한 절차를 기본적으로 다음과 같이 단순화한다[그림.2].



[그림.2] 공급자로서 JavaRmi 이벤트 서비스를 사용하기 위한 절차

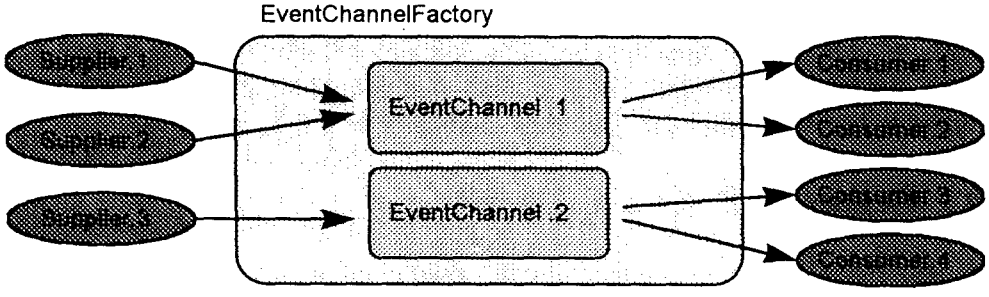
JavaRmi 이벤트 서비스의 Supplier에서는 EventChannel의 객체를 얻어와서(1) 단지 자신이 공급자로서 연결하는 것만으로 모든 연결 준비가 끝나도록 함으로써(2), 사용자의 관점에서 보다 쉽게 사용할 수 있도록 한다.

소비자의 역할을 위한 설계는 공급자의 경우와 반대가 되며, 부가적으로 공급자로부터 이벤트를 받았을 경우를 위한 push() 함수에 대한 구현을 개별적으로 포함하도록 한다.

나. 그룹별 이벤트 서비스의 생성과 관리

CORBA 이벤트 서비스에서는 그룹별로 이벤트 서비스를 생성하여 사용하기 위한 명세

를 정의하지 않으므로 이벤트 서비스를 사용하더라도 그룹 프로그래밍을 위해서는 개발자가 별도로 구현을 해야 했다. JavaRmi 이벤트 서비스는 이러한 문제를 위해 이벤트 서비스내에 기본적으로 Factory 개념을 도입함으로써 그룹 단위의 이벤트 서비스 생성이 가능하도록 한다[그림.3].



[그림.3] EventChannelFactory

EventChannelFactory객체는 그룹 단위로 이벤트 서비스를 생성하고 관리하기 위한 기능을 제공하는 객체로서, 새로운 EventChannel을 생성하거나 삭제하는 기능, 현재 등록된 EventChannel을 참조할 수 있는 기능 등을 제공하도록 한다.

다. 이벤트의 선별적인 수용(filter)과 상호작용(correlation)

CORBA 이벤트 서비스에서 공급자가 전달한 이벤트는 이벤트 채널에 연결된 모든 소비자에게 전달되며, 이벤트의 선별적인 수용을 위해서는 소비자 측에서 별도의 구현을 해야 했다. JavaRmi 이벤트 서비스는 이러한 문제를 위해 이벤트 서비스의 구현과 사용에 있어서 공유객체 개념을 지원함으로써 사용자가 이벤트를 보다 다양하고 쉬운 방법으로 처리할 수 있도록 하고자 한다. JavaRmi 이벤트 서비스에서 제공하는 공유객체 시스템은 본 연구실에서 97년도 한국학술진흥재단의 지원으로 연구된 IDL/SSO(IDL for Specifying Shared Object) 시스템[13,17,18]을 이용하여 구현될 예정이다.

IDL/SSO 시스템은 객체의 오퍼레이션이 스레드를 이용하여 상호 배제적이거나 병행적으로 수행될 수 있도록 하며, 그에 필요한 동기화 기능을 제공한다. 또한 객체의 상태에 따라 오퍼레이션의 수행을 지연시키거나 중단시키는 필터(filter) 기능 역시 제공하고 있다.

IDL/SSO에서 제공하는 오퍼레이션의 수행 형태는 function, procedure, guard, precondition, postcondition등이 있으며, IDL/SSO는 이벤트를 전달받는 소비자 객체 측에서 이벤트의 선별적인 수용 기능이나 상호작용 기능을 위해 사용될 수 있다. IDL/SSO 로 기술된 객체에서 오퍼레이션이 function으로 정의되면, 해당 오퍼레이션은 병행적으로 수행되며, procedure 오퍼레이션은 다른 오퍼레이션과 상호 배제적으로 수행된다. 그리고 guard 오퍼레이션은 인자로 전달된 boolean 오퍼레이션이 참이 될 때까지 수행을 지연시키며, precondition 오퍼레이션은 수행 시에 인자로 전달된 boolean 오퍼레이션이 참이 되지 않으면 수행을 중단시킨다. 마지막으로 postcondition 오퍼레이션은 수행을 마치더라도 인자로 전달된 boolean 오퍼레이션이 참이 되지 않으면 결과값을 반환하지 않는다.

function을 제외한 모든 오퍼레이션은 상호 배제적이다.

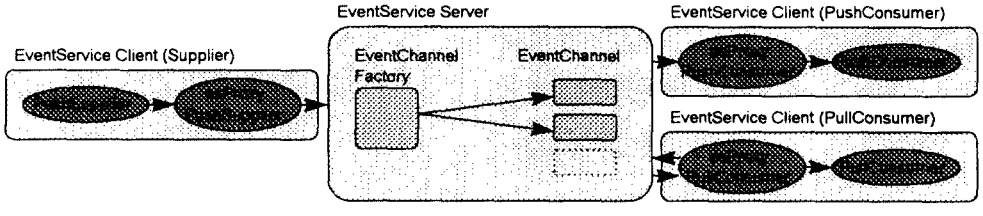
라. 이벤트의 병행 처리와 그에 따른 동기화 기능

병행성(concurrency)과 그에 따른 동기화(synchronization) 기능은 IDL/SSO 시스템을 사용하여 오퍼레이션 간의 동기화를 구현하거나, 또는 응용 프로그램 고유의 목적을 위해 IDL/SSO 시스템이 제공하는 세마포어(semaphore)나 조건 변수(condition variable)등의 도구를 이용하여 구현될 수 있다.

JavaRmi 이벤트 서비스 시스템은 내부적으로 IDL/SSO 시스템을 이용하여 구현함으로써 이벤트의 병행 처리나 동기화 기능의 구현에 대한 신뢰성을 체계적으로 지원하고자 한다.

3.2. JavaRmi 이벤트 서비스의 전체적인 구조도

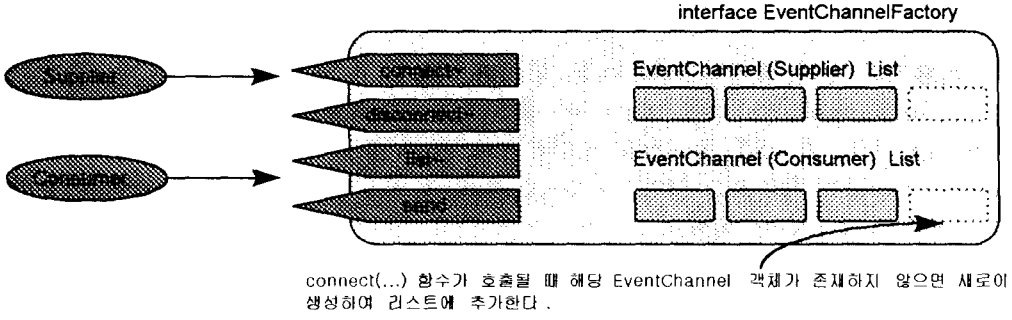
JavaRmi 이벤트 서비스는 그룹별로 이벤트 서비스를 제공하는 이벤트 서비스 서버와 이벤트를 공급하기 위한 공급자, 공급자로부터 전달된 이벤트를 사용하는 소비자로 구성되어 있으며, 각 공급자와 소비자는 이벤트 서비스와 보다 쉬운 인터페이스를 통해 연결하기 위해 대리자(Proxy)를 이용할 수 있도록 설계되었다. 이 절에서는 이벤트 서비스 서버의 구조에 대해 언급하며, 이벤트 서비스를 이용하는 소비자와 공급자에 대해서는 3.4절에서 다룬다.



[그림.4] JavaRmi 이벤트 서비스의 구조도

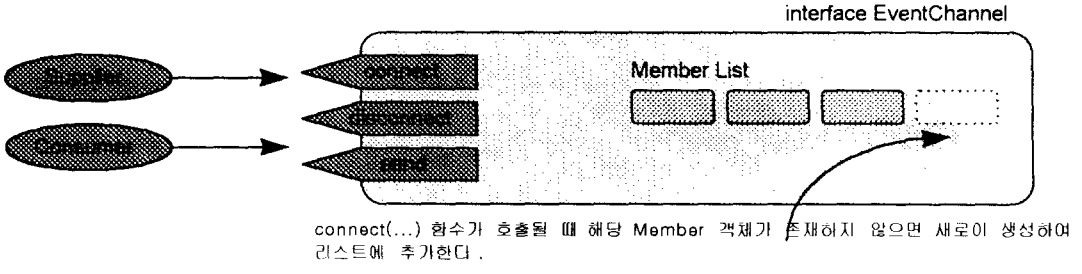
JavaRmi 이벤트 서비스 서버는 그룹 단위의 이벤트 서비스를 생성하기 위한 EventChannelFactory객체와 이벤트 서비스 고유의 기능인 소비자와 공급자 간의 이벤트의 다중전달 기능을 제공하는 EventChannel의 두 가지 객체로 구성된다.

EventChannelFactory 객체의 인터페이스에 대한 기본 설계는 [그림.5]와 같으며, EventChannelFactory 객체는 그룹 단위로 이벤트 서비스를 생성하고 관리하기 위한 기능을 제공하는 객체로서, 내부적으로 공급자와 소비자별로 EventChannel에 대한 리스트를 가지고 관리하게 되며, 새로운 EventChannel을 생성하거나 삭제하는 기능, 현재 등록된 EventChannel을 참조할 수 있는 기능, 특정 EventChannel의 소비자에게 이벤트를 보내기 위한 기능 등을 제공하게 된다.



[그림.5] "EventChannelFactory" 인터페이스

EventChannel 객체는 이벤트 서비스 고유의 기능인 소비자와 공급자 간의 이벤트의 다중전달 기능을 제공하는 객체로서 인터페이스에 대한 기본적인 설계는 다음 [그림.6]과 같으며, EventChannel객체에서는 사용자가 EventChannel에 접속하거나 해제하고, 또한 자신과 연결된 소비자에게 이벤트를 전달하기 위한 인터페이스를 정의하게 된다.



[그림.6] "EventChannel" 인터페이스

EventChannel객체는 connect() 오퍼레이션을 통해 연결된 멤버에 대한 리스트를 유지하며, send()를 통해 전달된 메세지는 연결된 모든 멤버에게 전달된다. 그리고 각 멤버에서는 자신에게 전달된 메세지를 다시 소비자에게 보냄으로써, 이벤트의 다중전달 기능을 구현하게 된다.

3.3. JavaRmi 이벤트 서비스 인터페이스(Interface)의 정의

JavaRmi 이벤트 서비스 인터페이스에는 이벤트 서비스 클라이언트로서 연결하거나 연결해제를 위한 오퍼레이션과 이벤트를 보내기 위한 오퍼레이션을 제공하게 된다. 다음은 JavaRmi 이벤트 서비스가 제공하는 오퍼레이션을 소개하고 있다[그림.7].

```
void connectSupplier(String group, String user);
void connectConsumer(String group, String user, Object cbobj);
void disconnectSupplier(String group, String user);
void disconnectConsumer(String group, String user);
void send(String group, String user, String msg);
```

[그림.7] JavaRmi 이벤트 서비스 인터페이스

connectSupplier()는 이벤트를 전달하기 위한 공급자로서 연결되기 위한 오퍼레이션으로서 해당 공급자가 속하는 그룹 이름과 사용자 이름을 전달한다. 이와 유사한 오퍼레이션으로서 ConnectConsumer()는 공급자로부터 전달받은 이벤트를 사용하기 위한 소비자로 연결되기 위한 오퍼레이션이며, connectSupplier()와는 달리 이벤트 서버에서 소비자로 이벤트를 역으로 전달하기 위해 필요한 객체(Callback Object)를 인자로 필요로 한다. 그리고 이들의 연결을 해제하기 위한 오퍼레이션 역시 제공한다. 마지막으로 공급자에서 메세지를 전달하기 위한 send() 오퍼레이션이 정의되어 있으며, 전달하는 공급자의 그룹과 사용자 정보와 전달할 메세지 객체를 인자로 사용한다.

위의 인터페이스는 IDL/SSO 시스템을 이용하여 정의함으로써 기본적으로 병행성과 그에 따른 동기화 문제를 지원할 수 있도록 한다.

3.4. JavaRmi 이벤트 서비스의 클라이언트측 사용예

3.4.1. 공급자(Supplier)의 사용예

JavaRmi 이벤트 서비스에서 공급자는 공급자측의 대리자(Proxy)를 이용해 [그림.8]과 같이 간단한 방법으로 이벤트 서비스에 연결하고 메세지를 보낼 수 있다. 대리자와 연결시에는 이벤트 서비스 서버의 IP 어드레스와 서버 이름, 자신이 연결할 그룹 이름과 사용자 이름을 인자로 전달한다. 연결이 된 후에 공급자는 send() 오퍼레이션을 통하여 메세지를 전달하게 된다.

```
public static void main(String argv[]) {
        esProxyPushSupplier ec = new esProxyPushSupplier(203.205.83.150,
EventServer, Animal; Dog);
es.send(Bow Wow !!);
    }
```

[그림.8] JavaRmi 이벤트 서비스의 공급자의 사용예

공급자측의 대리자는 전달된 인자로부터 내부적으로 서버에 대한 정보를 항상 유지하고 있으며, send() 오퍼레이션이 호출될 때마다 이러한 정보를 이용하여 이벤트 서비스 서버에게 이벤트를 전달하게 된다.

3.4.2. 소비자(Consumer)의 사용예

JavaRmi 이벤트 서비스에서 소비자는 공급자와 마찬가지로 소비자측의 대리자(Proxy)를 이용해 [그림.9]와 같이 간단한 방법으로 이벤트 서비스에 연결하고 공급자로부터 다중 전달된 메세지를 받을 수 있다. JavaRmi 이벤트 서비스에서 소비자는 공급자와 달리 Push 모델과 Pull모델을 지원하며, 소비자가 Push모델의 대리자를 사용할 경우에는 IP 어드레스, 서버, 그룹, 사용자 이름 이외에, 자신이 이벤트 서비스 서버로부터 메세지를 되돌려받기 위한 자신의 this 포인터를 추가하여 인자로 전달한다. 연결된 후에 소비자는 receive() 오퍼레이션을 통하여 메세지를 전달받게 되며, JavaRmi 이벤트 서비스의 소비자는 반드시 receive() 오퍼레이션이 호출되었을 때에 대한 구현을 포함하도록 한다.

```
public static void main(String argv[]) {
        esProxyPushConsumer ec = new esProxyPushConsumer(203.205.83.150,
EventServer, Animal; Dog, this);
...;
    }
public void receive(Object data) { ... }
```

[그림.9] JavaRmi 이벤트 서비스의 Push모델 소비자의 사용예

그리고 JavaRmi 이벤트 서비스에서 소비자가 Pull모델의 대리자를 사용할 경우에는 별도의 this 포인터를 추가하여 인자로 전달하지 않으며, 소비자가 능동적으로 전달된 이벤트를 받아오기 위해 대리자에서 receive(), start(), stop() 등의 오퍼레이션을 제공한다. receive() 오퍼레이션은 전달된 메세지 중 하나의 메세지만 가져오며, 계속적으로 메세지의 전달을 체크하기 위해 start()와 stop()오퍼레이션이 사용될 수 있다. 연결된 후에 Pull모델의 소비자는 Push모델과 마찬가지로 receive() 오퍼레이션을 통하여 메세지를 전달받게 되며, JavaRmi 이벤트 서비스의 소비자는 반드시 receive() 오퍼레이션이 호출되었을 때에 대한 구현을 포함하도록 한다.

```
public static void main(String argv[]) {
        esProxyPullConsumer ec = new esProxyPullConsumer(203.205.83.150,
EventServer, Animal; Dog);
ec.receive();
ec.start();
ec.stop();
    }
public void receive(Object data) { ... }
```

[그림.10] JavaRmi 이벤트 서비스의 Pull모델 소비자의 사용예

소비자측의 대리자는 공급자와 마찬가지로 전달된 인자로부터 내부적으로 서버에 대한 정보를 항상 유지하고 있게 된다.

4. 결 론

본 논문에서는 CORBA 이벤트 서비스 개념을 발전시켜 JavaRmi 분산객체를 위한 이벤트 서비스의 개발에 도입함으로써, Java 분산객체 응용 프로그램의 작성시에 이벤트의 다중 처리를 위한 Java 개발자의 노력을 최소화하는 이벤트 서비스의 설계를 제안하였다. 설계된 JavaRmi 이벤트 서비스는 JavaRmi를 하부 구조로 사용함으로써 CORBA 이벤트 서비스에 비해 이벤트의 처리 과정에 있어서 효율적이며, 분산 환경의 세부 구현 사항에 독립적으로 구현할 수 있고, 또한 Java가 지니는 범용성으로 인해 쉬운 대중화를 기대할 수 있다. JavaRmi 이벤트 서비스에서는 쉬운 인터페이스와 Factory개념을 통한 그룹별 이벤트 서비스의 생성과 관리 기능, 이벤트의 선별적인 수용과 상호작용 기능, 이벤트의 병행 처리 기능과 그에 필요한 동기화 도구를 지원하여 그룹 프로그래밍을 위한 개발자의 별도의 노력을 줄일 수 있도록 하였다.

설계된 JavaRmi 이벤트 서비스는 현재 구현중에 있으며, 정보공유를 목적으로 한 그룹 분야의 대부분의 분산 응용 시스템 작성에 적용되어 Java를 기반으로 한 사용자들의 호응을 얻을 수 있을 것으로 기대된다. JavaRmi 이벤트 서비스 시스템은 시스템 관리 도구, 특성 리스트 객체, CASE(Computer Aided Software Engineering) 도구, 스프레드 시트, 주식 시세 통보 시스템 등의 주로 이벤트의 공급자와 소비자 간의 느슨한 연결 관계를 지닌 그룹 소프트웨어 분야에 광범위하게 적용될 수 있다.

현재 Java 언어와 같은 순수 범용 언어 수준에서 직접적으로 다중 이벤트 처리를 지원하는 연구나 상품은 전무한 상황이므로, JavaRmi 이벤트 서비스에 대한 연구는 추후의 다른 JavaRmi 분산객체를 위한 이벤트 서비스에 대한 연구에도 도움을 줄 수 있을 것으로 예측된다.

참고문헌

[1] "The Java Language Tutorial", Sun Microsystems, Inc., 1995

[2] Andreas Vogel, Keith Dubby, "Java Programming with CORBA", John Wiley & Sons, Inc., 1996

[3] F. Ranno, S.K. Shrivastava and S.M.Wheater, "A System for specifying and Coordinating the Execution of Reliable Distributed Applications" (DAIS'97), Cottbus, Germany, September 30 - October 2, 1997

[4] Douglas C. Schmidt, Steve Vinoski, "Distributed Callbacks and Decoupled Communication in CORBA (Column 8)", SIGS C++ Report Magazine, February 1997

[5] Douglas C. Schmidt, Steve Vinoski, "Object Interconnections: The OMG Events

Service (Column 9)", SIGS C++ Report Magazine, February 1997

[6] Douglas C. Schmidt, Steve Vinoski, "Overcoming Drawbacks in the OMG Event Service (Column 10)", SIGS C++ Report Magazine, February 1997

[7] Object Management Group, "CORBAservices: Common Object Services Specification", OMG 95-3-31, 1995

[8] "JavaRmi: Java Remote Method Invocation", Sun Microsystems, Inc., 1996

[9] Object Management Group, "Common Object Request Broker: Architecture and Specification", 1995

[10] IONA "OrbixTalk White Paper: The CORBAservice Event Service", IONA Technologies Ltd., 1996

[11] Gerald Brose, "JacORB-A Java Object Request Broker", Technical Report B 97-2, Frieie University, April 1997

[12] L.Bermans, M.Aksit & J.Bosch, Composition-Filters: Extended Expressiveness for OOPLS, OOPSLA, 92 Workshop on Object-Oriented Programming Languages: The Next Generation, October 8, 1992.

[13] 박양수, 김현규, 이명준, 한상영, A Specificaiton Language System for Shared Object under CORBA Open Distributed Computing Environment, 한국정보처리학회 논문지 제5권 2호 pp.404-414, May 1998.

[14] 정혜영, 김현규, 박양수, 구자록, 이명준, 자바를 이용한 CORBA 이벤트 서비스의 개발, 한국정보과학회 가을 학술발표논문집 Vol.24, No.2 pp.237-240, October 1997.

[15] "JCE: The Java Collaboration Environment", Sun Microsystems, Inc., 1997

[16] 김현규, 최혁재, 김영곤, 박양수, 이명준, CORBA 개방형 분산환경을 위한 공유객체 명세언어의 설계, 한국정보과학회 봄 학술발표논문집, April 1997.

[17] 김현규, 이동헌, 박양수, 이명준, CORBA 개방형 분산환경을 위한 Java 공유객체의 지원, 한국정보과학회 가을 학술발표논문집, Vol.24, No.2 pp.241-244, October 1997.

[18] 정혜영, 문남두, 김현규, 박양수, 이명준, CORBA 유형 이벤트 서비스의 지원, 한국정보과학회 봄 학술발표논문집, Vol.25, No.1 pp.208-210, April 1998.