분산 객체 통합을 위한 CORBA 이벤트 서비스 기법

The CORBA Event Service Mechanism for Distributed Object Integration

이 재 완*
Jae-Wan Lee

요 약

CORBA는 특정 플랫폼이나 기술에 관계없이 객체간에 상호작용을 가능하게 하는 미들웨어이다. CORBA 통신 모델은 분산 객체를 클라이언트와 서버간에 동기적으로 연결되기 때문에 대기 시간이 필요하며, 멀티캐스트를 지원하지 않는다. 이러한 점을 해결하기 위해, OMG에서는 다중 응용 객체간의 멀티캐스트를 지원할 수 있도록 CORBA 이벤트 서비스(Event Service)를 제안하였다.

본 논문에서는 이벤트 채널에 등록된 각각의 소비자와 공급자에 두 개의 인터페이스 객체를 두어 양방향 통신을 지원하며, 신뢰성을 향상시키는 기법을 제안한다. 또한, 효율적으로 객체를 통합하기 위해 분산된 채널들을 그룹화하고 뷰로 관리한다. 채널그룹 중에 하나를 조정자로 선택하여 그룹과 뷰를 관리하도록 한다.

Abstract

CORBA is a middleware which enables distributed objects to cooperate regardless of specific platforms and techniques. But the ordinary CORBA communication model does not support multi-cast, and needs delay time, because it synchronously connects the distributed objects between client and server. To solve these problems, OMG suggests CORBA Event Service which can provide multi-cast among application objects. This paper presents a new technique for improving reliability, and supporting two-way communication by laying two interface objects on each consumers and suppliers that are registered in event channel. Also, to integrate objects efficiently, we, group distributed event channels and management it as view. A coordinator selected from channel group controls group and view.

1. 서 론

최근 정보통신망 및 미들웨어(Middleware)의 발달로 독자적으로 사용되던 자원은 분산환경에서 시스템간에 공유하면서 활용하고 있다. 그러나 정보통신 시대의 최대 목표라 할 수 있는 분산환경에서의 통합 시스템(System Integration)을 구축하여 단일 시스템처럼 편리하게 사용할 수 있는 단일 시스템 뷰(Single System View)의 구축은 통신망, 하드웨어, 소프트웨어 및 사용하는 프로그래밍 언어 등의 이질성으로 인해 많은 어려움을 겪고 있다. 정보의 효율적인 활용을 위해서는 이종의 정보자원을 효과적으로 통합해야 하며, 이러한 문제점을 해결하기 위해 서로 다른 시스템 환경에서 상호 동작할 수 있는 미들웨어가 필요하다. 이러한 문제점을 해결하기 위해 OMG(Object Management Group)에서는 객체지향 기법에 근거를 두어 특정 플랫폼(Platform)이나 기술에 관계없이 분산되고 이질적인 시스템간에 상호 운용할 수 있는 CORBA(Common Object Request Broker Architecture)를 표준으로 제안하였다[1,2].

분산 객체간에 연결을 위한 CORBA 통신 모델은 하나의 클라이언트와 서버가 동기적으로 연결되는 형태로서, 비동기적이나 다중 응용 프로그램간의 멀티 캐스팅(Multicasting) 등의 기능을 지원하지 않는다. 따라서 OMG(Object Management Group)에서는 다중 응용 프로그램간의 비동기적인 통신을 지원할 수 있도록 CORBA 이벤트 서

* 종신회원 : 군산대학교 전자정보공학부 교수
jwlee@kunsan.ac.kr

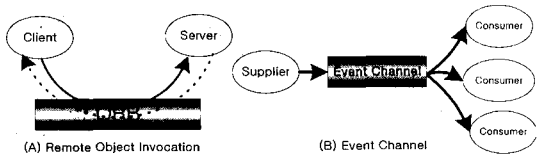
비스(Event Service)를 제안하였다[4,8]. 이벤트 서비스는 CORBA 객체 서비스 명세(COSS : Common Object Service Specification)에 정의된 서비스들 중 하나로서 분산 객체들간에 비동기적이고 결합도가 낮은 통신을 위한 모델을 제공한다[3,4,5,6,7]. 소비자와 공급자를 연결하는 채널의 내부적인 구조는 현재 개방된 상태로서 어플리케이션 개발자에 의해 구성하도록 하였다[12].

본 논문에서는 분산 객체들간에 CORBA를 기반으로 한 이벤트 채널을 사용하여 소비자와 공급자들 사이에 결합도를 완화시키고, 기존의 이벤트 채널 내부에 클라이언트와 서버가 메시지 전송에 관여하지 않고, 메시지 전송을 담당하는 인터페이스 객체 및 정보를 관리하는 정보 관리자 객체를 두어 메시지를 전송할 수 있게 하였다. 메시지를 전송하는 객체로서, 소비자 측에는 서비스 요구객체와 서비스 수신객체, 공급자 측에서는 서비스 요구 수신객체와 서비스 응답객체로 분리하여 구성함으로써 공급자와 소비자간의 신뢰성 및 양방향 메시지 통신을 제공하는 이벤트 서비스 기법을 개발하였다.

분산된 채널을 통합, 관리하기 위해 분산된 채널들을 하나의 채널그룹으로 구성하고, 채널그룹에 등록하는 채널에 대한 정보를 관리하기 위해, 채널그룹에 등록된 채널 중에 하나의 채널을 조정자 채널로 선정하여, 채널그룹에 등록하는 채널에 대한 뷰를 관리함으로써 채널그룹의 모든 채널들이 모든 시간에서 동일한 뷰를 가질 수 있도록 하였다.

2. CORBA 이벤트 서비스

CORBA는 OMG에 의해서 정의된 분산 객체 컴퓨팅 미들웨어이며, 특정한 플랫폼과 기술에 상관없이 객체들을 통합 운용할 수 있는 기본구조를 제공한다. 그림 1과 같이 CORBA에서는 다른 두 가지 형태의 통신을 제공한다[4,5,6,7]. 중요한 구성요소는 ORB(Object Request Broker)이며, 주요



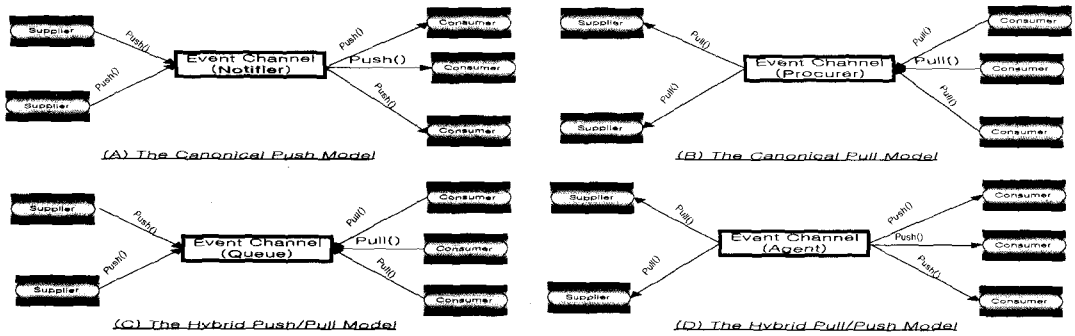
(그림 1) Remote Object Invocation 과 Event Channel 통신 모델

기능은 투명성 있게 메시지를 객체들간에 전달하는데 있다. 원격 객체 호출(Remote Object Invocation)은 ORB를 통해 원격 객체 메소드를 호출한다(그림 1(A)). 이벤트 채널은 CORBA 이벤트 서비스의 일부이며 많은 공급자와 생산자들이 연결된 버퍼 이벤트 채널이다(그림 1(B)). 소비자와 공급자는 채널에 메시지를 전송하고 자신들의 작업을 계속 수행하며, 메시지들은 채널에 등록된 소비자들에게 분배된다. 소비자의 관점에서 채널은 메시지들을 생산하는 단일 생산 객체로서 행동하게 된다.

이벤트 서비스는 비동기적인 메시지나 이벤트들을 보내고 받는 결합도가 낮은 객체들을 대상으로 제공하며, 일반적인 CORBA 모델은 하나의 소비자와 생산자가 동기적(Synchronous)으로 연결되는 형태인 반면에, CORBA 이벤트 서비스는 다중 응용 프로그램간에 비동기적인(Asynchronous) 통신을 지원한다[5,10,11].

공급자와 소비자 사이에 사건을 초기화하는 방법으로 Pull 모델과 Push 모델을 지원한다. Push 모델은 사건들의 공급자가 소비자에게 사건 데이터의 전송을 초기화시키고, Pull 모델은 사건들의 소비자가 공급자에게 사건 데이터를 요구할 수 있게 한다. OMG 이벤트 서비스의 기본구조는 그림 2와 같이 구성요소 간의 4가지 일반 모델을 정의하고 있다 [3,4,6,7].

- 표준 Push 모델 : 표준 Push 모델(그림 2(A))은 이벤트의 공급자가 소비자들에게 이벤트 데이터 전송을 초기화한다. 이벤트 채널은 통보자(Notifier)의 역할을 한다. 능동적인 공급자는 이벤트 채널에 등록된 수동적인 소



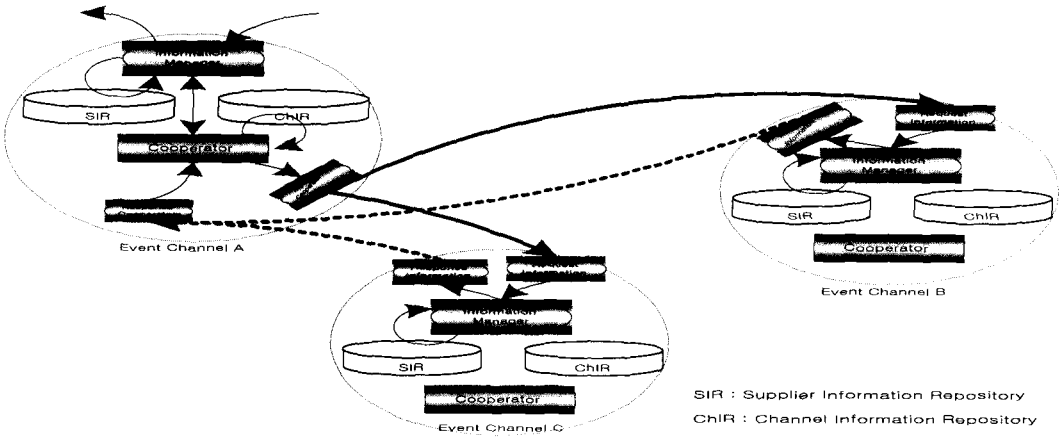
(그림 2) 이벤트 서비스 통신모델

비자들에게 데이터를 보내기 위해 이벤트 채널을 사용한다(push).

- 표준 Pull 모델 : 표준 Pull 모델(그림 2(B))은 소비자가 공급자로부터 사건들을 요구할 수 있다. 이벤트 채널이 소비자 대신에 이벤트들을 획득하기 때문에, 이벤트 채널은 획득자(Procurer)의 역할을 한다. 따라서, 능동적인 소비자는 이벤트 채널을 경유하여 수동적인 공급자로부터 데이터를 받을 수 있다(pull).
- 혼합형 Push/Pull 모델 : 혼합형 Push/Pull 모델(그림 2(C))은 소비자가 공급자에 의해 이벤트 채널에 저장된 이벤트를 요구하는 혼합형이다. 이벤트 채널은 큐(Queue)의 역할을 한다. 따라서, 능동적인 소비자는 이벤트 채널을 경유하여 공급자에 의해 위탁된 데이터를 받을 수 있다.
- 혼합형 Pull/Push 모델 : 혼합형 Pull/Push 모델(그림 2(D))은 이벤트 채널이 공급자로부터 이벤트를 받고(Pull) 소비자에게 이벤트를 보낸다(Push). 이벤트 채널은 대리자(intelligent agent)의 역할을 한다. 따라서, 능동적인 이벤트 채널은 수동적인 공급자로부터 데이터를 받고(pull), 수동적인 소비자에게 데이터를 보낸다(push).

3. 시스템 구조

3.1. 이벤트 채널간의 연결



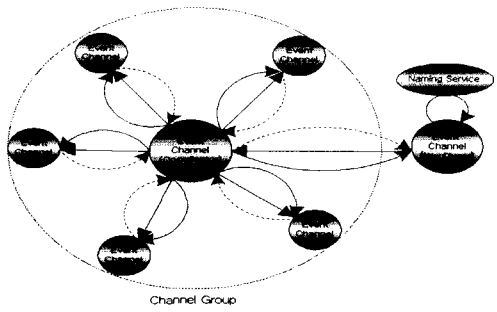
(그림 3) 분산된 채널간의 연결구조

분산 환경에서 본 시스템의 이벤트 채널간의 연결구조는 그림 3과 같다. 공급자들의 서비스 정보 및 채널에 대한 정보를 저장·관리하는 정보 관리자 객체(Information Manager)와 채널간에 서비스 정보를 공유하는 협력자 객체(Cooperator)를 두어 정보 관리자 객체 간에 상호연동을 통해 연결한다. 소비자로부터 서비스가 요청되면 서비스 요구를 전달하는 인터페이스 객체는 이벤트 채널에 연결된 정보 관리자 객체를 통해 지역 채널에 공급자들 중 서비스를 제공할 수 있는 공급자를 우선적으로 검색하여 공급자와 연결하고, 서비스를 제공할 수 있는 공급자의 정보가 존재하지 않는 경우 정보 관리자 객체와 협력자 객체를 통해 원격지의 정보 관리자 객체들 간의 상호연동하여 원격지 공급자와 연결하게 된다.

3.2 이벤트 채널 그룹 등록 및 뷰 관리

지역 채널에 연결된 공급자뿐만 아니라 원격지 채널에 존재하는 공급자와 연결하여 다양한 서비스를 제공하기 위해 분산된 이벤트 채널간의 연결을 위한 방법으로 정보 관리자 객체와 협력자 객체를 통해 연결하는 방법을 사용한다. 또한 분산된 모든 채널에 대해 신뢰성 있는 멀티캐스트를 지원할 수 있도록 채널 그룹에 등록한 채널에 대한 뷰(View)를 관리한다. 채널 간의 뷰를 관리하는 방법으로 각 채널 그룹 중에 하나의 채널을 조정자 채널(Coordinator Channel)로 선정하여 채널그룹의 뷰를 관리하도록 하였다. 채널 그룹간에 조정자 채널에 대한 정보 또한 조정자 채널 간에 상호 연동을 통해 보장할 수 있도록 하였다.

신규 이벤트 채널은 조정자 채널을 통하여 채널 그룹에 등록하며, 채널 그룹에 등록된 분산된 이벤트 채널에 채널 자신의 정보를 알림으로써 채널 간에 상호 연동을 제공할 수 있다. 모든 시간에서 이벤트 채널은 자신의 채널 그룹에 속해 있는 다른 채널들의 뷰(View)를 가진다. 뷰는 채널 객체 그룹 멤버를 나타내는 채널 객체 레퍼런스의 동적인 리스트로서, 그룹 멤버의 변경은 각 그룹 멤버에서 뷰의 변경을 자동적으로 발생시킨다. 일관성 있는 채널 그룹 뷰는 채널 그룹에 신뢰성 있는 멀티캐스트를 수행하는 기법을 제공한다.

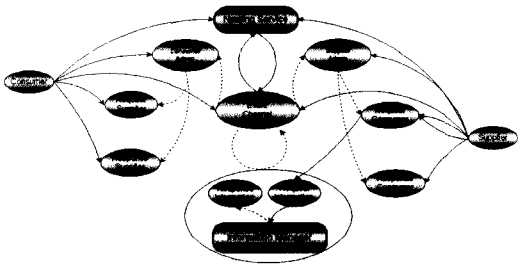


(그림 4) 이벤트 채널간의 뷰 관리

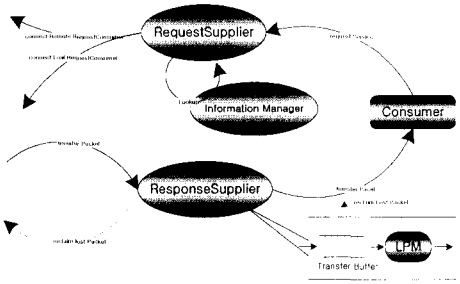
신규 채널이 채널 그룹에 등록하기 위해서는 다음과 같은 단계를 거쳐 채널 그룹에 등록한다. 신규 채널은 이름 서비스(Naming Service)로부터 조정자 채널의 레퍼런스를 구한다. 신규 채널은 이름 서비스로부터 반환 받은 레퍼런스를 참고하여 채널의 이름 및 레퍼런스를 포함한 등록 정보를 조정자 채널에 등록한다. 조정자 채널은 자신의 정보 관리자 객체의 협력자 객체를 통해 채널 그룹에 등록된 모든 채널 멤버에 새로운 채널이 추가 될 것이라는 메시지를 멀티캐스트 한다. 조정자 채널로부터 추가 메시지를 받은 멤버들의 정보 관리자 객체는 조정자 채널에 추가할 준비가 되었음을 알린다. 조정자 채널의 협력자 객체는 채널 그룹에 등록된 모든 채널 멤버의 이름 및 레퍼런스를 포함한 정보를 새로 등록한 신규 채널에 전송한다. 조정자 채널의 협력자 객체는 신규 채널에 대한 정보를 채널 그룹의 모든 정보 관리자 객체에 전송한다.

3.3 이벤트 채널의 서비스 구조

본 시스템에서 이벤트 채널 구조는 그림 5와 같이 서비스를 요청하는 소비자와 서비스를 제공하는 공급자, 소비자와 공급자 사이에 이벤트들을



(그림 5) 이벤트 채널에서 연결 구조



(그림 6) 소비자측의 프록시 객체

전달하기 위한 이벤트 채널, 채널에 등록된 공급자에 대한 정보 및 분산된 채널 정보를 관리하는 정보 관리자 객체로 구성된다.

채널을 구성하는 내부 객체로는 소비자와 공급자를 대신하여 서비스 요구 및 응답을 제공 해주는 프록시 객체와 이런 프록시 객체를 생성하는 공급자 관리자 객체(SupplierAdmin)와 소비자 관리자 객체(ConsumerAdmin)로 구성된다. 소비자를 대신하여 공급자에게 서비스를 요구하기 위한 서비스 요구 프록시 객체(RequestSupplier), 공급자로부터 반환된 서비스를 수신하는 서비스 수신 프록시 객체(ResponseSupplier)로 구성되며, 공급자측에서 소비자로부터 서비스 요구를 수신하는 서비스 요구 수신 프록시 객체(RequestConsumer)와 소비자에게 서비스를 제공하는 서비스 응답 프록시 객체(ResponseConsumer)로 구성된다.

또한, 지역 채널에 등록한 공급자에 대한 정보를 저장·관리하는 정보관리자 객체를 두어 소비자가 서비스를 요청할 때, 지역 채널에 등록된 공급자를 우선적으로 검색하여 서비스 공급자와 연결함을 목적으로 하고 있다. 서비스를 요구하는 객체와 서비스를 수신하는 객체를 분리해서 구성함으로서 공급자와 소비자 간에 동시에 양방향 통신을 지원할 수 있도록 설계하였다.

3.4 이벤트 채널의 객체 구성

3.4.1 소비자 측의 인터페이스 객체

소비자를 대신하여 서비스를 제공하는 소비자 측의 프록시 객체는 공급자에 서비스를 요청하는 서비스 요구 프록시 객체(RequestSuplier)와 공급자로부터 서비스를 수신하는 서비스 수신 프록시 객체(ResponseSupplier)로 구성된다. 서비스 요구 프록시 객체는 정보관리 객체와 연동 하여 소비자의 서비스 요구를 제공할 수 있는 공급자의 서비스 요구 수신 프록시 객체와 연결하는 객체이다. 서비스 수신 프록시 객체는 공급자로부터 전송되는 서비스를 수신하는 프록시 객체로서, 공급자로부터 패킷단위로 전송된 패킷을 검사하는 지역 패킷 관리자(Local Packet Manager : LPM) 객체와 전송중인 패킷을 보관하는 기능 및 네트웍의 에러나 프록시 객체의 에러로 인해 패킷 재요청 시, 재전송 요청 요구 동안 전송되는 패킷을 저장하는 전송 버퍼(Transfer Buffer)로 구성된다.

```
Moduel MibCossEventChannel {
    interface RequestSupplier:CosEventChannel::ProxyPushSupplier{
            /* . . . . . */
    };
    interface ResponseSupplier:CosventChannel::ProxyPushSupplier{
            /* . . . . . */
    };
    interface ConsumerAdmin{
        RequestSupplier  obtain_request_supplier();
        ResponseSupplier obtain_request_supplier();
        RID register_proxy(/* . . . */);
        boolean register_Inforef(/* . . . */);
        boolean withdraw_Proxy(/* . . . */);
    };
    interface MibEventChannel{
         ConsumerAdmin for_consumer();
         void destroy();
    };
};
```

(그림 7) 소비자측 인터페이스

소비자를 채널에 연결하기 위한 소비자 측의 주요한 인터페이스와 오퍼레이션은 그림 7과 같다.

3.4.2 공급자 측의 인터페이스 객체

공급자 측의 프록시 객체는 소비자로부터 요청된 서비스 요구를 수신하는 서비스 요구 수신 프록시 객체(RequstConsumer)와 공급자로부터 처리된 결과를 소비자에게 전송해 주는 서비스 응답 프록시 객체(ResponseConsumer)로 구성된다. 서비스 요구 수신 프록시 객체는 소비자 측의 프록시 객체로부터 전송된 서비스 요구를 서비스 응답 프록시 객체의 소비자 정보 관리자 객체(Consumer Information Manager : CIM)에 소비자에 대한 정보를 등록하는 기능 및 공급자에 소비자의 서비스 정보를 전송하는 프록시 객체이다. 서비스 응답 프록시 객체는 공급자로부터 전송되는 데이터를 소비자에게 전송하는 기능을 하는 프록시 객체로서, 소비자들의 서비스 요구 정보를 보관하는 소비자 정보 관리자 객체, 공급자로부터 수신된 데이터를 각 소비자에게 전달하기 위해 데이터를 분배하는 필터링 객체(Filtering), 각 소비자에 데이터 전송을 담당하는 전송 객체(Transfer)를 생성하는 예약자 객체(Subscriber)로 구성된다.

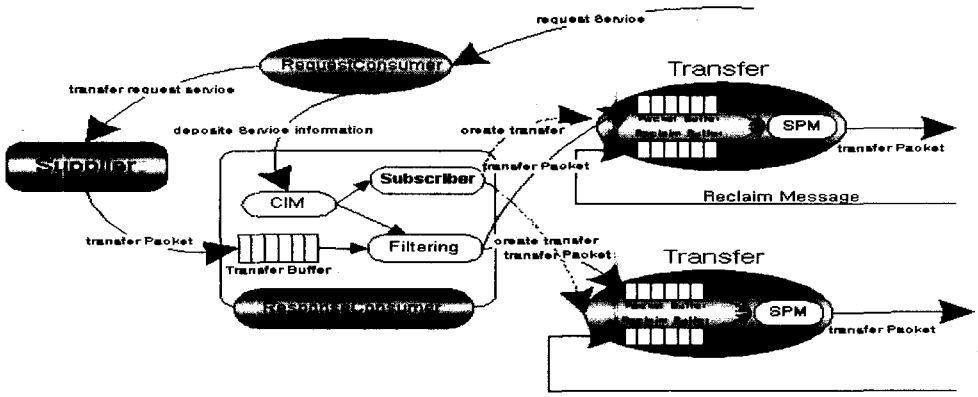
공급자를 채널에 연결하기 위한 공급자 측의 주요한 인터페이스와 오퍼레이션은 다음과 같다.

```
module MibCossEventChannel{
    interface RequestConsumer:CosEVentChannel::ProxyPushConsumer{
        /* . . . . . */
    };
    interface ResponseConsumer:CosEventChannel::ProxyPushConsumer{
        /* . . . . . */
    };
    interface SupplierAdmin{
        RequestConsumer obtain_request_consumer();
        ResponseConsumer obtain_response_consumer();
        RID     register_proxy(/* . . . */);
        boolean register_inforef(/* . . . */);
        boolean withdraw_proxy(/* . . . */);
    };
    interface MibEventChannel{
      SupplierAdmin for_supplier();
      void destory();
    };
};
```

(그림 9) 공급자측 인터페이스

3.4.3 정보관리자 및 협력자 객체

소비자의 서비스 요구 프록시 객체가 서비스를 제공할 수 있는 공급자와 연결하기 위해서는 공급자 서비스 요구 수신 프록시 객체의 레퍼런스를 알아야 한다. 또한, 공급자가 소비자에게 서비스를 제공하기 위해서는 서비스 하고자 하는 서비스 정보를 소비자에게 알려줘야 한다. 따라서



CIM : Consumer Information Manager
SPM : Server Packet Manager

(그림 8) 공급자측의 프록시 객체

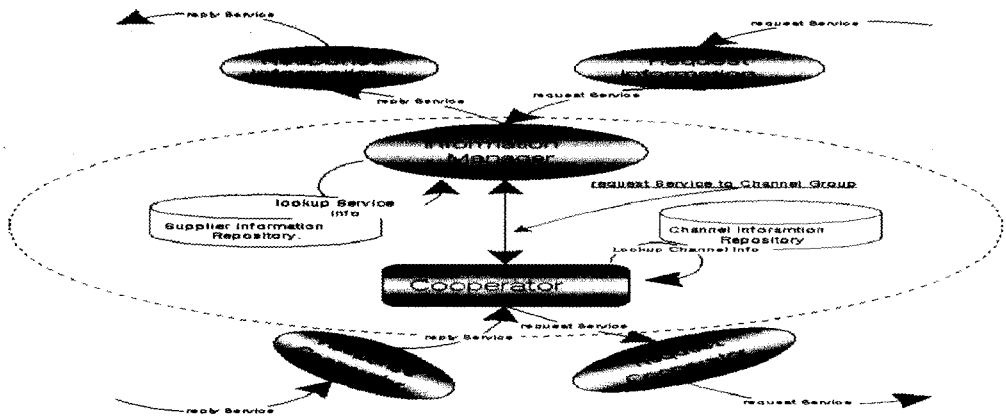
소비자에게 서비스 정보를 제공하고 공급자의 서비스 정보를 보관하는 기능을 제공하는 객체가 필요하다. 이러한 기능을 제공하는 정보 관리자 객체(Information Manager)는 채널에 연결된 공급자에 대한 정보를 저장·관리 기능 및 소비자에게 서비스 정보를 제공하는 객체로서, 채널에 등록한 공급자의 정보를 저장·관리하는 공급자 정보 저장소(Supplier Information Repository), 분산된 채널에 대한 정보를 저장·관리하는 채널 정보 저장소(Channel Information Repository)와 채널간에 상호연동을 통해 분산된 채널간에 서비스를 제공하는 협력자 객체(Cooperator)로 구성된다. 채널에 연결된 공급자들은 서비스 하고자 하는 서비스 정보와 소비자가 접속할 수 있는 서비스 요구 수신 프록시 객체에 대한 레퍼런스를 정보 관리 객체에 등록한다.

4. 결 론

분산환경에서의 통합 시스템(System Integration)을 구축하여 단일 시스템 뷰(Single System View)의 구축은 통신망, 하드웨어, 소프트웨어 및 사용하는 프로그래밍 언어 등의 이질성으로 인해 많은 어려움을 겪고 있다. 따라서 OMG에서는 특정 플랫폼(Platform)이나 기술에 관계없이 분산되고 이질적인 시스템간에 상호 운용할 수 있는 CORBA를 표준으로 제안하였다.

본 논문에서는 현재 개방된 상태로서 어플리케이션 개발자들에 의해 구성하도록 되어있는 이벤트 채널의 내부 연결 구조를 정의·설계 및 프로토타입으로 구현하여 기존의 이벤트 채널을 확장하였다. 소비자를 지역 채널에 연결하여 등록된 공급자들 중에 서비스를 제공할 수 있는 공급자가 존재하지 않는 경우 채널의 협력자 객체를 통해 채널그룹에 등록된 채널에 서비스를 요구할 수 있도록 하였다.

기존의 이벤트 채널은 단방향 통신으로서 서버가 클라이언트에게 메시지를 전송하는 형태인 반면에 본 논문에서 확장한 이벤트 채널은 소비자와 공급자를 채널에 연결하기 위해 소비자 측에는 서비스 요구 프록시 객체와 서비스 수신 프록시 객체로 구성하고, 공급자 측에서는 공급자로부터 요청한 서비스를 수신하는 서비스 요구 수신 프록시 객체와 공급자로부터 전송된 패킷을 소비자에 전송하는 서비스 응답 프록시 객체로 분리 구성함으로써 양방향 통신을 지원하도록 하였다. 비동기적 통신방법의 신뢰성을 보장하기 위해 공급자 프록시 객체와 소비자 프록시 객체 사이에 전송 객체를 두어 대리자(Agent) 기능을 수행하게 함으로써 신뢰성을 향상시켰다.



(그림 10) 정보관리자 객체간의 협력

또한, 모든 신규 채널들을 채널 그룹으로 그룹핑 하고 채널 그룹에 등록된 하나의 채널을 조정자 채널로 선정하여 채널 그룹에 등록한 채널에 대한 정보를 관리함으로써 채널 그룹에 등록된 모든 채널들이 동일한 그룹에 대한 뷰를 유지하여 채널 간에 멀티캐스트를 할 수 있다.

본 논문에서 설계·개발한 이벤트 서비스는 시스템 통합(System Integration) 및 다중 질의 검색 서비스에 활용할 수 있으며 객체 그룹 간에 신뢰성 향상을 위한 그룹 통신기법 등의 계속적인 연구가 필요하다.

Acknowledgement

이 논문은 1998년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

참 고 문 헌

[1] OMG, 'CORBA/IIOP 2.3.1 Specification', http://www.omg.org/corba/corbaiiop.html, October 1999

[2] J. Siegel, 'CORBA Fundamentals and Programming', John Wiley & Sons Inc., pp. 196-204, 1996

[3] OMG, 'Event Management Service', ftp::://ftp.omg.org/pub/docs/formal/97-07-13.ps, 1997

[4] P. Felber, R. Guerraoui, and A. Schiper. 'Replicating Objects using the CORBA Event Service.', In Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems, Tunis, October 1997.

[5] Xavier Defgo, Pascal Felber, Benoit Garbinato, Rachid Guerraoui, 'Reliability with CORBA event channels.', In Proceedings of the 3rd USENIX Conference on Object-Oriented Technologies and Systems (COOTS), pages 237-240, Portland, Oregon, USA, June 1997.

[6] D.C. Schmidt, S. Vinoski, 'Object Interconnections: The OMG Events Service(Columm 9)', SISG C++ report Magazine, February 1997.

[7] 정혜영, 김현규, 박양수, 구자록, 이명준, '자바를 이용한 CORBA 이벤트 서비스의 개발', 한국정보과학회, 가을학술발표논문집, Vol. 24, No. 2, 10, 1997.

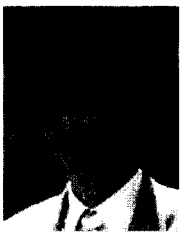
[8] Silvano Maffeis. 'The Object Group Design Pattern', Proceedings of the 1996 USENIX Conference on Object-Oriented Technologies. Toronto, Canada: The USENIX Association, June 1996

[9] Chris Gill, tim Harrison, Carlos O'Ryan, 'Using the Real-Time Event Service', http://siesta.cs.wustl.edu/~schmidt/events_tutorial.html

[10] David Levine and Tim Harrison, DavidL. Levine, Douglas C, Schmidit, 'The Design and Performance of a Real-time CORBA Object Event Service', Proceedings of OOPSLA '97, Atlanta, Georgia, October, 1997.

[11] Greg Eisenhauer, Bodhi Mukherjee and Chris Codella, 'On the Implementation of CORBA Event Channels', Tech Report GIT-CC-97-04, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280.

◐ 저 자 소 개 ◑



이 재 완

1984년 중앙대학교 전자계산학과 졸업(학사)
1987년 중앙대학교 대학원 전자계산학과 졸업(석사)
1992년 중앙대학교 대학원 전자계산학과 졸업(박사)
1992년-현재 군산대학교 전자정보공학부 교수
관심분야 : 분산 및 운영체제, 실시간 시스템, 컴퓨터 네트워크, 멀티미디어 등
E-mail : jwlee@kunsan.ac.kr