CORBA 이벤트 서비스의 연동 방법 및 성능 평가

(Interoperability Strategy Performance Evaluation of the Event Service of CORBA)

나 길 성 † 이 상 호 ††
(Gil Seong Na) (Sang Ho Lee)

요 약 웹은 단순하고 편리한 사용자 인터페이스로 인하여 폭 넓게 보급되었다. 사용자 인터페이스로서의 웹과 CORBA를 연동하기 위해 CORBA 분산객체를 자바 애플릿으로 작성하여 웹 브라우저에 직접 전달, 웹 브라우저에서 실행함으로써 웹과 CORBA를 연동한다. 웹에서 애플릿의 보안제약 사항을 위배하지 않고 CORBA의 이벤트 서비스를 사용하기 위해서는 이벤트 서비스의 연동이 필요하다. 본 논문에서는 CORBA와 웹의 연동에서의 CORBA 이벤트 서비스를 연동하기 위한 세 가지 방법을 제시하고 제안하는 세 가지 방법에 대한 성능 평가를 수행하여, 각 방법에 대한 장단점을 비교 분석하였다.

Abstract Since simple and convenient user interface, the Web is diffused widely. For integrating Web as a view of user interface and CORBA, CORBA object is made in applet, which is sent to Web browser in Client and executes by Web browser. To use CORBA Event Service in Web without violating a security constraints of applet, we need a interoperability of Event Service. This paper investigates interoperability of event channels of the event service of CORBA. We present three different methods that make one event channel be connected with other event channels. Pros and cons of each method are described. All mentioned approaches have been implemented in Java. An experimental performance evaluation has been carried out and evaluation results are also given.

1. 서 론

OMG(Object Management Group)에서 제정한 CORBA(Common Object Request Broker Architecture)는 현재 산업계의 표준으로 자리잡고 있다. CORBA는 분산 환경에서 여러 시스템을 통합하는데 필요한 기능으로써, 분산 시스템들 간의 운영체제나 프로그래밍 언어에 관계없이 객체서비스를 제공하여 독립적이고 표준화된 환경을 제공하며 객체간의 통신을 담당한다. CORBA는 객체 지향 개념을 바탕으로 하여 객체의 함수를 호출함으로써 서비스가 이루어진다. CORBA에서는 IDL(Interface Definition Language) 이라는 표준 언어를 제공하며 클라이언트 객체와 구현 객체 사이의 인터페이스를 정의한다. ORB(Object Request Broker)는 클라이언트와 구현 객체사이의 중계자 역할을 수행하는 여러 개의 컴포넌트와 인터페이스로 구성되어 있으며 분산된 객체들을 클라이언트/서버 관계로 만들어 주는 객체 미들웨어이다. ORB는 클라이언트 객체가 구현 객체의 함수를 호출할 때 ORB는 클라이언트의 호출을 받아 구현 객체들의 정보를 참조하여 클라이언트가 원하는 구현 객체를 찾고 클라이언트에서 보내준 서비스 요청 정보를 해당 구현 객체에게 전달한다. 구현 객체의 함수가 완료되면 ORB는 결과 값을 받아 클라이언트에게 전달한다.

1992년 소개된 월드 와이드 웹(World Wide Web, 또는 웹(Web)) 기술은 단순하고 편리한 사용자 인터페이스로 인하여 인터넷상에 분산되어 있는 정보 및 서비스를 접근하는 매체로 폭 넓게 보급되었다. 사용자 인터페이스로서의 웹과, 기존의 다양한 형태로 존재하는 하

† 비 회 원 : 숭실대학교 컴퓨터학부
gsna@drion.soongsil.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
shlee@computing.soongsil.ac.kr
논문접수 : 1999년 1월 20일
심사완료 : 1999년 11월 29일

드웨어 및 소프트웨어간의 이질적 시스템 통합을 제공하는 CORBA를 연동하기 위한 많은 기술적인 시도가 있었다[1-6]. 현재 웹과 CORBA 연동은 Java ORB를 이용한 방식[4]과 CGI(Common Gateway Interface)의 연동방식이 쓰이고 있다.

CGI를 통한 웹과 CORBA의 연동은 기존의 자원을 그대로 사용하면서 CORBA의 객체 서비스를 사용할 수 있도록 하는 방식이며 Java ORB를 이용한 연동은 CORBA 분산객체를 자바 애플릿(Applet)으로 작성하여 웹 브라우저에 직접 전달, 웹 브라우저에서 실행함으로써 웹과 CORBA를 연동하는 방식이다. 애플릿은 CGI와 HTTP 프로토콜을 통해서 서버로부터 클라이언트의 웹브라우저에 전송된다. 웹브라우저 안에서 실행되는 CORBA 애플릿 객체와 서버 객체는 IIOP 프로토콜을 통하여 CORBA 객체를 직접 호출할 수 있다.

CORBA에서 특정 객체에게 이벤트를 전달하여 새로운 서비스를 요구하거나 수행시키기 위해 제공되는 서비스가 이벤트 서비스이다. CORBA 이벤트 서비스에는 이벤트를 생성하는 역할을 하는 공급자(supplier) 객체와 공급자로부터 생성된 이벤트를 받아 처리하는 소비자(consumer) 객체, 그리고 공급자와 소비자의 중간 매개 역할을 하는 이벤트 채널 객체(Event Channel Object)가 존재한다. 공급자로부터 생성된 이벤트는 ORB를 통해 소비자에게 전송된다. CORBA 이벤트 서비스는 멀티캐스트 기능을 제공하여 복수의 이벤트 공급자와 소비자를 지원한다.

인터넷 사용자는 웹을 통해 자신이 원하는 정보만을 원하는 시간에 지속적으로 받고 싶어하며, 이에 대한 해결책으로 인터넷의 푸쉬(push) 기술이 개발되었다. 푸쉬 기술은 누군가가 웹사이트에 방문해 주기를 기다리는 수동적인 성격의 서핑(surfing)과는 달리 서비스업체가 이용자들이 원하는 정보를 자동으로 보내주는 것이다. 푸쉬 기술들은 종래의 브라우징과는 다른 방식으로 정보를 수집, 가공하는 방식을 제공한다. 푸쉬 기술을 이용하면, 인터넷 사용자들이 지금처럼 정보 수집을 위하여 웹사이트를 찾아 다녀야 하는 불편이 사라진다. 현재 푸쉬 기술을 제공하는 제품으로는 PointCast[7], Castanet[8], Downtown[9], BackWeb[10], NewsCatcher[11] 등이 있다. 이러한 푸쉬 기능을 웹과 CORBA의 이벤트 서비스를 이용하여 제공될 수 있다. CORBA의 이벤트 서비스를 이용한 애플릿을 클라이언트의 웹 브라우저로 전송한 후 애플릿은 서버 측의 이벤트 채널에 접속을 한다. 이렇게 함으로써 웹 브라우저의 애플릿은 공급자가 제공하는 이벤트를 이벤트 채널을 통하여 받을 수 있다.

CORBA의 이벤트 서비스를 이용하는 애플릿은 클라이언트의 브라우저로 전송된 후 애플릿 객체를 보내준 서버 이외에는 접속이 불가능하다는 제약사항이 있기 때문에 애플릿은 클라이언트의 컴퓨터로 전송된 이후에는 애플릿을 보낸 서버 이외의 다른 컴퓨터에는 접속을 할 수 없다. 이러한 제약사항 때문에 애플릿은 다른 서버에 존재하는 공급자 객체의 이벤트를 받을 수 없고 이 문제를 해결하기 위해서 공급자 객체는 서로 다른 서버에 존재하는 이벤트 채널에 등록하고, 이벤트를 전송해야 할 경우 공급자 객체는 각각의 이벤트 채널에 이벤트를 중복 전송해야 하는 번거로움이 발생한다. 이 문제를 해결하기 위해서는 서로 다른 서버에 존재하는 이벤트 채널을 연동함으로써 해결할 수 있다.

본 논문에서는 웹 환경 하에서 Java ORB를 이용하는 이벤트 서비스 연동 기술에 관하여 고찰하였다. 본 논문에서는 첫째, CORBA 이벤트 서비스를 연동하기 위한 세 가지 방법을 제시한다. 둘째, 제안하는 세 가지 방법에 대한 성능 평가를 수행하여, 각 방법에 대한 장단점을 비교 분석하였다. 본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로 본 논문에서 사용한 CORBA의 구현 프로그램인 Java ORB에 관해 기술하고 CORBA의 이벤트 서비스에 관해 살펴본다. 제 3장에서는 웹 환경 하에서 CORBA 이벤트 서비스의 기능 개선 목적, 기능과 구조 및 동작에 대해 기술한다. 제 4장에서는 구현된 CORBA 이벤트 서비스를 사용해 성능 평가를 수행한다. 제 5장에서는 CORBA 이벤트 서비스의 연동 방법에 대한 결론을 기술한다.

2. 관련연구

2.1 웹과 Java ORB의 연동

웹을 중심으로 중요한 환경으로 부각된 것은 자바와 분산 환경이다. 시스템 구축 환경으로서 웹을 사용하고, 웹 환경 하에서 동적인 시스템을 구축하기 위해 자바를 사용한다. 또한 이종의 시스템을 통합하기 위한 중간 기술로서 CORBA를 사용한다. 그러나 자바는 분산환경에서 다양한 기능을 제공하지만 자바만으로는 클라이언트/서버 구조를 지원하기에는 부족하다. 자바가 제공하는 유연성과 이식성으로 클라이언트를 구축하고 CORBA가 제공하는 분산 객체 지원 기능을 이용하여 서버를 구축하여 클라이언트/서버 구조를 지원한다[4].

웹과 CORBA를 연동하는 방법에는 웹의 CGI를 통한 연동 방법이 있다. 이 방법은 웹 서버의 외부 프로그램 연동 방법인 CGI를 이용해 CORBA 게이트웨이 프

로그램을 작성하는 것이다. 그러나, 이러한 방식은 게이트웨이를 항상 통해야 하므로 CORBA 객체를 직접 접근하지 못하고 동적으로 HTTP 문서를 생성해야 하며, HTTP 프로토콜은 상태를 유지하지 못하므로 상태유지를 위한 추가적인 처리가 요구된다.

위와 같은 웹의 CGI를 통한 CORBA 연동 방법의 문제점을 해결하기 위해 자바로 구현된 CORBA 프로그램을 사용하여 웹과 CORBA를 연동할 수 있다. 자바는 애플릿이라고 하는 인터넷 응용 프로그램의 작성이 가능하다. 자바로 작성한 애플릿은 넷스케이프(Netscape) 또는 익스플로러(Explorer)와 같은 웹 브라우저 안에서 실행 가능한 프로그램이다. 따라서 CORBA IDL(Interface Definition Language)을 자바로 구현하면 인터넷에서 CORBA의 응용 프로그램을 이용하게 된다. 자바에 의한 웹과 CORBA의 연동을 제공하는 자바 ORB는 CGI와 CORBA 연동의 문제점을 해결하고 다음과 같은 장점을 제공한다[4].

첫째, ORB를 바탕으로 객체와 객체 사이의 상호 작용으로 CGI 병목현상을 해결할 수 있다. CGI 프로그램은 외부로부터의 요청이 있을 경우 매번 실행이 되기 때문에 CORBA에 비해 클라이언트/서버 부하가 많다. 또한 HTTP를 통한 CGI 프로그램 호출은 CGI 프로그램이 실행된 후 결과를 반환하고 종료함으로써 호출간의 상태를 유지하지 못하나 CORBA는 서버 객체가 계속 존재함으로 상태 정보를 관리할 수 있다.

둘째, CORBA는 확장성을 가지는 서버와 서버 사이의 하부구조를 제공한다. CORBA에서는 ORB를 이용하여 서버 객체간에 통신이 이루어지며, 이러한 서버 객체는 부하조정(load-balancing)을 위하여 다중 서버에서 실행 될 수 있다. 그러나 CGI 방식에서는 모든 요구를 다중 서버 또는 다중 프로세서에게 분산 할 수 없기 때문에 병목현상이 발생한다.

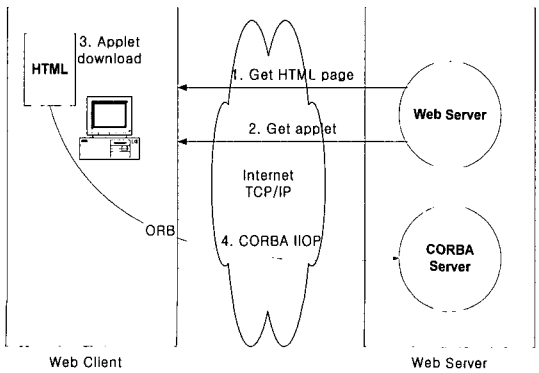

그림 1 Java 클라이언트와 Java ORB 기반 연동

[그림 1]은 자바 클라이언트와 자바 ORB를 이용하는 다음과 같은 객체 웹 모형을 갖는다. 웹 브라우저, 자바 클라이언트 응용, 애플릿 등은 클라이언트에 속하며, 웹 서버는 CORBA 및 HTTP 요구를 모두 서비스한다. 자바 클라이언트는 자바 ORB를 사용하여 CORBA 객체와 직접 통신한다.

2.2 CORBA 이벤트 서비스

CORBA 서비스는 광범위하게 CORBA 객체를 조작하는데 필요한 모든 서비스들을 지원하기 위한 인터페이스 명세이다[12,13]. 복잡한 시스템을 디자인할 경우, 특정 객체에게 이벤트를 전달하여 새로운 서비스를 요구하거나 수행시키기 위한 서비스가 필요하다. 이러한 기능을 제공하는 것이 CORBA 이벤트 서비스이다. CORBA 이벤트 서비스에는 이벤트를 생성하는 역할을 하는 공급자(supplier) 객체와 공급자로부터 생성된 이벤트를 받아 처리하는 소비자(consumer) 객체, 그리고 공급자와 소비자의 중간 매개 역할을 하는 이벤트 채널 객체(Event Channel Object)가 존재한다. 공급자로부터 생성된 이벤트는 ORB를 통해 소비자에게 전송된다. CORBA 이벤트 서비스는 멀티캐스트 기능을 제공하여 복수의 이벤트 공급자와 소비자를 지원한다.

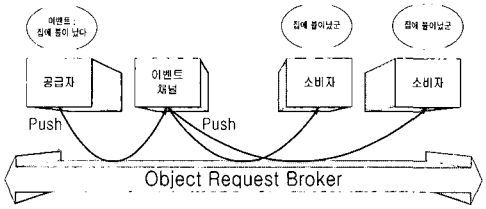

그림 2 푸쉬 모델

이벤트 서비스에는 두 가지 형태로 제공된다[14]. 첫 번째 모델을 푸쉬 모델이다. 푸쉬 모델은 이벤트 공급자가 주도적으로 소비자에게 이벤트를 전달한다. 공급자는 이벤트 채널의 *push* 함수를 실행함으로서 이벤트를 이벤트 채널에게 전송한다. 이벤트를 받은 이벤트 채널은 공급자로부터 전송된 이벤트를 원하는 소비자에게 전송한다. [그림 2]는 CORBA 이벤트 서비스 중 푸쉬 모델이다. 두 번째 모델은 풀 모델이다. 풀 모델에서는 소비자가 주도적으로 공급자의 이벤트를 이용하는 방식이다.

풀 모델에서는 소비자가 이벤트 채널 객체의 *pull* 함수를 실행함으로서 이벤트 채널에게 이벤트 전송을 요구한다. 소비자로부터 이벤트 전송을 요구받은 이벤트 채널은 공급자 객체에게 이벤트를 요구하고, 이벤트가 있을 경우 이벤트 채널을 통하여 소비자에게 전달된다. [그림 3]은 풀 모델이다.

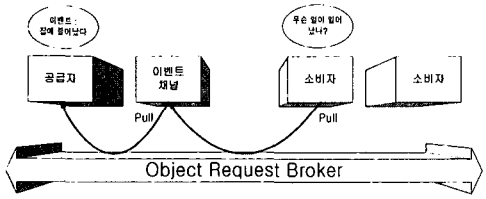

그림 3 풀 모델

3. CORBA 이벤트 서비스 기능 개선

3.1 이벤트 채널 연동 목적

CORBA 이벤트 서비스는 분산 객체들 사이에서 비동기적인 이벤트 발생과 그룹 통신을 유연성 있게 제공하며, 하나 또는 그 이상의 공급자가 하나 이상의 소비자에게 이벤트를 전달할 수 있도록 지원하고 있다. 웹 환경 하에서 CORBA의 이벤트 서비스를 이용해서 푸쉬 기술을 적용을 고려해 보자.

현재 이벤트 서비스는 공급자 객체가 이벤트를 소비자 객체에게 전달하려면 반드시 이벤트 채널을 통해서만 가능하다. 또한 소비자 객체가 공급자 객체로부터 이벤트를 받으려면 반드시 이벤트 채널을 통해서 받아야만 한다. 이는 소비자 객체가 원하는 이벤트가 여러 개이고 각각의 이벤트를 제공하는 공급자 객체가 서로 다른 이벤트 채널에 등록이 되어 있을 경우, 소비자 객체는 각 이벤트 채널에 접속을 해야 한다. [그림 4]에서 소비자 객체(3)이 공급자 객체(1)이 제공하는 이벤트를 받으려면 이벤트 채널(1)에 접속해야 한다.

그러나, 소비자 객체(3)이 애플릿으로 구현되고 클라이언트의 브라우저로 다운로드되어 브라우저 안에서 실행이 된다고 가정할 때 [그림 4]는 애플릿의 보안 제약사항으로 인하여 불가능하다. 즉, 애플릿은 애플릿 객체를 보내준 서버 이외에는 접속이 불가능하다는 제약사항이 있기 때문에 애플릿은 클라이언트의 컴퓨터로 전송된 이후에는 애플릿을 보낸 서버 이외의 다른 컴퓨터에는 접속을 할 수 없다. 구체적으로 소비자 객체는 자

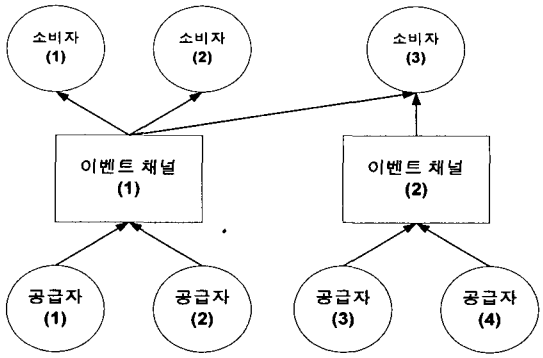

그림 4 소비자와 이벤트 채널

바 애플릿으로서 구현되며 웹 브라우저의 요청에 의해 클라이언트 컴퓨터에 전송된다. 클라이언트 컴퓨터에 전송된 소비자 애플릿 객체는 소비자 객체를 보낸 서버 컴퓨터 이외에는 접속할 수 없다. 따라서 웹 환경 하에서 CORBA의 이벤트 서비스를 이용하기 위해서 반드시 서버 컴퓨터에 이벤트 채널이 존재해야만 하고 클라이언트 컴퓨터에 전송된 소비자 애플릿 객체는 다른 서버 컴퓨터의 이벤트 채널에 접속 할 수 없다. 따라서 [그림 4]에서 소비자 객체(3)이 공급자 객체(1)의 이벤트를 받을 수 없다. 또한 공급자 객체는 소비자 객체가 어느 이벤트 채널에 접속할 지 모르므로 모든 이벤트 채널에 이벤트를 보내야만 한다. 소비자 객체와 공급자 객체가 각각 다른 이벤트 채널에 접속해도 이벤트 전송이 가능하기 위해서는 이벤트 채널을 연동해야 한다. 각각의 이벤트 채널을 서로 연동할 경우, 소비자 객체나 공급자 객체는 하나의 이벤트 채널에만 접속을 해도 원하는 이벤트를 받거나 전송할 수 있다[그림 5].

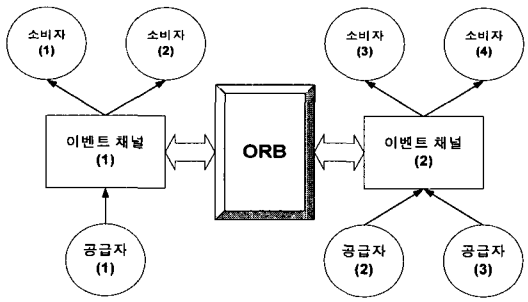

그림 5 이벤트 채널의 연동

3.2 이벤트 채널 연동 방법

본 논문에서는 위에서 제시한 CORBA 이벤트 채널의 단점을 보완하기 위해서 이벤트 채널을 연결하는 세 가지 방법을 제시한다.

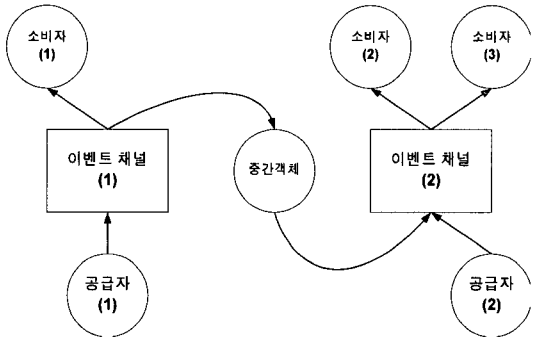

그림 6 중간객체를 이용한 연동

첫 번째 방법으로는 이벤트 채널 사이에 중간 객체를 두어 한쪽 이벤트 채널에서는 소비자 객체로 등록하고 다른 이벤트 채널에는 공급자 객체로 등록하는 것이다. [그림 6]은 첫 번째 방법으로 이벤트 채널을 연동한 것이다. [그림 6]에서 보듯이 중간객체는 이벤트 채널(1)에는 소비자 객체로서 등록하고 이벤트 채널(2)에는 공급자 객체로서 등록한다. 공급자(1)이 제공한 이벤트는 이벤트 채널(1)을 통해 소비자(1)과 중간객체에게 전달된다. 이벤트를 받은 중간객체는 이벤트 채널(2)에게 이벤트를 전송한다. 이벤트 채널(2)를 통해 소비자(2)와 소비자(3)은 공급자(1)의 이벤트를 받을 수 있다. 이처럼 이벤트 채널(1)의 이벤트를 중간객체가 받아 이벤트 채널(2)에 이벤트 공급하는 방법이다. 이 방법의 장점은 표준 CORBA의 이벤트 서비스 사양을 준수해서 만들어진 이벤트 채널이면 어떤 언어로 구현되었는지에 관계없이 연동이 가능하다. 단점으로는 중간객체를 통해서 이벤트가 전송됨으로 인한 속도저하가 발생할 수 있다.

두 번째 방법으로는 이벤트 채널 안에 있는 프락시 객체(Proxy Object)를 다른 이벤트 채널에 있는 프락시 객체에 바로 연결하는 방법이다. 프락시 객체는 이벤트 채널 안에서 생성되며 공급자 객체와 소비자 객체 사이에서 이벤트를 전달하는 역할을 한다. 프락시 소비자 객체는 공급자 객체가 이벤트 채널에 접속할 때 생성된다. 프락시 공급자 객체는 소비자 객체가 이벤트 채널에 접속할 때 생성되어 프락시 소비자 객체로부터 이벤트를 받아 소비자 객체에게 전송한다. [그림 7]에서 보듯이 공급자 객체가 전달한 이벤트는 소비자 객체에게 직접 전달되는 것이 아니라 프락시 객체들을 통해서 전달된다. 따라서 이벤트 채널(1)의 프락시 소비자 객체와 이벤트 채널(2)의 프락시 소비자 객체를 연결하면 푸쉬 공급자(1)이 제공하는 이벤트를 푸쉬 소비자(3)과 푸쉬 소비자(4)가 받을 수 있다. [그림 7]은 프락시 객체를 이용하여 연동한 그림이다.

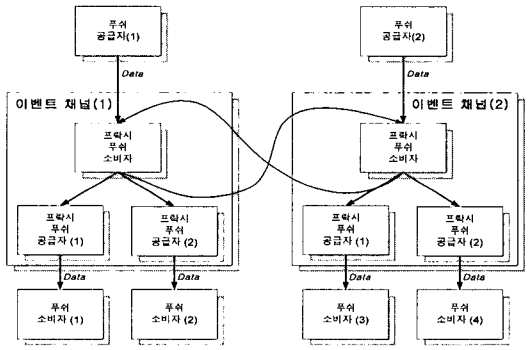

그림 7 프락시 객체를 이용한 연동

이 연동방법의 장점은 중간객체를 이용한 방법보다 이벤트 전달 속도가 빠르다는 점에 있다. 그러나 [그림 7]에서 이벤트 채널(1)에 있는 프락시 푸쉬 소비자 객체는 이벤트 채널(2)의 IOR이나 레퍼런스를 알고 있어야 한다.

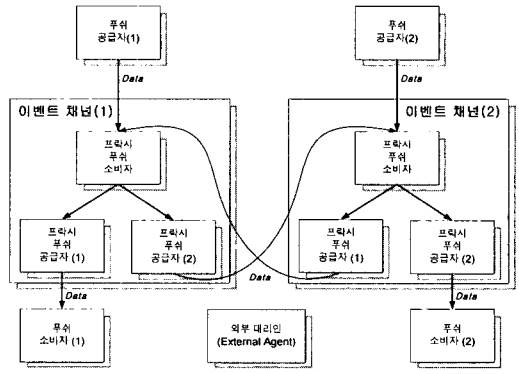

그림 8 외부 대리인을 이용한 이벤트 채널 연동

세 번째 방법으로는 외부 대리인 객체(External Agent Object)를 이용한 방법이다. 이 방법은 중간객체를 이용한 방법과 비슷하지만 외부 대리인 객체는 이벤트 전송에는 참여하지 않는다. 외부 대리인은 각 이벤트 채널의 프락시 객체를 연결해 주는 역할을 한다. 이벤트 채널이 외부 대리인 객체에게 다른 이벤트 채널과의 연

동을 요구 할 경우 외부 대리인 객체는 이벤트 채널 연동을 요구한 이벤트 채널에서 프락시 푸쉬 공급자 객체를 할당받고 다른 이벤트 채널로부터 프락시 푸쉬 소비자 객체를 요구하게 된다. 프락시 푸쉬 공급자 객체와 프락시 푸쉬 소비자 객체를 얻은 외부 대리인 객체는 이벤트 소비자로서 프락시 푸쉬 소비자 객체를 프락시 푸쉬 공급자 객체에게 연결한다. 한번 연결된 이벤트 채널은 외부 대리인 객체가 소멸되어도 이벤트 채널끼리 이벤트를 주고받을 수 있다. [그림 8]에서 보듯이 이벤트 채널(1)에서 외부 대리인 객체에게 이벤트 채널(2)와 연동을 요구할 경우 외부 대리인 객체는 프락시 푸쉬 공급자(2)의 레퍼런스와 이벤트 채널(2)의 프락시 푸쉬 소비자의 레퍼런스를 할당받은 후 프락시 푸쉬 소비자 객체와 프락시 푸쉬 공급자(2) 객체를 연결한다. 푸쉬 공급자(1)이 제공한 이벤트는 프락시 푸쉬 공급자(2)를 통해 이벤트 채널(2)에 있는 프락시 푸쉬 소비자에게 전달되고 푸쉬 소비자(2)는 푸쉬 공급자(1)이 제공한 이벤트를 받을 수 있다.

3.3 IDL

중간객체는 CORBA 객체로서 ORB를 통하여 이벤트 채널과 데이타를 주고받는다. 중간객체는 하나의 이벤트 채널에는 소비자 객체로 등록이 되고 다른 이벤트 채널에는 공급자 객체로 등록이 되기 때문에 소비자 객체가 가지고 있는 함수와 공급자 객체가 가지고 있는 함수를 모두 포함하고 있어야 한다. 이탤릭체로 표현된 함수는 새로 추가된 함수이다. 다음은 중간객체의 IDL이다.

```
module org
{
    module omg
    {
        module CosEventComm
        {
        interface ChannelBridge :
                    CosEventComm::PushConsumer,
                      CosEventComm::PushSupplier
            {
            }
        };
    };
};
```

PushConsumer와 PushSupplier에게 상속을 받은 ChannelBridge 객체는 세 개의 함수를 가진다. *void push(in any data) raises(Disconnected)* 함수는 중간객체가 소비자 객체로 등록된 이벤트 채널로부터 호출되는 함수이다. 공급자 객체로부터 이벤트를 받은 프락시 소비자 객체는 프락시 공급자 객체에게 이벤트를 전달하고 프락시 공급자 객체는 ORB를 통해서 소비자 객체의 push 함수를 호출한다. 이벤트를 받은 중간객체는 다른 이벤트 채널의 프락시 객체의 push 함수를 호출함으로서 이벤트를 전달한다. *void disconnect_push_consumer()* 함수는 소비자 객체에 있는 함수로서 이벤트 채널로부터의 접속을 중단할 때 호출되는 함수이다. *void disconnect_push_supplier()* 함수는 공급자 객체에 있는 함수로서 이벤트 채널과의 접속이 중단될 때 호출되는 함수이다.

프락시를 이용하여 이벤트 채널을 연동하려 할 경우 프락시 객체에 새로운 기능을 추가하여야 한다. 즉, 다른 이벤트 채널에 있는 프락시 객체에게 이벤트를 전달할 수 있는 인터페이스를 추가하여 구현해야 한다. 다음은 새로운 기능이 추가된 프락시 객체의 인터페이스이다.

```
module org
{
    module omg
    {
        module CosEventChannelAdmin
        {
            interface ProxyPushConsumer:
                 CosEventComm::PushConsumer
            {
                 void connect_push_supplier(
                      in CosEventComm::PushSupplier
                         push_supplier)
                      raises(AlreadyConnected);
                 void connect_proxy_push_consumer()
                      raises(AlreadyConnected);
        };
        };
    };
};
```

새로 추가된 *void connect_proxy_push_consumer()* 함수는 다른 이벤트 채널 안에 있는 프락시 객체와 접속하는 기능을 하는 함수이다. 다른 이벤트 채널의 ProxyPushConsumer 객체와 연결을 하기 위해서는 연결하고자 하는 이벤트 채널의 IOR을 알고 있어야 한다. 또한 기존의 ProxyPushConsumer 객체는 ProxyPushSupplier 객체에게만 이벤트를 전달하였다. 하지만 ProxyPushConsumer 객체와 ProxyPushConsumer 객체가 연결됨에 따라서 추가적인 기능이 요구된다.

외부 대리인 객체를 이용하여 이벤트 채널을 연동할 경우 외부 대리인 객체의 IDL 정의가 필요하다. 외부 대리인 객체는 이벤트 채널로부터 두 가지 요청을 받아 처리한다. 다음은 외부 대리인 객체의 IDL이다.

```
module org
{
    module omg
    {
        module CosEventChannelAdmin
        {
            interface ExternalAgent
            {
                void RegisterEventChannel(
                  EventChannel EventChannelObj );
                void ConnectEventChannel(
                  EventChannel EventChannelObj );
        };
        };
    };
};
```

RegisterEventChannel() 함수는 이벤트 채널이 외부 대리인 객체에게 등록하는 함수이다. *RegisterEventChannel()* 함수를 호출한 이벤트 채널이 외부 대리인 객체에게 등록함에 따라 후에 다른 이벤트 채널과 연동하려 할 경우, 또는 다른 이벤트 채널이 이벤트 채널과 연동하고자 할 경우 등록된 정보를 사용한다. *ConnectEventChannel()* 함수는 실제적인 연동을 하는 함수이다. *ConnectEventChannel()* 함수를 호출한 이벤트 채널은 현재 외부 대리인 객체에 등록된 이벤트 채널과 연동하게 된다.

4. 성능평가

본 장에서 기술한 이벤트 채널 연동 방법에 대한 성능 평가를 수행한다. 연동 방법에 대한 성능 평가를 수행하기 위한 시험 환경은 [그림 9]와 같다. 푸쉬 서버 컴퓨터의 CPU는 펜티엄프로-233MHz, RAM은 96MB이며 푸쉬 클라이언트 컴퓨터의 CPU는 펜티엄 II-233MHz, RAM은 96MB이다. 두 컴퓨터간의 네트워크 구성은 10BASET 이더넷을 사용하여 독립된 시험 환경을 구축하며, 네트워크의 대역폭은 10Mbps이다. 2개의 시스템은 Windows NT 4.0 운영체제에서 운영되고 있다. 본 논문에서 구현된 CORBA 이벤트 서비스는 자바로 구현되었으며, Windows NT 4.0 버전에서 Visual J++ 1.1로 컴파일하였다. 본 성능 평가에서는 푸쉬 서버에는 공급자 객체와 이벤트 채널이 존재하며, 푸쉬 클라이언트에는 소비자 객체와 이벤트 채널이 존재한다. 중간객체로 이벤트 채널을 연동할 경우 중간객체는 푸쉬 서버에 존재한다.

본 논문에서는 두 가지 평가 항목에(지연 시간(latency), 에러율(error rate)) 대하여 성능 평가를 수행한다. 지연 시간은 소비자가 이벤트를 처음 받는 시간과 마지막 이벤트를 받았을 때의 시간차로 계산한다. 이벤트 채널 연동에 있어서 각 연동방법에 의한 이벤트 전송 시간의 차이는 전체적인 성능에 영향을 준다. 에러율은 에러 발생 빈도를 측정하는 것으로, 공급자가 이벤트를 전송하고 소비자가 이벤트를 안전하게 받을 수 있는 확률을 의미한다. 많은 개수의 이벤트를 동시에 전달하여 이벤트 채널 연동방법의 안정성을 측정한다.

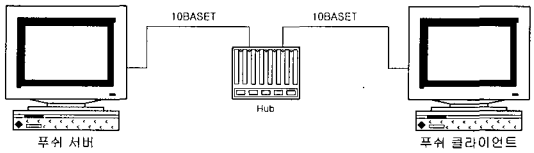

그림 9 성능평가환경

본 논문에서는 공급자의 개수를 1, 10, 20개로 10개씩 증가시키면서 이벤트 연동 방법의 성능을 평가하였다. 또한 각 이벤트 연동 방법에 대하여 10000개의 순차적인 이벤트 전송을 하였고 5회 측정하여 평균값을 지연 시간으로 산출하였다.

본 실험의 목적은 연동 방법에 의한 성능 평가이므로, 이벤트 채널내의 에러에 의한 오류는 제외하여야 한다. 예를 들어 같은 이벤트 채널 안에 있는 프락시 객체들간의 이벤트 전송에서 생기는 오류는 제외해야 한다. 이를 위하여 본 실험에서는, 이벤트 채널 자체의 이벤트 전송 에러를 검사하기 위해 각 이벤트 채널의 프락시 객체들은 전송된 이벤트를 모두 체크하여 실험 중 전송된 이벤트를 메모리에 저장한 후 실험이 끝난 후 지정된 파일에 전송된 이벤트의 리스트를 출력하여 이벤트 채널 자체의 에러율은 제외하였다.

[표 1]은 각 연동방법의 실험 결과이다. 1:1 이벤트 전송에서는 프락시 객체를 이용한 연동방법이 가장 좋은 결과를 나타내었고 중간객체를 이용한 연동방법이 가장 낮은 결과를 나타냈다. 또한 에러율은 0%를 나타내어 10000개의 이벤트 데이타가 모두 전송됨을 나타내었다. 공급자의 수를 10개로 증가해서 실험한 결과에서는 프락시 객체를 이용한 연동방법과 외부객체를 이용한 연동방법의 차이가 뚜렷하였다. [그림 10]은 5회 실험결과의 지연시간 평균을 나타내는 그래프이다.

위의 실험결과를 보듯이 이벤트 채널 연동방법에 있어서 프락시 객체를 이용한 연동 방법이 가장 좋은 성능을 나타내며, 외부객체를 이용한 연동 그리고 중간객체를 이용한 연동 방법 순으로 성능이 좋다. 중간객체를 이용한 연동방법의 실험 결과가 가장 낮게 나온 것은

표 1 실험결과

공급자 수	중간객체를 이용한 연동	프락시 객체를 이용한 연동	외부 객체를 이용한 연동
1	2분 2초 (0%)	1분 25초 (0%)	1분 26초 (0%)
10	24분 35초 (0%)	15분 20초 (0%)	15분 40초 (0%)
20	46분 29초 (0%)	32분 19초 (0%)	33분 11초 (0%)

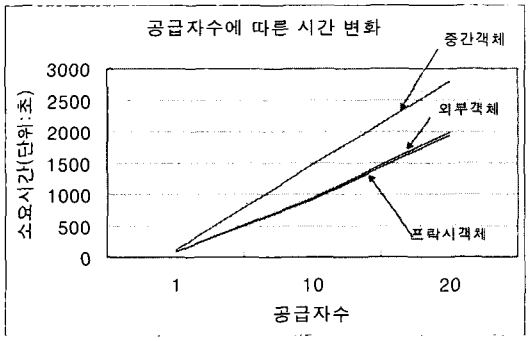

그림 10 공급자수와 평균소요시간의 변화

이벤트의 전송 경로가 가장 길기 때문이다. 중간객체는 하나의 CORBA 객체로서 존재하게 된다. 따라서 중간객체를 통해서 이벤트를 전송하려 할 경우 이벤트 채널 안에 있는 프락시 객체는 이벤트 전송을 위해 중간객체의 특정 함수를 실행해야 하고 이에 따른 오버헤드가 발생함으로써 성능이 저하된다. 이에 비해 프락시 객체와 외부 객체를 이용한 연동 방법은 중간객체를 이용한 방법보다 이벤트 전송 경로가 짧기 때문에 더 나은 성능을 나타냄을 알 수 있다. 외부 객체를 이용한 연동 방법이 프락시 객체를 이용한 연동방법보다 성능이 약간 낮게 나온 것은 내부적인 추가 경로이기 때문이다. 프락시 객체를 이용한 방법에서는 프락시 소비자 객체가 다른 이벤트 채널이 프락시 소비자 객체에게 이벤트를 바로 전송하지만 외부 대리인 객체를 이용한 방법에서는 프락시 소비자 객체가 이벤트를 전달하기 위해서 프락시 공급자 객체는 이벤트를 전송하고 프락시 공급자 객체가 다른 이벤트 채널에 있는 프락시 소비자 객체에게 이벤트를 전송해야 한다. 따라서 중간객체를 이용한 방법에는 없는 추가적인 이벤트 경로가 필요하게 된다.

한편, 프락시 객체를 이용한 방법은 프락시 소비자 객체의 내부 구조를 변경할 필요가 있어, 이를 구현하기 위해서는 이벤트 서비스 모듈의 소스 코드가 필요하다. 그러나 외부 객체를 이용한 방법은 이벤트 채널 내부의 소스코드를 수정할 필요가 없기 때문에 다른 이벤트 채널과의 연동에서 프락시 객체를 이용한 방법보다 호환성이 우수하다.

5. 결 론

월드 와이드 웹 기술은 단순하고 편리한 사용자 인터페이스로 인하여 인터넷상에 분산된 있는 정보 및 서비스를 접근하는 매체로 폭발적인 인기를 누리고 있다. 사용자 인터페이스로서의 웹과, 기존의 다양한 형태로 존재하는 하드웨어 및 소프트웨어간의 이질적 시스템 통합을 제공하는 CORBA를 연동하기 위한 많은 기술적인 시도가 있었다. 본 논문에서는 CORBA와 웹의 연동에서의 CORBA 이벤트 서비스를 연동하기 위한 세 가지 방법을 제시하고 제안하는 세 가지 방법에 대한 성능 평가를 수행하여, 각 방법에 대한 장단점을 비교 분석하였다.

전체적으로 고찰할 때, 이벤트 채널 연동방법에 있어서 프락시 객체를 이용한 연동 방법이 가장 좋은 성능을 나타내며, 외부객체를 이용한 연동 그리고 중간객체를 이용한 연동 방법 순으로 성능이 좋다. 공급자의 수가 증가함에 따라 중간객체를 이용한 연동 방법은 평가시간이 급격하게 증가하고 이에 비해 프락시 객체를 이용한 방법과 외부 객체를 이용한 방법은 완만한 증가를 보였다. 공급자 수가 1개일 때는 프락시 객체를 이용한 방법과 외부 객체를 이용한 방법의 차이가 미비했으나 공급자의 수를 증가할수록 추가적인 이벤트 경로를 가진 외부 객체를 이용한 방법에서 그 차이가 뚜렷함을 알 수 있다. 향후 연구방향으로는 위에서 제시한 방법을 실제 웹 환경에 적용하고, Visibroker나 Orbix와 같은 다른 상용 CORBA 제품의 이벤트 채널과 연동해서 각 상용제품과의 성능 평가를 수행할 예정이다.

참 고 문 헌

[1] G. Almasi and V. Jagannathan, Integrating the WWW and CORBA-based Environments, Technical Report CERC-TR-RN-95-005, Concurrent Engineering Research Center, West Virginia University, 1995.

[2] O. Rees, N. Edwards, M. Madsen, M. Beasley, and A. McClenaghan, A Web of Distributed Objects, Proceedings of the 4th International World Wide Web Conference, 1995.

[3] P. Merle, C. Gransart and J.M. Geib, CorbaWeb: A

Generic Object Navigator, Proceedings of the 5th International World Wide Web Conference, 1996.
[4] R. Orfali, D. Harkey, Client/Server Programming with JAVA and CORBA, John Wiley and Sons, pp.47-72, 1997.
[5] G. Brose, JacORB: Implementation and Design of a Java ORB, Proc. of IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems: 143-154, 1997.
[6] D. Glance, Multicast Support for Data Dissemination in OrbixTalk, IEEE Data Engineering Bulletin 19(3): 31-39, 1996.
[7] PointCast <http://www.pointcast.com>
[8] Marimba <http://www.marimba.com>
[9] Downtown <http://www.incommon.com>
[10] BackWeb <http://www.backweb.com>
[11] NewsCatcher <http://www.airmedia.com>
[12] OMG, Event Service Specification, 1995. <http://www.omg.org>
[13] R. Orfali, D. Harkey, J. Edwards, The Essential Distributed Objects Survival Guide, John Wiley and Sons, 1996.
[14] Visigenic Software, VisiBroker for Java Naming and Event Services Programmer's Guide Version 3.2, 1998.
[15] The problem of Push By Alex Lash, Staff Writer, CNET News.com September 30, 1997. <http://www.news.com/SpecialFeatures/0,5,14757,00.html>
[16] M. Campione, K. Walrath, The Java™ Tutorial Object-Oriented Programming for the Internet, Addison Wesley, 1996.
[17] M. Franklin, Push vs Pull, 1997. <http://www.cs.umd.edu/users/franklin/slides/index.htm>
[18] M. Franklin and S. Zdonik, Data in Your Face: Push Technology in Perspective, Proceedings of ACM SIGMOD, 1998.
[19] M. Franklin and S. Zdonik, A Framework for Scalable Dissemination-Based System, Proceedings of OOPSLA, 1997.
[20] M. Hughes, C. Hughes, M. Shoffner, M. Winslow, Java Network Programming, MANNING, 1996.
[21] R. Orfali, D. Harkey, J. Edwards, Instant CORBA, John Wiley and Sons, 1997.
[22] J. Siegel, CORBA Fundamentals and Programming, John Wiley and Sons, 1996.
[23] A. Vogel, Building Distributed Systems in Java, 1996.<http://www.dstc.edu.au/AU/staff/andreas-vogel/ papers/obj-exp96/paper.html>

나 길 성

1997년 숭실대학교 전자계산학과(학사). 1999년 숭실대학교 컴퓨터학과(석사). 1999년 ~ 현재 닉스테크주식회사 재직중. 관심분야는 CORBA, 데이타베이스

이 상 호

1984년 서울대학교 컴퓨터공학과(학사). 1986년 미국 노스웨스턴 대학교 전산학과(석사). 1989년 미국 노스웨스턴 대학교 전산학과(박사). 1990년 ~ 1992년 한국전자통신연구소 데이타베이스실. 1992년 ~ 현재 숭실대학교 컴퓨터학부 부교수. 1999년 ~ 2000년 미국 George mason 대학교 방문교수. 관심분야는 인터넷 데이타베이스, 데이타베이스 성능평가, 트랜잭션 처리