SOAP 기반 응용프로그램 디스패처의 설계 및 구현

김신강† 임효상† 이정훈† 한욱신† 황규영†

†한국과학기술원 전산학과/첨단정보기술연구센터

{sgkim, lorien, handol, wshan, kywhang}@mozart.kaist.ac.kr

Design and Implementation of a SOAP-Based Application Logic Dispatcher

Shin-Gang Kim† Hyo-Sang Lim† Jeong-Hoon Lee† Wook-Shin Han† Kyu-Young Whang†

†Department of Electronic Engineering & Computer Science

Division of Computer Science and

Advanced Information Technology Research Center

Korea Advanced Institute of Science and Technology

요 약

컴퓨터 기술의 발전과 인터넷의 보급에 따라 분산된 컴퓨팅 자원과 정보를 효율적으로 사용하기 위한 분산 응용프로그램의 개발이 활발히 이루어 지고 있다. 분산 응용프로그램 개발 표준으로는 RPC, CORBA, EJB 등이 있고, 각각 독자적인 통신 프로토콜을 사용하여 분산 응용프로그램을 호출할 수 있는 응용프로그램 디스패처를 제공한다. 응용프로그램 디스패처는 분산 응용프로그램 개발 플랫폼의 핵심 모듈로서, 개발자가 작성한 응용프로그램을 저장, 관리하면서 클라이언트로부터의 응용프로그램 수행 요청을 받아 그에 맞는 응용프로그램을 분기 시켜 수행하고 결과를 돌려주는 기능을 수행한다. 최근 W3C 에서는 분산 응용프로그램을 통합된 방법으로 호출할 수 있도록 하는 통신 프로토콜인 SOAP 을 제정하였다. 본 논문은 SOAP 을 기반으로 하는 응용프로그램 디스패처를 설계하고 구현한다. 본 논문에서 제안하는 시스템은 다음과 같은 특징을 가진다. 첫째, C, C++와 Java 로 작성된 다양한 응용프로그램 수행을 지원함으로써 분산 응용프로그램 개발을 위한 핵심 플랫폼으로서 사용될 수 있다. 둘째, 응용프로그램의 오류로 인하여 발생할 수 있는 문제에 대처하여 안정적인 수행을 제공한다. 셋째, SOAP 표준을 따름으로써 SOAP 을 지원하는 다른 분산 응용프로그램 개발 표준들과 상호 호출이 가능하다.

1. 서 론

컴퓨터 기술의 발전과 인터넷의 보급에 따라 분산된 컴퓨팅 자원과 정보를 효과적으로 활용하기 위하여 네트워크 상에 존재하는 여러 대의 컴퓨터를 사용하여 서비스를 제공하는 분산 시스템 환경이 널리 쓰이고 있다[1]. 이에 따라 응용프로그램도 분산 시스템을 활용하는 분산 응용프로그램 형태로 발전하고 있다.

현재까지 분산 응용프로그램을 개발하기 위하여 여러 가지 프로그래밍 모델이 제시 되었으며 실제 응용프로그램 개발에 널리 사용되고 있다. 그 예로는 네트워크를 통하여 다른 컴퓨터에 있는 프로시져 형태의 응용프로그램을 호출하는 RPC(Remote Procedure Call)[2]와 객체 형태의 분산 응용프로그램을 호출하기 위해 OMG(Object Management Group)에서 제정한 CORBA(Common Object Request Broker Architecture)[3], SUN 에서 제정한 EJB(Enterprise JavaBeans)[4], 그리고 마이크로소프트(Microsoft)사에서 개발한 윈도우즈(Windows) 기반 분산 객체 표준인 DCOM(Distributed Component Object Model) 등이 있다.

그러나 RPC, CORBA, EJB, DCOM 과 같은 분산 응용프로그램 개발 표준들은 서로 호환되지 않는 각자의 독립적인 통신 방법을 사용하여 응용프로그램을 호출하고 있기 때문에 서로 다른 표준으로 개발된 분산 응용프로그램 간의 연동이 불가능하여 상호 운용성(interoperability)이 부족한 문제점을 가진다. 최근 W3C 에서는 기존의 분산 응용프로그램 개발 표준을 통합된 방법으로 호출할 수 있도록 SOAP(Simple Object Access Protocol)[5]이라는 통신 프로토콜을 제정하였고 이에 따라 분산 응용프로그램 개발 표준과 상용 시스템에서도 기존의 통신 프로토콜 이외에 SOAP 을 추가로 지원하는 추세에 있다[6][7][8][9][10].

본 논문은 SOAP 을 기반으로 하는 응용프로그램 디스패처를 설계하고 구현한다. 응용프로그램 디스패처는 개발자가 작성한 응용프로그램을 저장, 관리하고 클라이언트의 응용프로그램 수행 요청을 받아 그에 맞는 응용프로그램을 분기 시켜 수행하고 결과를 돌려주는 시스템이다. 응용프로그램 디스패처는 네트워크를 통하여 클라이언트의 응용프로그램 수행 요청을 처리함으로써 분산 응용프로그램 개발을 위한 핵심 플랫폼으로서 사용될 수 있다.

본 논문에서 구현한 SOAP 기반 응용프로그램 디스패처는 C++와 Java 형식으로 개발된 다양한 응용프로그램의 수행을 지원하여 분산 응용프로그램 개발을 위한 핵심 플랫폼으로 사용될 수 있다. 또한 개발자가 작성한 응용프로그램 오류로 인하여 발생할 수 있는 문제점들에 대처할 수 있는 방법들을 제공하여 디스패처가 안정적으로 동작할 수 있도록 한다. 그리고 디스패처에 저장된 응용프로그램에 대한 수행 요청과 결과 전달을 위한 통신 수단으로는 SOAP 표준을 따름으로서 다른 분산 응용프로그램 개발 표준들과의 상호 호출이 가능하다.

본 논문의 구성은 다음과 같다. 제 2 장에서 기존에 많이 사용되고 있는 분산 응용프로그램 호출 표준을 소개한 뒤 SOAP 의 소개와 현재의 지원 현황에 대해서 설명한다. 제 3 장에서 본 논문에서 구현한 SOAP 기반 응용프로그램 디스패처 시스템의 아키텍처와 구성 모듈들에 대하여 설명하고 제 4 장에서는 응용프로그램의 오류로 인하여 발생할 수 있는 문제점과 이를 해결하기 위한 방법에 대해 설명한다. 마지막으로 제 5 장에서는 결론을 내린다.

2. 관련 연구

본 장에서는 연구 배경으로서 2.1 절에서는 RPC, CORBA, EJB 와 같은 기존의 분산 응용프로그램의 개발 표준에 대해서 설명한다. 2.2 절에서는 기존의 분산 응용프로그램 개발 표준이 지니는 상호 운용성 문제를 해결하기 위해 최근 W3C 에서 제정한 XML 기반의 분산 응용프로그램 호출 표준인 SOAP 에 대해서 설명한다. 마지막으로 2.3 절에서는 기존 분산응용프로그램 개발 표준들의 SOAP 지원 현황에 대해서 살펴본다.

2.1. 기존의 분산 응용프로그램 개발 표준

본 절에서는 SOAP 이 제정되기 이전부터 널리 사용되고 있는 분산 응용프로그램 개발 표준인 RPC, CORBA, EJB 에 대해서 설명한다.

RPC(Remote Procedure Call)는 네트워크 상의 다른 컴퓨터에 있는 프로시져를 호출하는데 사용되는 응용프로그램 개발 표준이다[2]. RPC 는 클라이언트와 서버간에 위치에 상관없이 네트워크를 통하여 프로시져

* 본 연구는 첨단정보기술연구센터를 통하여 한국과학재단의 지원을 받았음.

형태의 응용프로그램을 호출할 수 있도록 한다. 그러나 객체지향(object-oriented) 프로그래밍 언어가 널리 쓰이게 되면서 객체 기반의 분산 응용프로그램 개발 표준이 필요하게 되었다.

CORBA(Common Object Request Broker Architecture)는 분산 환경에서 객체들간의 상호 운용을 제공하기 위해 OMG(Object Management Group)에서 제정한 객체 생성, 배포, 관리에 대한 표준이다[3]. CORBA 는 객체 형태로 개발된 응용프로그램들 간에 통신과 데이타 전송에 대한 표준을 제공하여 개발된 객체들을 프로그래밍 언어와 플랫폼에 상관없이 상호 접근할 수 있는 상호 운용성을 제공한다.

CORBA 와 더불어 최근 많이 사용되는 분산 객체 호출 표준으로는 EJB(Enterprise JavaBeans)가 있다. EJB 는 Java 형식의 응용프로그램을 개발하고 배포하기 위해 SUN 에서 제정한 Java 응용프로그램 개발 플랫폼에 관한 규격이다[4].

그러나 앞에서 설명한 기존의 분산 응용프로그램들은 각각의 통신 프로토콜을 가지고 통신하기 때문에 서로 다른 응용프로그램 개발 표준과는 상호 운용되지 못하는 문제점을 가지고 있다.

2.2. XML 기반의 분산 응용프로그램 호출 표준 (SOAP)

SOAP(Simple Object Access Protocol)은 분산 환경의 객체들을 표준화된 방법으로 접근하기 위해 W3C 에서 제정한 통신 프로토콜이다[5]. SOAP 은 XML(eXtensible Markup Language)형식으로 이루어져 있고 HTTP(Hyper Text Transfer Protocol)를 사용하여 전송된다. SOAP 은 응용프로그램 수행을 요청하는 클라이언트와 응용프로그램 수행을 담당하는 서버로 이루어진 '클라이언트-서버' 환경에서 클라이언트와 서버간에 응용프로그램을 호출할 수 있는 통신 수단으로 사용된다.

SOAP 은 웹에서 문서 교환 표준으로 자리잡고 있는 XML[11]형식과 현재 인터넷에서 가장 널리 사용되고 있는 웹 환경의 통신 표준인 HTTP 를 사용함으로서 응용프로그램의 개발 표준, 개발 언어 그리고 플랫폼에 독립적으로 사용될 수 있다는 특징을 가진다. 이러한 특징에 따라 기존의 분산 응용 프로그램 개발 표준들도 SOAP 을 지원하고자 하는 움직임을 보이고 있다.

2.3. SOAP 지원 현황

RPC 의 경우는 RPC 호출에 필요한 정보를 SOAP 의 형태로 주고 받는 XML-RPC 방법을 제안하고 사용한다[6]. CORBA 의 경우 상업시스템 업체들이 SOAP 과 CORBA 와의 연동 방안을 OMG 에 제시하였고, OMG 에서는 이를 받아들여 현재 기존의 CORBA 표준에 SOAP 지원을 추가하는 방안을 추진중이다[7]. 상용 CORBA 시스템 개발 업체에서도 SOAP 을 통신 수단으로 할 수 잇는 시스템을 개발하였다. EJB 의 경우는 기존의 통신 방법 이외에 SOAP 으로 통신할 수 있도록 API[8]를 제공한다.

현재 상용 응용프로그램 서버들 가운데 가장 많이 사용되고 있는 BEA 사의 WebLogic 은 SOAP 을 사용하여 응용프로그램을 수행할 수 있는 플랫폼과 라이브러리를 제공한다[9]. 마이크로소프트사의 차세대 응용프로그램 개발 플랫폼인 .NET 의 경우도 웹 환경의 HTTP 를 사용한 통신은 SOAP 을 채택하여 사용한다[10].

이와 같이 SOAP 이 가진 상호 운용성이라는 장점을 활용하기 위해 기존의 분산 응용프로그램 개발 표준과 상용시스템에서 SOAP 을 지원하는 안을 추진하고 있다. 이러한 흐름에 따라 분산 환경에서 개발되는 대부분의 분산 응용프로그램의 통신 방법은 SOAP 으로 통합될 것으로 예상된다.

3. SOAP 기반 응용프로그램 디스패처의 설계 및 구현

본 장에서는 SOAP 기반 응용프로그램 디스패처의 설계 및 구현에 대하여 설명한다. 3.1 절에서는 SOAP 기반 응용프로그램 디스패처의 시스템 아키텍처에 대하여 설명한다. 그리고 3.2 절부터 3.5 절까지 시스템 아키텍처를 구성하고 있는 모듈에 대해 각각 설명한다.

3.1. 시스템 아키텍처

본 절에서는 본 논문에서 구현한 SOAP 기반 응용프로그램 디스패처의 시스템 아키텍처에 대해서 설명한다. SOAP 기반 응용프로그램 디스패처는 클라이언트로부터 SOAP 형식의 응용프로그램 수행 요청을 받아 클라이언트가 원하는 응용프로그램을 수행한 뒤 수행 결과를 SOAP 형식의 XML 문서로 작성하여 돌려주는 시스템이다. 네트워크 통신을 사용하여 다양한 언어로 작성된 응용프로그램 수행 요청을 처리함으로써 분산 응용프로그램 개발을 위한 핵심 플랫폼으로서 사용될 수 있고 응용프로그램 수행 요청을 위한 통신 방법으로는 XML 기반의 응용프로그램 호출 표준인 SOAP 을 사용함으로써 기존의 분산 응용프로그램 개발 표준과 상호 운용될 수 있다.

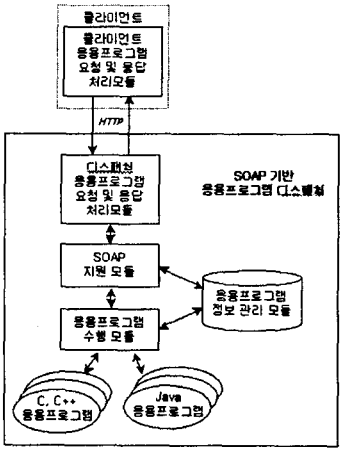

그림 1. SOAP 기반 응용프로그램 디스패처 아키텍처.

그림 1 은 SOAP 기반 응용프로그램 디스패처의 아키텍처를 나타낸다. 본 논문의 시스템은 크게 응용프로그램 요청 및 응답 처리 모듈, SOAP 지원 모듈, 응용프로그램 수행 모듈 및 응용프로그램 정보 관리 모듈로 구성된다. 응용프로그램 요청 및 응답 처리 모듈은 클라이언트와 디스패처간의 HTTP 를 통한 네트워크 통신 기능을 담당한다. SOAP 지원 모듈은 응용프로그램 수행 요청 및 응답을 위하여 SOAP 표준에 따라 작성된 XML 문서와 디스패처 내부 자료간의 변환 기능을 담당한다. 응용프로그램 수행 모듈은 프로그래밍 언어와 개발자가 기술한 수행방법에 따라 응용프로그램을 수행하고 결과를 돌려 주는 기능을 담당한다.

본 논문에서 구현한 디스패처는 여러 개의 응용프로그램 수행 요청을 동시에 수행하기 위하여 멀티 프로세스(multi-process) 구조로 운용된다. 이러한 구조는 여러 개의 클라이언트로부터 응용프로그램 수행 요청이 디스패처에 전달되었을 때 각각의 요청을 처리할 프로세스를 할당함으로써 다른 클라이언트의 요청이 끝나기를 기다리지 않고 동시에 수행될 수 있도록 한다.

3.2. 응용프로그램 정보 관리 모듈

본 절에서는 SOAP 기반 응용프로그램 디스패처의 응용프로그램 정보 관리 모듈에 대해서 설명한다. 응용프로그램 정보 관리 모듈에서는 응용프로그램에 대한 정보를 각각 객체정보, 메쏘드정보, 매개변수정보, 반환정보의 자료구조로 저장하면서 SOAP 지원 모듈과 응용프로그램 수행 모듈에게 응용프로그램 수행에 필요한 정보를 제공한다.

3.3. SOAP 지원 모듈

본 절에서는 SOAP 기반 응용프로그램 디스패처의 SOAP 지원 기능에 대해서 설명한다. SOAP 지원 모듈은 응용프로그램 수행 요청 및 응답을 위하여 SOAP 표준에 따라 작성된 XML 문서와 디스패처 내부 자료구조간의 변환을 담당함으로서 디스패처가 SOAP 을 지원하도록 한다. 이때 변환 기능을 수행하기 위하여 DOM(Document Object Model)[12] API 를 사용한다.

SOAP 지원 모듈을 통하여 SOAP 표준을 따르는 응용프로그램 수행 요청과 응답 메세지를 생성 및 처리하여 SOAP 을 지원하는 시스템과 표준화된 방법으로 상호 호출될 수 있도록 한다.

3.4. 응용프로그램 수행 모듈

본 절에서는 SOAP 기반 응용프로그램 디스패처의 응용프로그램 수행 기능에 대해서 설명한다. 응용프로그램 수행 모듈은 SOAP 지원 모듈로

부터 받은 내부 자료구조 형식의 응용프로그램 수행 요청 정보를 사용하여 응용프로그램을 수행하고 수행 결과를 SOAP 지원 모듈로 돌려주는 역할을 담당한다. 수행하는 응용프로그램은 개발 언어와 디스패처에 등록될 때 개발자에 의해서 주어지는 수행모드 값에 따라 각각 다른 방식으로 수행 된다.

본 논문에서 구현한 SOAP 기반 응용프로그램 디스패처는 C, C++, Java 프로그래밍 언어로 개발된 응용프로그램을 각각 C, C++의 경우는 동적으로(dynamically) 링크(link)방법을 Java 의 경우 SUN 의 JNI 라이브러리를 사용하여 Java 클래스를 수행한다. 또한 응용프로그램에 오류가 있을 가능성이 있는 경우 오류로 인하여 전체 시스템에 영향 받는 것을 막기 위하여 디스패처 프로세스와 분리된 독립된 프로세스에서 수행되도록 하는 방법을 제공한다. 이러한 방법은 응용프로그램 오류로부터 시스템을 보호하기 위한 것으로 4 장에서 자세히 설명한다.

3.5. 응용프로그램 요청 및 응답 처리 모듈

본 절에서는 본 논문에서 구현한 SOAP 기반 응용프로그램 디스패처의 응용프로그램 요청 및 응답 처리 기능에 대하여 설명한다. 응용프로그램 요청 및 응답 처리 기능은 응용프로그램 수행을 요청하는 클라이언트와 응용프로그램 수행을 제공하는 디스패처 간의 HTTP 를 통한 네트워크 통신 기능을 수행한다. 응용프로그램 요청과 응답 처리 기능은 응용프로그램 수행 요청을 SOAP 형식의 XML 문서로 변환하여 서버쪽으로 전송하고 결과를 받는 클라이언트쪽 부분과 클라이언트의 요청을 받고 결과를 전송하는 디스패처쪽 부분으로 나뉘어 수행된다.

4. 디스패처에서의 응용프로그램 오류 처리 방법

본 장에서는 개발자가 작성한 응용프로그램 오류에 대한 SOAP 기반 응용프로그램 디스패처에서의 대처 방법을 설명한다. 4.1 절에서는 디스패처에서 개발자가 작성한 응용프로그램 수행 중 발생한 오류로 인하여 어떤 문제점이 발생할 수 있는지 살펴본다. 4.2 절에서는 응용프로그램 수행 오류가 시스템에 미치는 영향을 줄이기 위한 방법으로 응용프로그램을 디스패처와 분리된 프로세스에서 수행하는 방법을 설명한다. 4.3 절에서는 응용프로그램 오류가 발생했을 때 디스패처에서 그에 대처하는 방법을 설명한다.

4.1. 디스패처에서의 응용프로그램 오류 문제

본 절에서는 디스패처에서 응용프로그램의 오류로 인하여 발생할 수 있는 문제를 설명한다. 디스패처에서의 *응용프로그램 오류 문제*란 디스패처가 수행을 제공하는 응용프로그램의 오류로 인하여 디스패처 자체나 디스패처에서 수행 중인 다른 응용프로그램이 올바르지 못하게 동작할 수 있다는 문제를 말한다. 본 논문에서 구현한 디스패처는 사용자가 작성한 응용프로그램과 연동하여 동작하기 때문에 잘못 작성된 응용프로그램의 수행 오류로 인하여 시스템이 영향 받을 수 있다.

본 논문에서 구현한 시스템은 응용프로그램의 오류로 인하여 발생할 수 있는 문제로부터 시스템을 보호하고 클라이언트에게 지속적인 서비스를 제공하기 위하여 두가지 방법을 사용한다. 첫째, 응용프로그램 수행 중 발생하는 오류가 시스템에 영향을 주는 것을 막기 위하여 디스패처가 수행되는 프로세스와 응용프로그램을 수행하는 프로세스를 분리시키는 방법을 제안한다. 둘째, 타임아웃 정보를 사용하여 응용프로그램 오류가 발생한 경우를 알아내고 오류 발생시 적절히 대처할 수 있는 방법을 제안한다. 이러한 방법들은 4.2 절과 4.3 절에서 자세히 설명한다.

4.2. 디스패처와 응용프로그램 수행 프로세스의 분리

본 절에서는 디스패처에서 수행을 제공하는 응용프로그램의 오류로 인하여 전체 시스템이 영향 받는 것을 방지하기 위한 방법을 설명한다. 디스패처 내에서 응용프로그램 수행 오류로 인해 발생하는 잘못된 동작이 시스템에 영향을 주는 것을 방지하기 위하여 응용프로그램을 수행하는 프로세스와 디스패처가 동작하는 프로세스를 분리한다.

분리된 프로세스에서 응용프로그램을 수행하는 것은 프로세스 생성 오버헤드(overhead)와 안정적인 디스패처 수행을 고려하여 사용자의 선택에 디스패처와 분리된 프로세스에서 수행할 수 있도록 하고 이러한 방법을 통하여 응용프로그램 오류로부터 시스템을 보호하도록 한다.

4.3. Timeout 정보를 사용한 응용프로그램 오류 처리 방법

본 절에서는 개발자가 작성한 응용프로그램이 디스패처에서 수행되는 도중에 오류로 인하여 중단되었을 때의 처리 방법에 대해서 설명한다. 본 논문의 디스패처에서는 응용프로그램 오류 발생 여부를 판단하기 위하여 타임아웃 정보를 사용한다. 타임아웃이란 어떠한 사건 a 가 발생하고 난 뒤 일정 시간 t 이내에 또 다른 사건 b 가 발생해야 하는 조건 하에 b 가 시간 t 이내에 발생하지 않은 상태를 말한다.

앞의 2.1 절과 4.3 절에서 설명한 바와 같이 디스패처는 멀티 프로세스로 응용프로그램을 수행하기 때문에 수행 상태를 쉽게 알수 없다. 따라서 디스패처는 응용프로그램의 오류 발생 여부를 판단하기 위하여 수행시간에 대한 타임아웃 방법을 채택하였다.

본 논문에서 구현한 시스템은 타임아웃 정보를 사용하여 응용프로그램 오류 발생 여부를 정확히 판단할 수 있도록 하고 오류가 발생하면 수행을 요청한 클라이언트에게 오류 메시지를 보내고 응용프로그램을 수행중이던 프로세스에 대하여 필요한 조치를 수행하여 시스템이 안정적으로 운용되도록 한다.

5. 결론

날로 발전하는 컴퓨팅 자원과 인터넷의 보급에 따라 분산된 컴퓨팅 자원을 효과적으로 활용하기 위하여 RPC, CORBA, EJB, DCOM 과 같은 다양한 분산 응용프로그램 개발 표준이 등장하였다. 그리고 이러한 표준들 간의 상호 호출이 가능하도록 하기 위하여 W3C 에서는 XML 기반의 분산 응용프로그램 호출 표준인 SOAP 을 제정하였다. SOAP 은 HTTP 로 전송되는 XML 기반의 통신규약으로 응용프로그램 개발 언어와 플랫폼에 독립적으로 동작할 수 있는 특징을 가지고 있어서 기존의 분산 응용프로그램 개발 표준과 상용 시스템에서 널리 채택되고 있다.

본 논문에서는 SOAP 지원 추세에 따라 SOAP 에 기반한 응용프로그램 디스패처를 설계하고 구현한다. SOAP 기반 응용프로그램 디스패처는 개발자가 작성한 응용프로그램을 저장, 관리하면서 클라이언트로부터 응용프로그램 수행 요청을 받아 그에 맞는 응용프로그램을 분기시켜 수행하고 결과를 돌려주는 기능을 제공한다. 이러한 기능을 통하여 응용프로그램 디스패처는 분산 응용프로그램을 개발하고 운용하기 위한 플랫폼의 핵심 모듈로 사용된다.

본 논문에서 설계하고 구현한 SOAP 기반 응용프로그램 디스패처의 특징은 다음과 같이 요약될 수 있다. 첫째 C, C++, Java 와 같은 다양한 응용프로그램 수행을 지원함으로써 분산 응용프로그램 개발을 위한 핵심 플랫폼으로서 사용될 수 있다. 둘째 디스패처 안에서 응용프로그램의 오류로 인하여 발생될 수 있는 문제에 대처하여 안정적인 수행을 제공한다. 셋째 SOAP 표준에 따라 클라이언트의 응용프로그램 수행 요청을 처리하여 기존의 다른 분산 응용프로그램 표준과 상호 호출이 가능하다.

참고 문헌

[1] Object Management Group, A Discussion of the Object Management Architecture Guide, Techinical Report formal/00-06-41, Object Management Group, Aug. 1997.

[2] Stevens, W. R., *UNIX Network Programming*, Prentice Hall, 2nd Ed., 1999.

[3] Object Managemenet Group, The Common Object Request Broker : Architecture and Specification, Object Management Group, July 2002.

[4] DeMechal, G. L., Yacinalp, U. L., and Krishnan, S., EnterpriseJavaBeans Specification, V2.0, SUN, Aug. 2001.

[5] Box, D. et al.,Simple Object Access Protocol(SOAP) V1.1, W3C, May 2000.

[6] Winer, D., XML-RPC Specification, Userland Software, June 1999.

[7] Gilman, J., "OMG Members Meet to Advance Integration Standards, SOAP Technology," OMG Articles, Object Management Group, Mar. 2002.

[8] Kassem, K., Vijendran, A., and Mordani, R., SOAP with Attachments API for Java(SAAJ) V1.1, SUN, June 2002.

[9] WebLogic, BEA WebLogic Server Programming WebLogic Web Services, BEA WebLogic Server Document V6.1, BEA, June 2002.

[10] Ewald, T., "Understanding XML Web Services The Web Services Idea," Microsoft Corporation Articles, Microsoft Co., Sept. 2002.

[11] Simon, H., *Strategic Analysis of XML for Web Application Development*, Computer Research, 2000.

[12] Apparao, V. et al., Document Object Model Level 1, W3C, Oct. 1998.