다양한 서비스 유형을 지원하는 CORBA와 XML의 연동구조

(XML and CORBA Integration Architecture for Supporting Various Service Types)

이 호 섭 † 홍 충 선 ††

(Ho-Seob Lee) (Choong-Seon Hong)

요 약 오늘날 XML은 구조적인 데이타를 표현하는 수단으로서 널리 쓰이고, 웹 환경에서 그 중요성이 더욱 확대되고 있다. 이러한 XML 기반의 구조적인 데이타를 처리하기 위해서 분산처리환경인 CORBA 환경에서는 XML문서를 주고받는 것이 보편화되고 있다. 그러나 XML문서를 처리하기 위해서 CORBA 환경을 수정해야만 한다. 즉 IDL정의, CORBA 클라이언트, CORBA 서버, 그리고 구현객체까지도 XML문서를 처리하기 위해 수정을 해야만 하거나 설계 단계부터 XML문서를 처리할 수 있는 시스템이 요구된다. 본 논문에서 제안하는 연동 구조는 기존의 CORBA 환경의 수정사항 없이 웹 환경과 CORBA 환경사이에 연동을 지원한다. 본 제안구조는 다양한 서비스유형 즉, 서블릿을 이용한 웹응용서버와 SOAP을 사용하여 기존 시스템과 CORBA 환경과의 확장성을 제공하고 있다. 따라서 본 논문에서 제안하는 구조는 XML과 CORBA의 연동을 위해 CORBA 환경의 수정을 하지 않고 유연하게 확장할 수 있다. 따라서, 본 구조를 채용한 웹 플랫폼환경에서는 CORBA환경과의 연동을 용이하게 제공할 수 있다.

키워드 : CORBA, XML, XML-RPC, SOAP, DOM, XML Schema, DTD, 서블릿, 애플릿, CGI, 메타데이타

Abstract Today, XML is widely used to present structured data and its importance has been expanded much in the web environment. Generally, we exchange XML documents to process XML-based data in distributed processing environments like a CORBA. But we need to configure CORBA environments to process XML documents. The system that has initially been designed to process XML is required. If there is no such system, we should configure most parts of the system, namely, IDL definition and CORBA client, CORBA server, implementation objects.

In this paper, we propose a system that can apply an integration structure of web environment and CORBA without any additional configuration. Our proposed system can support various service types, namely, Also it can offer extensibility to legacy system with CORBA by using servlet and SOAP. Our proposed system can extend the structure to integrate XML with CORBA, Hence the web platform environment that applies this architecture can support much easier integration with CORBA environment.

Key words : CORBA, XML, XML-RPC, SOAP, DOM, XML Schema, DTD, Servlet, Applet, CGI Metadata

1. 서 론

인터넷을 이용한 정보 습득에 있어서 사용자가 다른 프로그램을 설치하지 않아도 웹 브라우저만으로 서비스를 이용할 수 있고 플랫폼에 상관없이 정보의 공유가 용이하다. 또, 분산 환경 하에서 어플리케이션간 통신 수단으로써 웹을 이용하고자 하는 흐름은 XML(eXtensible 후에 보다 가속화되고 있다.

현재 가장 널리 사용되고 있는 분산 컴포넌트 기술은

· 본 연구는 한국과학재단 목적기초연구(R05-2001-000-00976-0) 지원으로 수행되었음.

† 비 회 원 : 경희대학교 전자정보학부
hslee@networking.kyunghee.ac.kr
†† 종신회원 : 경희대학교 전자정보학부 교수
cshong@khu.ac.kr

논문접수 : 2001년 12월 17일
심사완료 : 2002년 7월 2일

OMG(Object Management Group)의 CORBA(Common Object Request Broker Architecture)[2], 마이크로소프트의 COM+, Java Beans 등이 존재한다. 본 논문에서는 OMG의 CORBA와 최근 웹의 표준으로 자리잡고있는 XML과의 상호연동을 위해서 DOM(Document Object Model)[3,4]과 SAX(Simple API for XML)표준 API를 사용하였다. 컴포넌트 기술과 웹과의 호환성을 직접 가지지 못하기 때문에 웹을 통해 데이타 교환에 사용하는 어플리케이션에서는 별도의 프로토콜을 지원하도록 요구받게 된다[5].

이러한 컴포넌트 기술과 웹과의 호환성 문제를 해결하기 위해서 XML을 이용하고자 하는 많은 노력이 있어왔다. XML은 플랫폼 및 프로그래밍 언어에 독립적이고, 확장 가능하며, 그 자체만으로도 데이타와 메시지 전달을 하는데 있어 표현 수단으로 사용될 수 있다. 또한 파싱 이벤트를 애플리케이션의 콜백(callback)을 통해 직접 알려주는 이벤트 기반의 SAX와 XML문서나 DTD를 파싱해 내부 트리로 만들며, 애플리케이션은 이 트리를 탐색 조작하는 식으로 작업을 수행하게 하는 객체모델 기반의 DOM과 같은 표준 API를 통한 접근 방법을 제공하고, XML 스키마와 같은 것들을 통해서 데이타 유형을 확장시킬 수 있으며, 특히 HTTP 프로토콜을 이용함으로써 웹 환경에 곧바로 이식할 수 있는 특징을 갖고, 방화벽 때문에 발생된 분산컴포넌트에 접근할 수 없는 문제도 HTTP프로토콜을 이용함으로써 해결하였다. 또, 웹서버 기술은 Server-Side-애플릿인 서블릿을 사용하였다. 서블릿은 동적인 웹 컨텐츠 개발의 한 기술로 자바의 모든 특징을 상속한 자바의 확장 기술이다. 뛰어난 이식성과, 웹 서버의 최적의 성능을 유지하고 또한 웹 서버의 유지보수가 쉽다는 장점을 가지고 있다[5].

본 고에서는 다양한 서비스유형을 지원하는 CORBA와 XML의 연동구조를 제안한다. 2장에서는 최근의 CORBA와 XML의 상호연동에 관한 연구들에 대해서 알아보고, 3장에서는 본 논문에서 제안하는 CORBA와 XML 연동구조를 설명, 4장에서는 본 구조의 구현과 시뮬레이션에 대하여 설명하고, 5장에서는 결론에 대하여 기술한다.

2. CORBA와 XML의 상호연동의 최근 동향

CORBA와 XML의 연동에 관한 연구가 활발히 진행되어왔고 현재도 많이 진행되고 있으며, 이는 OMG와 여러 단체들이 주도가 되고 있다. 특히 OMG에서 제안한 방법들이 연구되어져 왔다.

표 1은 분산환경 기반의 CORBA와 인터넷 기반의 XML의 전반적으로 비교한 것이다[6].

먼저 XML문서의 CORBA 환경에 접근 방법적인 측면을 보면, OMG에서 제안한 방법에는 동적 접근과 정적 접근의 2가지 시나리오가 있다[7]. 이 두 가지 방법들은 CORBA 환경에서 XML 문서를 String으로 전송하는 것이 가능하지만, 이 XML 문서에 대해 전송하기전과 전송한 후에 XML 문법 구조를 분석을 해야만 하기 때문에 이러한 것이 미리 해결하고자 XML 문서를 메모리에서 조작할 수 있는 데이타 구조를 생성하여 송신측이나 수신측에서 처리과정을 거치지 않고, XML 문서를 전달하는 것이다. 이 두 가지 방법은 XML 문서로부터 XML DTD에 기반을 둔 IDL Value까지도 매핑을 제공한다.

우선 동적인 시나리오는 XML 문서에서 발견된 XML 엘리먼트의 뜻이 정의되지 않았을 때의 XML문서의 처리과정이다. 이 경우 단지 최소한의 정보가 알려져 있

표 1 CORBA와 XML의 비교

특징	CORBA	XML
운영체제 독립성	Yes	Yes
언어 독립성	Yes	Yes
지원 데이타 형식	String, Int, Boolean...more	Only String
동기적인 호출	Yes	No
비동기적인 호출	Yes	Yes
반환 값	Yes	No
예외 처리	Yes	No
위치 투명성	Yes using naming service	Yes, using other Web protocols
마샬링(marshalling)	Yes	Only possible to check data against DTD
사람 인식성	No	Yes
네트워크 성능	optimized	New connection for each message
범위성(Scalability)	Yes, but no to the internet	To the internet
방화벽	No, Use Tunnels	Yes

다. 이 정보는 DOM에 의해서 나타내어진다. DOM은 XML문서의 완전한 contents 표현의 표준이다.

DOM은 XML 문서를 제어하는 메카니즘을 제공하기 위한 W3C XML Information Set의 요구를 만족시키고, IDL Valuetype에 대한 DOM표현에 의해 CORBA 구현은 유용해지고, 표준화되며, XML문서의 모든 정보에 직접 제어하게 된다.

두 번째 시나리오인 정적인 시나리오는 정적인 정보가 XML DTDs와 XML 스키마로부터 나타나는 곳에 첫 번째로 만들어진다. DTDs/스키마는 XML문서에 나타날 정보의 종류와 부합되는 Valuetype 생성에 사용되는 메타데이타(metadata)이다. DTDs/스키마와 Valuetype으로부터 생긴 메타데이타는 OMG 표준에 의한 분산된 메타데이타를 제공하는 CORBA 인터페이스 저장소와 Meta Object Facility에 보내진다[7].

지난 몇 년간 CORBA와 XML의 상호 연동을 위해서 많은 연구가 이루어져 왔다. 다음은 CORBA와 XML의 연동방법을 나열한 것이다.

1) CGI, 서블릿, Web API, 애플릿 등 웹 기술의 이용[1]
2) 메타데이타(Metadata)의 이용[7,8]
3) SOAP(Simple Object Access Protocol)의 이용[9,10,11]

2.1 CGI, 서블릿, Web API, 애플릿 등 웹 기술의 이용

우선 CGI, 서블릿, Web API, 애플릿 등 웹 기술을 사용한 CORBA와 XML연동 방법은 서버측 구동 방법인 ISAPI, NSAPI, CGI, 서블릿 등의 Web API를 이용하는 방법이 있고, 클라이언트측 구동 방법인 애플릿을 사용하는 방법으로 구분한다. 서버측 구동 방법은 웹 서버측에서 CORBA 클라이언트 객체를 활성화한 후, CORBA 클라이언트가 IIOP 프로토콜을 사용하여 CORBA 서버 객체에 요청을 한다. 클라이언트측 구동 방법은 클라이언트가 웹 브라우져를 통해 웹 서버객체에 접속해서 CORBA 모듈을 포함한 모듈을 다운받아 웹 클라이언트측에서 CORBA 클라이언트를 실행시켜 웹 클라이언트에서 IIOP를 통해서 CORBA 서버 객체와 통신을 할 수 있는 방법이다.

또한 이 경우 XML과 CORBA의 연동에서 XML문서의 처리방법은 CORBA 클라이언트와 CORBA 서버측에서 XML 문서를 직렬화 형태로 전송하여 CORBA 클라이언트와 서버측에서 이 XML문서를 처리하는 방법과, CORBA 클라이언트가 CORBA 서버측에 요청하여 받은 응답을 XML로 변형하는 방법이 있다. 그러나, 이 구조는 기존의 CORBA환경에서 XML문서를 처리해야 하기 때문에 CORBA 클라이언트객체와, CORBA 서버 객체, IDL 등을 수정해야만 한다.

그림 1은 JVM(Java virtual machine)기반의 Web API를 이용한 경우의 CORBA와 XML의 연동구조를 나타내고 있다[1].

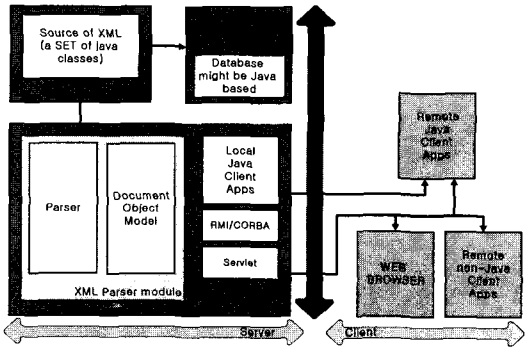


그림 1 Web API를 이용한 CORBA와 XML의 연동구조

2.2 Metadata의 이용

두 번째 방법은 메타데이타를 이용한 CORBA와 XML연동 방법이다. 메타데이타에는 XMI(XML Metadata Interchange), W3C의 RDF(Resource Description Framework), OMG의 MOF(Meta Object Facility), SMIF(Stream_based Model Interchange Format) 등이 있다. UML기반의 메타모델을 정의하여 분산 환경에서 메타데이타의 교환 및 전송을 위한 XML형식의 메타데이타를 생성하여 데이타베이스에 저장한다. 이 XML형식의 메타데이타를 이용하는 것은 인터넷상에서 자료전달과 교환시 이진(Binary) 양식이 아닌 텍스트 양식으로 전달이 가능하므로 플랫폼에 따라 발생할 수 있는 호환성 문제를 해결할 수 있기 때문이다. 그러나 이 구조는 서로 다른 메타데이타들간의 호환성에 문제가 제기 되고 있다. 그림 2는 메타데이타의 관리 구조에 대한 그림이다[7,8].

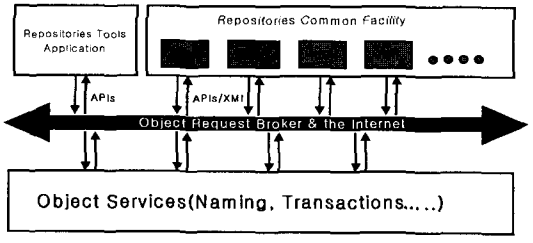


그림 2 Metadata의 관리구조

2.3 SOAP의 이용

마지막으로 SOAP(Simple Object Access Protocol) 프로토콜을 이용한 CORBA와 XML연동 방법이다. SOAP은 W3C의 표준으로서, 플랫폼에 독립적으로 인터넷상에 분산되어 있는 서비스, 객체, 또는 서버에 접근할 수 있도록 HTTP 프로토콜을 지원하는 XML프로토콜의 한 예이다. 그림 3은 CORBA와 SOAP을 이용한 구조에 대한 것이다[9,10].

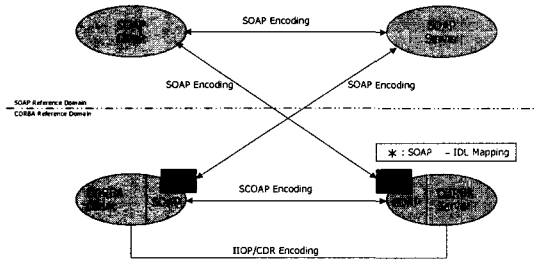


그림 3 CORBA간 연동을 위한 SOAP의 사용 구조

이 경우에서 CORBA+SOAP 클라이언트와 CORBA+SOAP 서버사이에 SOAP 부호화를 통한 접근에서는 SOAP/HTTP로 IIOP의 터널링 효과를 가능하게 하고, IDL형을 위해 XML 매핑이 필요하게 된다. 또, 방화벽이 존재할 경우 SOAP를 통해 ORB가 인터넷을 통과할 수 있다. SOAP 클라이언트와 CORBA+SOAP 서버사이에 SOAP 부호화를 통한 접근에서는 SOAP 클라이언트를 인식하지 못하기 때문에 CORBA서버가 SOAP를 이용해서 인식할 수 있도록 해서 SOAP 클라이언트가 호출할 수 있도록 해야한다. 이때 SOAP 형과 IDL 형의 매핑이 필요하게 된다. 이 SOAP방식으로 구현된 것이 Rogue Wave's사의 XORBA와 CapeClear사의 CapeConnect Tree이 있다.

본 논문에서는 위의 3가지 방법 중에서 첫 번째 방법 중 서블릿을 이용한 방법을 채택하였다. 그 이유는 서블릿은 최초 접속자에 의해서 서블릿이 실행되면 이후의 접속자는 쓰레드 단위로 처리되어 서버의 오버헤드가 감소되고, 클라이언트가 증가하더라도 시스템 성능의 저하현상이 적으며, 기존의 자바 API를 모두 사용할 수 있고, 애플릿과의 연동으로 네트워크의 부하를 감소시킬 수 있기 때문이다. 그리고 아래에서 해결해야될 여러 가지 문제점을 해결하기 위해서 웹응용서버를 두고, SOAP를 같이 사용하였다.

CGI 방식은 웹 환경의 사용자들이 정적인 HTML자료에서 동적인 자료 생성을 요구하게 된 후, 이를 가능하게 한 것이 바로 CGI이다. 그러나 CGI는 낮은 수행능력과 하나의 독립적인 프로세스이기 때문에 오버헤드가 너무 크고, 각 CGI 프로그램이 별개의 프로세스로 시작되기 때문에 실행중인 프로그램들 사이의 통신을 위한 채널을 만들기 힘들고, CGI는 서버측에 많은 부하를 가져올 수 있다. 또한 HTTP 기반의 통신 규약은 매번 접속이 새로 이루어져 세션 정보나 트랜잭션 정보를 관리할 수 없게 되어 궁극적으로 어플리케이션에 많은 보안상의 제약을 주게 된다.

애플릿은 CORBA 모듈을 포함한 모듈의 다운로드로 인한 서버측의 부하가 많은 문제, 보안에 대한 취약점, 네트워크의 오버헤드를 가져올 수 있다.

이러한 웹 기술들을 이용하는 것은 수십 명의 요구사항을 처리하는 인트라넷 시스템의 경우에는 문제가 없을지 모르지만 수백, 수천 명이 사용하는 시스템을 구성하고 수십만에 달하는 외부고객의 요구를 처리하고 거래처 수백 곳의 공급선을 관리해야 하는 경우 엑스트라넷의 성능의 문제에 직면하게 될 것이다. 이외에도 웹 서버와 미들웨어, DBMS와의 연동에 있어서 단순한 데이타 전달 기능만을 제공하고 보안이나 트랜잭션, 로드 밸런싱과 같은 기능을 수행하지 못한다면 시스템의 효율성을 장담할 수 없게 된다.

메타데이타를 이용하는 구조는 서로 다른 메타데이타들 간의 호환성에 문제가 제기 되고 있고, 이 메타데이타를 관리하는 프로그램도 모든 메타데이타를 지원해야 되기 때문에 프로그램이 커질 수 있다. 메타데이타를 지원하는 API개발도 시급하다. 또 메타데이타를 관리하는 방법과 메타데이타를 생성하는 방법에 대해서도 아직 표준화 작업이 이루어지지 않고 있다.

SOAP을 이용한 구조는 클라이언트와 서버의 메시지는 XML형식의 메시지를 사용하게 됨으로써 서로 다른 플랫폼기반의 다른 언어로 구현된 서로 다른 컴포넌트 모델을 통합하거나 연동할 수 있는 이점을 제공하지만 XML을 사용함으로써 기존의 분산환경에서 제공하는 각종 서비스를 제공하지 못하며 각각의 분산컴포넌트들의 Fault가 발생했을 때 근본적인 해결방안을 제공하지 않는다. 그리고 모든 플랫폼을 지원을 하지만 각각의 플랫폼을 지원하는 API들의 표준화가 시급하다.

3. 제안연동구조

3.1 웹응용서버와 SOAP를 이용한 제안연동구조

본 논문에서 제안하는 CORBA와 XML의 연동을 위한 웹응용서버구조를 기술한다.

본 구조에서는 서로 다른 서비스를 제공하기 위해서

웹 기술인 서블릿과 XML-RPC 기술인 SOAP를 사용하였다. 서블릿을 사용함으로써 Web서버는 쓰레드 단위로 처리를 하기 때문에 초기화에 따른 오버헤드가 없고, 클라이언트가 증가하더라도 시스템의 성능의 감소하지 않는다. 기존 Java API를 모두 사용할 수 가 있고, 자바 애플릿과 연동이 가능하고, HTTP의 비연결형 특성에 의한 네트워크 오버헤드를 줄일 수가 있고, SOAP를 사용함으로써 플랫폼에 독립적으로 인터넷 상의 분산되어 있는 서비스, 객체, 또는 서버에 접근 할 수 있다.

CORBA와 XML연동구조의 구성은 Web 클라이언트, 웹응용서버, CORBA 객체, SOAP 클라이언트, SOAP 서버 모두 다섯 가지 구성요소로 구성되어 있다. 이 구조에서는 서로 다른 서비스를 위해 웹 클라이언트, 웹응용서버, CORBA객체로 이루어지는 서비스와 웹클라이언트, SOAP 클라이언트와, SOAP 서버객체, CORBA 객체로 이루어지는 서비스로 이루어진다.

우선 웹응용서버를 이용한 서비스제공은 웹 클라이언트로부터 HTTP를 통해서 요청값을 In_Value모듈을 통해서 받고, 받은 값을 CXP(Call XML Parser)를 통해서 요청된 값들의 타당성 검사(Validity Check)를 위해 CDT(Create DOM Tree)와 CXD(Create XML Document)에게 전달된다. CDT에서는 요청된 값의 타당성 검사를 위해 DTD 문서와 JDOM API를 사용해서 DOM Tree형태로 만들어진 다음, CXD로 전달되어 XML문서로 만들어진다. 이 만들어진 XML문서는 JDOM API와 DTD를 사용해서 입력받은 값이 DTD에 설정해 놓은 것과 같은지를 검사하게 된다. 이것이 타당성 검사이다. 타당성 검사를 하고 나면 다시 CXP로 결과를 전달하고, 결과값이 이상이 없으면 CCC(Call CORBA 클라이언트)모듈을 통해서 CORBA클라이언트 객체를 호출하게 된다. 이 호출된 CORBA 클라이언트 객체는 Set_CoVal를 통해서 CCC에서 호출할 때의 요청 값들을 CORBA클라이언트 객체로 가져오게 된다.

CORBA 클라이언트 객체는 CORBA 서버 객체에게 서비스를 요청을 하고, 이에 따른 결과 값을 받는다. 결과 값을 받은 CORBA 클라이언트 객체는 결과를 보여주기 위해 Out_Doc를 호출하고 이 Out_Doc는 CORBA 클라이언트 객체에 받은 결과값을 Set-Val을 통해서 받아 최종 결과를 XML 문서로 작성한다음 XSL 스키마 파일을 참조해서 클라이언트에게 전달되게 된다.

두 번째 SOAP를 이용한 서비스는 웹 클라이언트의 요청값을 SOAP 클라이언트에 전달하며, 전달 받은 요청값들은 SOAP 메시지로 직렬화 되어, SOAP 서버에게 전달되어진다. SOAP 서버는 메시지를 파싱하여 유효성 검증을 마친후, 해당되는 메소드를 SOAP Mapper 객체를 통하여 검색한 후 웹응용서버의 CORBA 클라이언트로 전달되어진다. CORBA 클라이언트는 CORBA 서버객체에게 해당 서비스를 요청을 하게 되고, CORBA 서버객체로부터 응답을 받은 CORBA 클라이언트는 SOAP 서버에게 전달된 후, 해당 오브젝트로부터 받은 결과 값은 다시 SOAP Mapper를 통하여 SOAP 메시지로 직렬화한 후, SOAP 클라이언트에게 전달되어진다. 전달된 결과값은 SOAP 클라이언트의 리더를 통하여 메시지를 해석한 후 결과 값을 웹 클라이언트에 전달되는 방법을 취하고 있다.

본 논문에서 제안한 CORBA와 XML연동 구조와 기존 연동구조들과 비교를 하면, 우선 그림 1의 JVM기반의 Web API를 이용한 구조는 CORBA 환경의 모든 객체들의 수정이 불가피하다. 즉 CORBA 클라이언트들과 CORBA 서버객체들 사이에 XML문서를 교환하여 처리를 하는 방식이어서 CORBA 클라이언트와 서버객체가 직렬화된 XML 문서를 받았을 경우 이것을 파싱하여 처리를 해야하기 때문에 기존에 존재하고 있는 CORBA 객체들을 다시 수정을 해야한다. 본 논문의 Service A의 구조는 기존의 CORBA 환경의 수정없이 CORBA 클라이언트와 연동하는 서블릿 기반의 객체를 위치시킴으로써 기존 CORBA 객체들이 XML문서를 전달받을 경우 서블릿 기반의 객체가 파싱을 하여 CORBA 환경에 요청값을 전달하고, CORBA 환경에서 받은 결과 값을 XML문서 변형하여 되돌려 줄 수 있다. 즉, CORBA 환경에 XML문서를 처리하고 생성하는 객체를 둠으로서 기존의 CORBA 환경의 수정없이 XML과 연동할 수 있다.

또, 그림 3의 CORBA간 연동을 위한 SOAP의 사용 구조에서는 CORBA+SOAP 클라이언트와 SOAP+CORBA 서버사이에 SOAP의 부호화를 통한 접근으로 IIOP의 터널링 효과를 가능하게 하고, 또 IDL형을 위해 XML 매핑이 필요하게 된다. 방화벽이 존재할 경우 SOAP를 통해 ORB가 인터넷을 통과 할 수 있다. SOAP 클라이언트와 SOAP+CORBA 서버사이에서는 SOAP의 부호화를 통한 접근에서 CORBA 서버객체가 SOAP 클라이언트를 인식하지 못하기 때문에 CORBA 서버가 SOAP서버를 이용해서 SOAP 클라이언트를 인식할 수 있도록 만들어서 SOAP 클라이언트가 CORBA 서버를 호출할 수 있도록 해야한다. 이때 SOAP 형과 IDL 형 사이에 매핑이 필요하게 된다. 그러나 이러한 구조도 기존의 CORBA 환경의 수정이 필요하게 된다. 즉, CORBA환경에서의 XML기반의 SOAP을 이용한

표 2 CORBA-XML 연동 구조의 비교

		장 점	단 점
Web API	애플릿	-CORBA모듈을 다운받기 때문에 어떤 플랫폼이라도 CORBA기반의 통신을 할 수 있음	-다운로드가 가장 큰 문제점
	CGI	-구현이 쉽고, 모든 언어로 구현 할 수 있음	-낮은 수행능력 -큰 오버헤드 -빈약한 CGI간 통신능력
	Servlet	-웹 서버의 부하를 쓰레드를 이용하여 줄였기 때문에 초기화에 따른 오버헤드가 없고, 사용자 수가 증가하더라도 시스템 성능이 비례적으로 감소하는 현상이 없음	-서블릿을 지원하는 웹서버가 많지 않음 -웹서버에서 서블릿을 인식할 수 있도록 하기 위한 서블릿 엔진이 필요함
	NSAPI	-네스케이프 웹서버 계열의 컴포넌트들과 연동이 수월함 -CGI에 비해 더욱 정교하고 빠른 애플리케이션을 작성할 수 있음	-표준이 아니기 때문에 웹서버가 바뀔 경우 구축된 프로그램을 재구축 해야함 -CORBA 모듈의 지원이 약함 -플랫폼에 의존적임
	ISAPI	-윈도우즈 계열의 컴포넌트들과 연동이 수월함 -CGI보다 더 빠르게 실행되는 웹 서버 프로그램을 작성할 수 있음	-표준이 아니기 때문에 웹서버가 바뀔 경우 구축된 프로그램을 재구축 해야함 -플랫폼에 의존적임
Metadata		-모든 객체를 관리하기가 편함	-메타모델간의 이식성 문제
SOAP		-모든 플랫폼에서 모든 분산객체들과 연동 할 수 있음	-단순히 텍스트형식의 멧세지만 주고 받을 뿐 자체적으로 서비스를 제공하지 못하고, Fault 처리를 통보하지만 처리를 하지는 못함 -API를 표준화하는 것이 시급함
제안구조		-확장이 쉬움 -무든 플랫폼에서 모든 분산객체들과 연동할 수 있음 -웹 서버의 부하를 줄였고, 사용자의 수가 증가하더라도 시스템 성능이 비례적으로 감소하는 현상이 없음	-서블릿을 지원하는 웹서버가 많지 않아서 웹서버에서 서블릿을 인식할 수 있도록 하기 위한 서블릿 엔진이 필요함

것이기 때문이다. 다른 분산환경과의 연동이 필요한 경우도 다시 수정해야하는 점이 있다. 본 논문의 Service B의 구조는 기존의 CORBA 환경의 수정사항 없이 SOAP 클라이언트와 SOAP 서버를 이용한 구조이다. CORBA 클라이언트와 CORBA 서버사이에서 SOAP를 이용하는 것이 아니라, SOAP을 이용해서 CORBA 환경 즉, CORBA 클라이언트와 통신을 하는 구조이다. CORBA 환경이 아니어도 CORBA 환경의 서비스를 제공받을 수 있다. 또 다른 분산환경과의 연동이 필요한 경우에도 별다른 수정사항 없이 추가할 수 있다.

또, 기존의 분산객체들과 통신에서 방화벽 때문에 야기되는 문제 즉 네트워크 상에 분산되어 있는 CORBA, COM+, Java Beans 객체등에 접근할 수 없었던 문제를 HTTP프로토콜을 사용해서 해결하였습니다. 방화벽은 웹 서비스를 위해서 일정한 포트만을 제외하고 나머지 포트를 이용할 수 없기 때문이다. 웹 클라이언트가 웹응용서버에게 HTTP프로토콜을 사용해서 서비스를 요청하면, 웹응용서버에서 관리하고 있는 해당 서비스를 제공하기 위해 CORBA 객체를 호출하게 되는 것이다. 따라서 방화벽 때문에 접근할 수 없었던 분산객체들에게도 접근하여 서비스를 받을 수 있다. 표 2는 기존 방식과 제안 방식의 CORBA-XML연동 구조의 장단점을 비교한 것이다.

3.2 제안연동구조의 정보모델 및 운용시나리오

그림 4는 본 구조의 간략한 클래스다이어그램을 나타내고 있다.

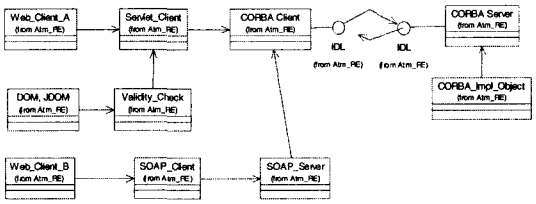


그림 4 본 구조의 ClassDiagram

본 구조에서는 요청값을 받아서 처리하는 구조와 XML문서를 전송하기 위해 SOAP를 이용한 구조로 나뉘어 있으며, CORBA 클라이언트를 호출하는 과정은 같다.

그림 5는 CORBA와 XML의 연동구조를 나타낸 것이며, Service A와 Service B로 구분을 해 놓았다. 웹응용서버와 SOAP서버는 동일 시스템 내에 존재한다.

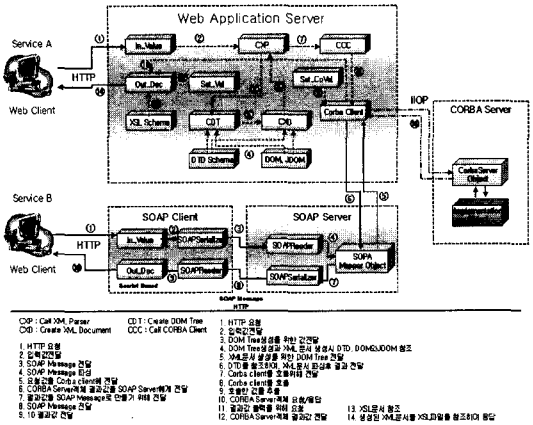


그림 5 분산환경에서 Web기반의 CORBA와 XML의 연동 구조

그림 6은 웹 클라이언트가 요청값만을 HTTP를 이용해서 요청하는 경우의 운용시나리오를 나타낸 것이다. 이 과정에서는 서블릿 클래스에서 값을 받아서 DTD에 기반한 타당성을 검사를 하기 위해서 XML 문서를 작성한다.

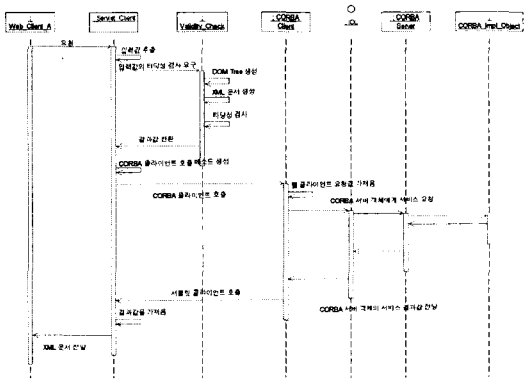


그림 6 Service A의 SequenceDiagram

그림 7은 웹 클라이언트가 요청값만을 HTTP를 이용해서 요청시 이 요청값을 XML문서를 전송하기 위해 SOAP를 이용하였으며, XML 문서를 SOAP 멧시지로 변경하여 전송한 후, SOAP 메시지를 파싱하여 유효성을 검증한다. 이 후의 과정은 서비스 A과정과 중복된다. 본 논문에서는 같은 시스템에 존재하지만 SOAP 클라이언트와 SOAP 서버는 꼭 같은 시스템에 존재하지 않아도 된다.

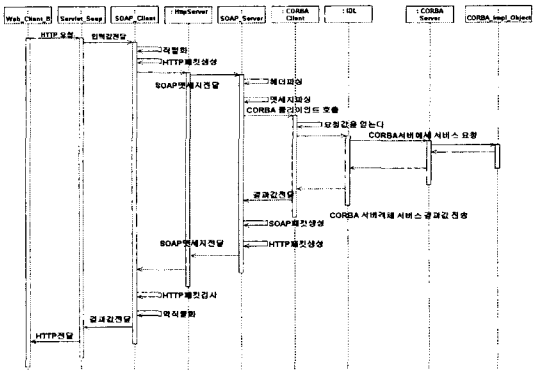


그림 7 Service B의 SequenceDiagram

4. 제안연동구조의 웹응용서버의 구현 및 성능분석

4.1 제안연동구조의 구현

본 논문에서 제안한 연동구조를 가지고 ATM QoS 파라미터를 전송해서 연결을 설정하는 웹응용서버를 구성하였다. Web 클라이언트에서 요청한 ATM QoS 파라미터값들을 웹응용서버에서 작성한 후 DTD문서를 참조하여 타당성을 검사하게 된다.

타당성이 검사되면, 이 파라미터값들을 CORBA 클라이언트를 호출을 하게 되고, CORBA 클라이언트에서는 이 파라미터값들을 가지고 와서 CORBA 서버에게 요청할 ATM QoS 파라미터를 추출하게 된다. CORBA 서버에서 이 파라미터들을 받아서 연결을 설정해서 CORBA 클라이언트에게 전송하게 된다.

CORBA 클라이언트에서는 받은 결과 값을 가지고 다시 서블릿 클래스에 전송을 하고 서블릿 클래스에서는 XML 문서를 만들어서 웹 클라이언트에 전송을 하게 되는데, 이때 XSL 스키마파일을 참조하여 요청한 ATM 연결에 대한 결과를 보여주게 된다.

아래 표 3는 CORBA IDL구조이다.

표 3 CORBA 환경에서의 IDL 구조

```
//Atm.idl
module Atm {
    interface AtmObject {
            string startpoint(in string startpoint);
            string endpoint(in string endpoint);
            long bandwidth(in long bandwidth);
            long backup(in long backup);
            long servicetype(in long servicetype);
            boolean connect(in boolean connect);
    };
};
```

아래 표 4은 XML 파일이다.

표 4 웹 환경에서의 XML 구조

```
<?xml version="1.0" encoding="euc-kr"?>
<?xml-stylesheet type= "text/xsl" href= "atm.xsl"?>
<ATM>
    <ATMObject>
        <startpoint> default </startpoint>
        <endpoint> default  </endpoint>
        <bandwidth> default </bandwidth>
        <backup> default </backup>
        <trafficclass> default </trafficclass>
        <connect> default </connect>
    </ATMObject>
</ATM>
```

IDL파일은 CORBA 클라이언트가 CORBA 서버에 요청하는 값과 CORBA 서버가 CORBA 클라이언트에게 전달하는 값을 정의하며, 정의한 것은 시작 주소의 목적지 주소 대역폭, 패킷전달 방법, 서비스 종류 등을 정의한다.

XML파일도 IDL 파일과 같은 ELEMENT를 정의하고 있다.

4.2 OPNET을 이용한 웹응용서버의 성능분석 및 평가

본 절에서는 CORBA와 XML 연동을 위한 웹응용서버에서 제안한 서블릿구조를 가지고 본 시스템과 CGI (Common Gateway Interface)구조와, 애플릿을 사용한 구조에 대해서 큐 지연, 처리량(Throughput), 이용도(Utilization)을 측정 비교하였다.

먼저 서블릿 구조와 CGI구조, 애플릿 구조에 대한 큐의 지연, 처리량, 이용도의 측정 시뮬레이션은 OPNET 시뮬레이터를 사용하였고, 시뮬레이션 환경과, 모듈구성은 다음과 같다.

- 언어 : C++, Java
- 코바 제품 : VisiBroker for java 4.5, VisiBroker for C++ 4.51
- O/S : Windows 2000 Server
- CPU : Pentium II 350 Mhz
- RAM : 448 Mhz
- 시뮬레이션 시간 : 5분
- 클라이언트수 : 1000 user
- 패킷 생성 모듈 : Packet Interarrival Time : exponential(1.0)(s)
  Packet Size : exponential(1024)
  Start Time : 1.0(s)
- 큐 모델
  - ServiceRate
  - 서블릿 : 9,061 CGI : 5,094 애플릿 : 2,860
  - Server : 3개
  - Subqueue : 2개

비교대상인 서블릿, CGI는 서버 구동구조이며, 애플릿은 클라이언트 구동구조이다. 이 세 가지 구조의 ServiceRate는 한 개의 프로세스당 처리시간으로 하여 도출해낸 것이다.

본 시뮬레이션에서 ISAPI 구조는 마이크로소프트 플랫폼에서만 사용되고, NISAPI는 코바모듈의 지원이 약하기 때문에 제외하였으며, 큐의 지연과 분당 평균 큐 지연, 분당 처리량과 분당 평균 처리량을 가지고 있고, 분당 평균 이용도를 측정하였다.

그림 8~14에서 나타난 것과 같이 각각의 구조에 따라서 차이가 많음을 알 수 있다. 그림 8의 애플릿 구조는 클라이언트로 CORBA 모듈을 다운로드 하기 때문에 큐의 지연이 상당히 많음을 알 수가 있고, CGI구조도 애플릿구조 보다는 낮음을 알 수 있으나, 역시 지연이 많다. 그러나 서블릿구조는 큐의 지연이 안정적임을 알 수 있다.

그림 9, 10, 11는 웹클라이언트와 웹응용서버사이의 처리량, 초당 평균 처리량을 나타낸 것이다. 처리량은 초당 패킷의 처리량을 말하는 것이고, 초당 평균 처리량은 해당 초까지의 평균 패킷의 처리량을 나타내는 것이다. 그림 9의 애플릿구조 경우에는 초당 1~6개의 패킷

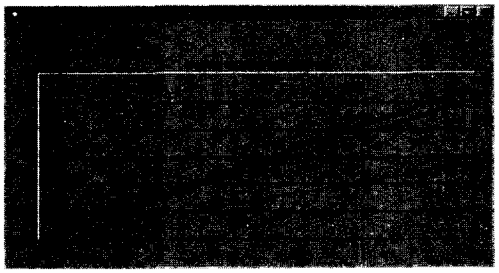


그림 8 애플릿, CGI, 서블릿 각 구조의 평균 지연

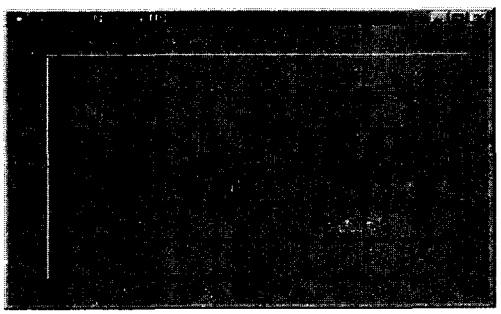


그림 9 애플릿을 이용한 구조의 처리량

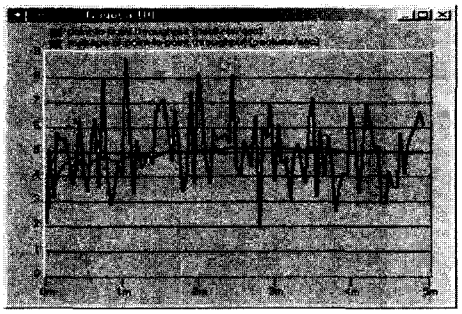


그림 10 CGI를 이용한 구조의 처리량

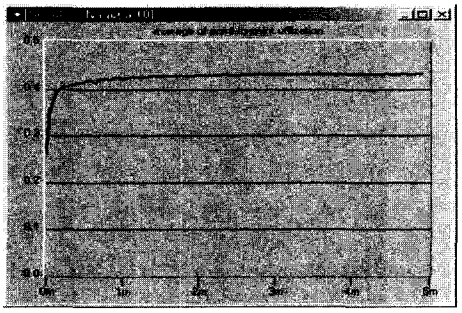


그림 11 애플릿을 이용한 구조의 이용도

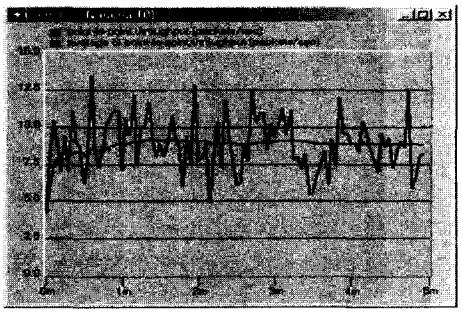


그림 12 서블릿을 이용한 구조의 처리량

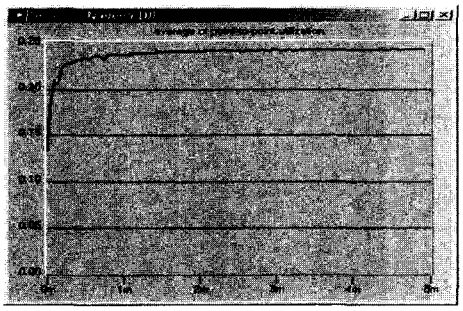


그림 13 CGI를 이용한 구조의 이용도

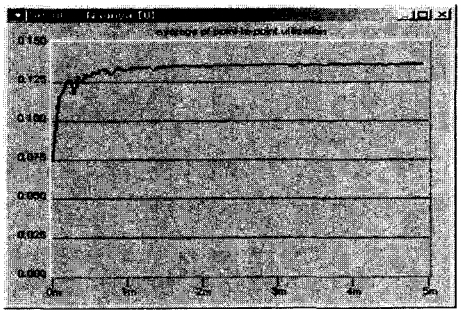


그림 14 서블릿을 이용한 구조의 이용도

을 처리하고 있고 평균 3개의 패킷을 처리하고 있음을 할 수가 있다. 그림 10의 CGI구조 경우에는 2~9개의 패킷을 처리하고 있고, 평균 5개의 패킷 처리량을 나타내고 있다. 본 논문에서 제안한 서블릿구조 경우에 그림 11을 보면 5~13개 이상의 패킷을 처리하고 있고, 평균 8개 이상의 패킷 처리량을 나타내고 있다.

그림 12, 13, 14은 웹클라이언트와 웹응용서버사이의 네트워크의 이용현황(Utilization)을 나타낸 것이다. 그림 12의 애플릿구조 경우에는 프로세스를 웹응용서버로 다운로드를 받기 때문에 이용현황이 CGI와 서블릿 보다 다소 높은 편을 보이고 있다. 그림 13의 CGI를 이용한 경우와 그림 14의 서블릿을 이용한 경우를 비교를 하면 CGI는 프로세스단위의 처리를 하고 서블릿은 쓰레드단위로 처리를 하기 때문에 네트워크 이용현황도 CGI경우가 서블릿의 경우보다 거의 2배 정도의 이용현황을 보이고 있다.

본 시뮬레이션 결과에서 알 수 있듯이 큐의 지연, 패킷의 처리량과 네트워크의 이용현황을 알아보았다. 본 논문에서 제안한 구조가 애플릿과 CGI를 이용한 구조보다 안정적이고, 네트워크의 패킷의 처리량도 많고 그리고 네트워크의 이용도 면에서도 다른 구조보다 네트워크에 부담이 덜되는 것으로 측정되었다.

5. 결론 및 향후 연구과제

본 논문에서는 CORBA 서비스들을 CORBA환경이 아닌 웹 환경도 이용할 수 있고, 방화벽의 문제점도 해결하였으며, 그리고 기존의 연동 구조 보다 안정성이 있는 CORBA와 XML의 연동구조를 제안하고, 성능을 평가하였다.

웹응용서버에 서블릿 기술과 SOAP 기술을 사용함으로써 웹응용서버측의 네트워크 오버헤드를 줄일 수 있고, 기존의 CORBA 환경의 수정사항 없이 CORBA 서

버객체의 다양한 서비스들을 제공, 추가할 수 있고, 서로 다른 플랫폼들과 다른 분산환경들과의 연동도 용이하다. 또한, 방화벽 때문에 CORBA환경에 접근할 수 없는 문제를 해결하였다.

향후 연구 과제로는 본 연구에서의 보안과 컴포넌트 관리에 대한 더 자세한 연구가 필요하다

참 고 문 헌

[1] W3C, "XML specification, Level 1," http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[2] CORBA Specification, Version 2.3.1, "http://www.omg.org," OMG.

[3] W3C, "Document Object Model, Level 1," http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001, 1998.

[4] W3C, "Document Object Model, Level 2," http://www.w3.org/TR/1999/REC-DOM-Level-2-19990923, 1999.

[5] 김창수, 정회경, "XML 응용 개발환경", 한국정보과학회 학회지, 제 19권 1호 통권 제 140호, pp 15-23, 2001.1.

[6] Steven Vermeulen, Bart Bauwens, Rrans Westerhuis, Rudi Broos, "XML and CORBA, Synergistic or Competitive?," Proc. IS&N 2000, pp 155-168, 2000.

[7] OMG, "XML to Valuetype mapping," http://cgi.omg.org/cgi-bin/doc?orbos/00-05-01.pdf, 2000.

[8] OMG, "CORBA, XML And XMI Resource Page," http://www.omg.org/technology/xml/index.htm, 2000.

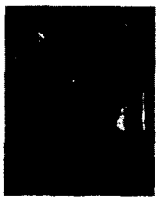
[9] W3C, "Simple Object Access Protocol(SOAP)1.1," http://www.w3.org/TR/SOAP, 2000.

[10] OMG, "Simple CORBA Object Access Protocol (SCOAP)," ftp://ftp.omg.org/pub/docs/orbos/00-09-03.pdf, 2000.

[11] UserLand Software, "The XML RPC Specifications," http://www.xml-rpc.com/spec

이 호 섭

2000년 호서대학교 컴퓨터공학과(공학사). 2000년 경희대학교 전자계산공학과 석사과정. 2002년 경희대학교 전자계산공학과 석사과정수료. 관심분야는 분산컴포넌트, 인터넷 서비스 관리



홍 충 선

1983년 경희대학교 전자공학과(공학사). 1985년 경희대학교 전자공학과(공학석사). 1997년 Keio University, Dept. of Infomation and Computer Science(공학박사). 1988년 ~ 1999년 KT 통신망 연구소 선임 연구원/네트워킹연구실장. 1999년 ~ 현재 경희대학교 전자정보학부 조교수. 관심분야는 인터넷 서비스 및 망 관리 구조, 분산컴포넌트관리 Mobile IP 프로토콜 등