RMI-IIOP를 이용한 CORBA 환경에서의 XML 객체 모델링

구태완[0] 정연진 엄상용 이광모
한림대학교 컴퓨터공학과
taewani@cie.hallym.ac.kr

XML Object Modeling for CORBA-Based System using RMI-IIOP

Tae-Wan Gu[0], Yeon-Jin Jung, Sang-Yong Uhmn, Kwang-Mo Lee
Dept. of Computer Engineering Hallym University

요 약

최근 WWW(World Wide Web) 환경이 발전함에 따라 기존의 정적인 데이터의 교환이 아닌 동적인 데이터 조작의 필요성이 대두 되었으며 이러한 데이터 형태의 표준으로 XML이 등장하게 되었다. 또한 분산 컴퓨팅 환경에서 클라이언트와 서버간의 데이터 교환 메커니즘이 WWW 환경으로 옮겨 가고 있고 XML과 같은 구조적이고 표준화된 형태의 데이터 교환이 필요하게 되었으나, HTTP를 사용하는 데이터 형태를 CORBA와 같은 분산 환경에 적용 시키기엔 상당한 제한 사항이 존재하므로 본 논문은 Java 객체 직렬화를 이용하여 RMI-IIOP를 통한 데이터 형태에 독립적인 교환 방법을 논의하고, 좀 더 효율적으로 XML 데이터 형태를 조작할 수 있는 시스템의 설계를 목적으로 한다.

1. 서 론

최근 인터넷을 이용한 비즈니스 모델의 형태가 B2C (Business to Customer)에서 웹을 기반으로 하는 B2B (Business to Business) 정보와 트랜잭션 시스템으로 옮겨 감에 따라 정보를 표현하는 표준의 필요성이 대두되고, 그에 대한 대안으로 XML(eXtensible Markup Language)이 등장하게 되었다[1]. 하지만 XML이 갖고 있는 수많은 장점에도 불구하고 HTTP를 기반으로 하는 문서 형태를 가지므로 이를 기존에 구현된 CORBA 기반 시스템에 적용시키기엔 많은 문제점이 존재한다[2]. 또한 CORBA 시스템은 IIOP(Internet Inter-ORB Protocol)을 사용하는데 이것은 일반적인 방화벽에 의해 차단될 수 있다[3]. 하지만 Elenko와 Reinetsen[4]에 의해 XML과 CORBA간의 포괄적인 통신 모델이 제안되기도 하였다. 그리고 XML 데이터를 위한 IIOP에 적합한 XIOP를 제안하기도 하였다[5].

본 논문에서는 외부 데이터 표현(External Data Representation)을 이용하여 XML 데이터 형태를 새롭게 정의한 후 이를 ORB(Object Request Broker)와 통신 할 수 있는 형태로 변형시킬 수 있는 기반 시스템을 설계하고자 한다.

2. 관련 연구

상호 통신 가능한 객체들에서 데이터를 교환하기 위해 CORBA의 CDR(Common Data Representation)과 Java의 객체 직렬화(Java Object Serialization)가 있다[6].

2.1 CORBA의 CDR

CDR은 CORBA 2.0 사양에서 정의된 외부 데이터 표현 방법인데 이것은 CORBA 기반 시스템에서 원격지 호출에서 수행된 결과값이나 인자로 사용되는 모든 형태의 데이터 타입을 새롭게 정의할 수 있는 장점이 있다. 이와 비슷한 것으로는 Sun Microsystems 사의 XDR 형태가 있다.

CDR은 CORBA 사양에 명세 된 것인 만큼 각 ORB의 IIOP를 이용한 상호 데이터 교환에 있어 아주 적합한 형태라고 할 수 있다.

그러나 원시 타입의 데이터를 CDR로 변환하여 전송하는 과정에서 송신측의 마샬링 연산(Marshalling Operation)과 수신측의 언마샬링 연산(Unmarshalling Operation)이 불가피 하다.

2.2 Java 객체 직렬화

객체의 원시 데이터 형태를 상호 교환이 가능한 바이트 스트림으로 송수신하기 위하여 객체를 스트림화 하는 것을 객체 직렬화라고 한다. 이것은 객체의 원시 데이터 형태가 입출력 형식에 독립적인 속성을 가지므로 데이터 영속성과 데이터 교환 방 법에도 편리함을 추구할 수 있다[7].

객체 직렬화를 하기 위해서 ObjectInputStream 클래스와 ObjectOutputStream 클래스에서 제공해 주는 여러가지 기능을 사용하게 되는데, ObjectInputStream 객체와 ObjectOutputStream 객체는 그 객체를 통하여 입출력된 객체들을 기억함으로써, 동일한 객체를 중복하여 입출력하지 않도록 해 준다. 이로써 객체 참조의 공유 및 원형 참조 문제를 해결하여 주고 있으며, 또한 배열을 전송할 경우 배열 역시 Object이므로 객체 직렬화로 입출력 될 수 있다.

그리고, CORBA에서 데이터 형태 교환 시 언급된 마샬링, 언마샬링 연산을 쓰지 않도록 Java 언어에서 기본적으로 리플렉션(Reflection) 기능을 지원한다는 장점이 있다.

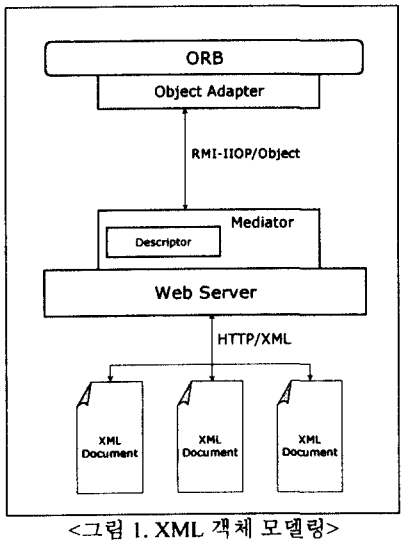
3. CORBA 환경에서의 XML 객체 모델링

3.1 XML 문서의 객체 모델링

XML 문서의 객체 모델링 기법으로는 W3C에서 제안된 DOM(Document Object Model)이 있다[8]. 이것은 어플리케이션 에서 XML 데이터에 대한 API를 제공하고 있다.

그러나, 이것은 HTTP를 사용하는 환경에서 IIOP를 사용하는 CORBA 환경으로의 이식성이 효율적이지 못하다.

다음은 XML 데이터를 RMI-IIOP를 이용하여 CORBA 환경으로 이식하기 위한 구조를 보여주고 있다.



<그림 1. XML 객체 모델링>

웹 서버로 들어오는 스트림들이 중간자(Mediator)에서 디스크립터를 참조하여 XML 구조를 객체로 변환하게 된다. 이때 객체는 클래스의 인스턴스로서 일반적인 구조체 형태를 띄게 되며, 이는 IDL로 작성할 수 있는 인터페이스의 변형된 모습과 동일한 구성을 갖게 된다. 아울러 상호 ORB와 통신할 수 있는 여건을 마련하기 위해 Serializable 인터페이스를 구현하고 RMI-IIOP를 이용한 데이터 전송을 모델링 한다.

3.2 중간자(Mediator)와 디스크립터 설계

중간자는 들어오는 XML 데이터 스트림을 XML 디스크립터에서 정의한 XML DTD를 참조하여 해당 CORBA 서비스로 바인딩 할 준비를 하고, 이를 전송 가능한 객체로 변환하는 일을 하게 되며 중간자, XML 디스크립터, 번역기(Translator)와 같이 세 가지 요소로 구성된다.

3.2.1 중간자(Mediator)

HTTP를 기반으로 하는 데이터 전송 메커니즘에서 유연성있는 서비스를 제공하기 위해 Java의 서블릿(Servlet)을 이용하여 XML DTD를 참조하고 각 연결을 유지 및 관리하기 위하여 별도의 쓰레드 풀(Thread Pool)을 유지하여야 한다. 이때 쓰레드 풀을 유지하기 위해 미리 n(≥10)개의 쓰레드를 생성한 후 들어오는 스트림에 대한 서비스를 제공하며 시스템의 성능을 고려하여 이를 다시 재사용하는 메커니즘으로 고안되어야 한다[9]. 아울러 스트림을 분석하여 이를 다시 객체로 변환 해 준다. 이때 미리 정의된 XML DTD는 정적으로 생성되어 있음을 가정한다.

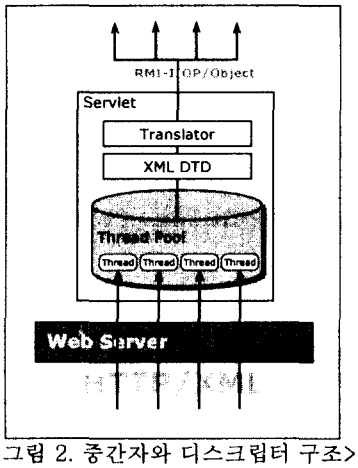
3.2.2 XML 디스크립터(Descriptor)

XML 클라이언트에서 요구된 XML DTD에 근거하여 이를 서블릿에서 참조 가능할 수 있도록 보관하게 된다. 이러한 디스크립터는 미리 정의된 XML DTD에 따라 해당 CORBA 서비스에 기초한 IDL 파일을 미리 생성하고, 이를 다시 객체 생성을 위한 런타임에 사용하게 된다. 그러나 디스크립터 자체는 미리 정의된 DTD에 근거하여 모든 클라이언트에 대해 동일하게 유지 되어야 한다.

3.2.3 번역기(Translator)

XML 디스크립터에 근거한 스트림을 객체화 시키는 과정으로 이는 구조화된 XML 데이터를 교환 가능한 객체 형태로 변환하고 RMI를 이용하여 데이터의 교환이 가능하게 하기 위해 이를 Serializable 인터페이스를 이용한 객체 직렬화를 구현한다. 이때 주의해야 할 사항은 다음과 같다[10].

- 원격 인터페이스의 모든 상수들의 정의가 런타임에 정의되어야 한다.
- IDL 컴파일러에 의해 생성된 이름과 Java 객체의 이름이 중복되면 안된다.
- 한개 이상의 서로 다른 기본 원격 인터페이스로부터 동일한 메소드 이름을 갖는 다른 원격 인터페이스로 상속할 수 없다.
- RMI를 이용하기 위해 RMISocketFactory, UnicastRemoteObject, Unreferenced 와 같은 형태는 사용하지 않는다.



<그림 2. 중간자와 디스크립터 구조>

3.3 RMI-IIOP

기존의 Java를 이용한 분산 환경에의 접근은 RMI 혹은 CORBA IIOP를 이용하였다. 하지만 이러한 불편함을 개선하기 위해 몇가지 제한 사항이 추가된 형태로 IIOP를 이용하여 CORBA 객체로의 접근이 가능 해졌다[10].

RMI-IIOP를 사용하기 위해서는 rmic 컴파일러와 idlj 컴파일러가 이용되는데 rmic는 IDL을 생략한 채 IIOP stub와 tie를 생성할 수 있으며, idlj는 RMI-IIOP 메커니즘에서 상호 연동에 필요한 데이터에 의한 새로운 CORBA 객체를 지원하고 있으며 이는 어떠한 플랫폼에서도 독립적으로 수행이 가능하다는 장점이 있다[10].

5. 결론 및 향후 연구 과제

XML은 구조적인 데이터 형태로서 구조체 형태로 정의될 수 있다는 것에 착안하여 이를 CORBA 환경으로 이식하는 것을 목적으로 한다. 이러한 과정에서 OMG에서 제안된 CORBA의 외부 데이터 표현 방법인 CDR(Common Data Representation)은 데이터 전송 시 마샬링, 언마샬링 연산이 필요하게 되고, 방화벽을 통한 데이터 전송이 불가피한 현실을 고려해야 한다. 그러므로 또 다른 외부 데이터 표현 방법인 Java의 객체 직렬화를 이용하여 XML 데이터 형태를 Serializable 인터페이스를 구현한 객체의 인스턴스 형태로 생성한 후 이를 다시 RMI-IIOP를 이용한 전송 기법을 도입하고 있다.

이러한 개념은 기존에 구성된 XML 데이터 형태를 IIOP 상에서 교환하기 위한 메커니즘의 대안으로 CDR을 사용했던 것과는 달리 Java 언어의 특징인 객체 직렬화를 이용하여 직접 데이터 교환을 가능하게 한다는 특징이 있다.

앞으로 추가적으로 논의되어야 할 사항으로는 각 벤더에 따라 서로 다른 특징을 갖는 어플리케이션 서버에서의 범용 조작기법에 대한 논의가 보완되어야 하며, 이를 단일 레이어에서 처리하는 방법에 의존하지 않고, 분산된 요청에 대해 동적인 XML 디스크립터를 구현하여 다양한 서비스를 요청할 수 있는 데이터 전송 기법이 요구된다.

5.참고 문헌

[1] Extensible Markup Language(XML) 1.0, W3C Recommendation 10-Feb 1998

[2] *Rogue Wave XORBA Link Technical White Paper*, 1999. 12., http://www.roguewave.com

[3] Wing Hang Cheung, Michael R. Lyu, and Kam Wing Ng, *Integrating Digital Libraries by CORBA, XML and Servlet*, JCDL'01, June 24-28, 2001, Roanoke, Virgia, USA

[4] Mark Elenko and Mike Reinertsen. *XML & CORBA. Application development trends*, September 1999.

[5] Homepage of XIOP, http://xiop.sourceforge.net

[6] *Distributed System Concepts and Design* (Third Edition), G. Coulouris, J. Dollimore, Tim Kindberg, 2001, Addison Wesley

[7] Core Java 2, Cay S. Horstmann, Gray Cornell, 1999, Sun Microsystems

[8] Beginning XML, K. Cargle, D. Gibbons, D. Hunter, N. Ozu, J. Pinnock, P. Spencer, 2000, Wrox Press

[9] 전상현, 엄상용, 정연진, 구태완, 이광모, *자바 쓰레드 풀을 이용한 웹 서버의 구현 및 성능 분석*, 2000. , 추계 정보처리 학회

[10] *RMI-IIOP Programmer's Guide*, 1999., Sun Microsystems