iORB 객체 호출을 위한 다중쓰레드 방식의 Basic Object Adapter (BOA) 구현

이 권일, 남궁 한

한국전자통신연구원 컴퓨터.소프트웨어기술연구소 인터넷서비스연구부

E-mail:{lki, nghan}etri.re.kr, Tel:042-860-6603, Fax:042-861-1030

The Implementation of Multi-Threaded Basic Object Adapter to Invoke Server Object on iORB

Kwon-Il Lee, Han Namgoong

Internet Service Department, Computer & Software Technology Laboratory, ETRI

요 약

CORBA 2.0 규격에 따라 구현된 인터넷 Java ORB 인 iORB 는 Common Object Request Broker Architecture (CORBA) 객체 호출을 위한 Basic Object Adapter (BOA)를 클라이언트와 서버 객체 사이의 연결 설정과 요청 처리를 분리한 다중 쓰레드 방식으로 제공하고 있다. 본 논문은 다중 쓰레드 방식을 지원하는 iORB 의 BOA 설계 및 구현에 관한 것이다.

1. 서론

Object Request Broker (ORB) 는 이기종 분산 환경에 있는 객체들 사이에서 클라이언트의 요청을 서버 객체에게 전달하고 서버 객체의 응답을 클라이언트에게 전달하는 객체 버스[1]로 ORB 를 이용하여 통신하는 객체들에게 위치 투명성을 제공해준다.

인터넷과 Web 환경의 급격한 확산은 Web 과 ORB 의 통합이 요구 되었고 이에 대한 해결책으로 자바 ORB 가 설계 구현되었다.

자바(Java)는 간단하고 플랫폼(시스템 기종, 운영체제) 독립적으로 수행되는 객체지향 프로그래밍 언어이다. 자바의 이러한 특성은 인터넷 발달과 함께 Web 환경에 적합한 언어로 인정받고 있다.

자바가 구현 투명성에 초점을 두고 있다면 ORB 의 표준 구조인 Common Object Request Broker Architecture (CORBA)는 통신 투명성에 초점을 두고 있다. 자바는 커다란 CORBA 시스템에서의 코드 분배를 단순화 할 것이다. 따라서 자바를 사용한 ORB 는 Web 을 이용하는 분산 객체들을 위해 보다 적합한 환경을 제공할 것이다[2,3]

ORB 의 서버쪽에 위치하는 Object Adapter (OA)는 클라이언트의 요청을 접수하여 적절한 서버 객체에게 전달하고 서버 객체의 처리 결과를 클라이언트로 보내는 일을 수행한다. 객체 어댑터는 서버쪽 ORB 의 맨 위에 위치하고 있으며, 호출 대상인 서버 객체의 종류에 따라 ORB 는 여러 개의 객체 어댑터를 제공할 수 있다. 예를 들어 객체 지향 데이터베이스 시스템 객체를 호출하기 위한 객체 지향 데이터베이스 Object Adapter, CORBA 객체의 호출을 처리하는 Basic Object Adapter (BOA) 등이 ORB 에 의해 제공될 수 있다.

그러나 CORBA 객체 외의 다른 객체들을 CORBA 환경에 적용할 때 필요에 의해 적절한 객체 어댑터를 구현하여 사용하여야 하므로 ORB 가 기본적으로 제공하는 것은 어렵다.

CORBA 2.0 규격도 CORBA 객체 호출을 담당하는 BOA 에 관해서만 명시하고 있다.

본 논문은 CORBA 2.0 규격에 따라 구현한 Java ORB 인 iORB 의 BOA 의 설계 및 구현에 관한 것이다.

본 논문의 2 장에서 iORB 에 관해 간략히 살펴보았고 3 장에서 다중 쓰레드를 지원하는 iORB 의 BOA 의 구조, 동작 절차, BOA 를 구성하는 클래스들 위주로 기술하였다. 마지막으로 4 장에서 결론을 기술하였다.

2. iORB

iORB 는 CORBA 2.0[4] 규격에 따라 Java 로 구현된 ORB 로 (그림 1)과 같은 객체 요청 중개자 인터페이스 구조를 가진다.

클라이언트는 Dynamic Invocation Interface 또는 IDL Stub 을 사용하여 서버 요청을 만들고, 서버는 Dynamic Skeleton 또는 IDL 에 의해 생성된 Skeleton 을 통해 클라이언트의 요청을 수용한다.

기본 객체 어댑터는 클라이언트의 요구를 서버에게 전달하는 역할을 수행하는 것으로 서버에 의해 사용되는 인터페이스를 제공한다.

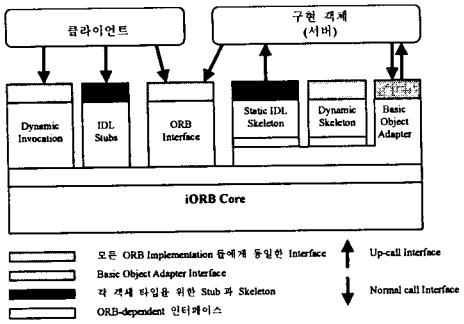
ORB 인터페이스는 클라이언트와 서버에 의해 호출되며 iORB Core 에 직접 접근할 수 있는 기능을 제

공한다.

iORB Core 는 앞에서 기술한 인터페이스들이 동작하는데 필요한 기본 기능을 제공하면서 Internet Inter-ORB Protocol (IIOP) 프로토콜을 사용하여 원격 시스템에 있는 iORB 나 다른 ORB 와의 통신을 담당하는 IIOP 엔진을 포함하고 있다.

또한 iORB 는 IDL-to-Java 컴파일러을 제공하여 IDL 인터페이스 화일을 IDL-to-Java 사상 규칙에 따라 자바 화일로 변환할 수 있게 한다.

iORB 에서 클라이언트 스텁 화일과 서버 스켈레톤 화일은 IDL 인터페이스 화일을 컴파일하여 생성된다. 클라이언트 스텁 화일은 Dynamic Invocation Interface 를 이용하여 구성되며, 서버 스켈레톤 화일은 Dynamic Skeleton Interface 를 이용하여 구성된다. 이는 IDL 인터페이스 화일을 컴파일하여 생성되는 클라이언트 스텁 화일과 서버 스켈레톤 화일의 이식성을 보장하기 위한 것이다.



(그림 1) 객체 요청 중개자 인터페이스 구조

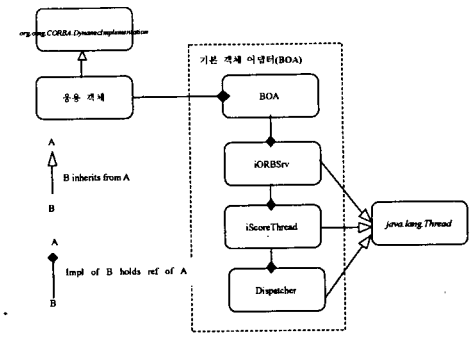
3. iORB Basic Object Adapter 설계 및 구현

iORB BOA 는 CORBA 2.0 에서 정의한 BOA 를 기준으로 하였다. CORBA 2.0 규격은 BOA 의 동작 구조 및 역할에 대해서만 정의하고 있을 뿐 ORB Core 나 서버 Skeleton 과의 인터페이스에 대해서는 정의하지 않고 있다. 이는 ORB 를 구현할 때 나름대로 정의하여 사용할 수 있게 한 것이다.

3.1. BOA 구성 클래스 구조도

BOA 는 BOA, iORBSrv, iScoreThread, Dispatcher 등 개의 클래스로 구성된다. iORBSrv, iScoreThread, Dispatcher 는 Java 의 Thread 클래스를 상속 받아 서버 쪽 다중 쓰레드를 지원하는 클래스들이다. BOA 클래스는 응용 인터페이스를 제공하는 클래스로 활성화된 객체 관리, 다중 쓰레드 초기화 등을 제공한다.

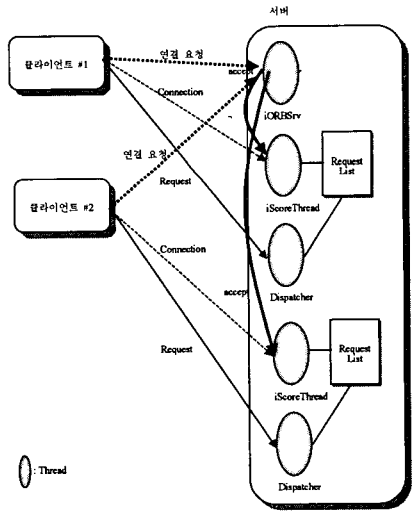
BOA 를 구성하는 클래스들 사이의 관계는 (그림 2)와 같다.



(그림 2) BOA 구성 클래스들 사이의 관계도

3.2. 다중 쓰레드 모델

iORB 의 BOA 는 클라이언트와 서버사이의 연결 설정을 위한 쓰레드와 클라이언트의 요청 처리를 위한 쓰레드를 분리하였다. (그림 3)은 iORB BOA 의 쓰레드 모델을 보여주고 있다.



(그림 3) iORB 다중 쓰레드 모델

iORBsrv 는 BOA 의 한 구성 요소로 클라이언트의 요청을 접수하는 데몬 쓰레드이다. iORBSrv 는 서버 프로세스 내에 하나 존재하며 다중 클라이언트의 요청을 접수한다.

iScoreThread 는 클라이언트와 서버의 연결당 하나 존재하여 클라이언트의 요청을 접수하는 일을 한다. iScoreThread 에 의해 접수된 클라이언트 요청은 Request List 에 매달리게 된다. iScoreThread 는 클라이언트가 연결을 해제할 때 종료된다.

Dispatcher 는 클라이언트의 요청마다 생성되는 쓰레드로 Request List 에 매달려 있는 클라이언트의 요청을 실제로 처리하는 일을 수행한다. 클라이언트의 요청

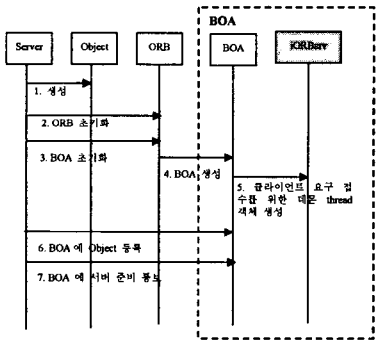
처리를 마치면 Dispatcher 쓰레드는 종료한다.

3.3. 서버 객체 등록 과정

서버가 활성화 객체를 BOA 에 등록하여야만 클라이언트가 이용할 수 있다. (그림 4)는 서버가 클라이언트의 요청을 처리할 준비를 하기위해 BOA 에 등록하는 과정을 보여준다.

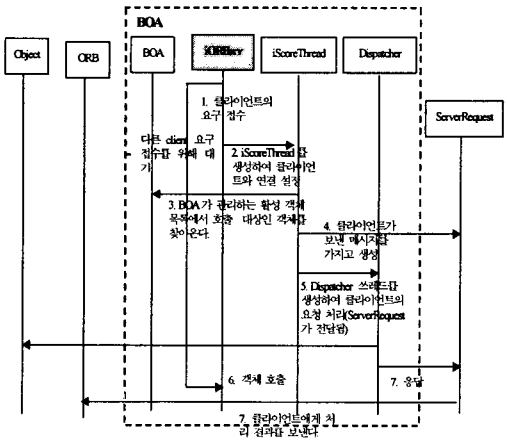
서버는 클라이언트의 요청을 처리할 객체를 생성하고 클라이언트와의 접속을 위해 통신 채널인 ORB 를 초기화 한다. 서버는 ORB 에게 BOA 초기화 요청을 하여 서버 호출을 위해 BOA 를 생성한다. 서버는 하나 이상의 BOA 를 초기화 할 수 있다. BOA 는 데몬 쓰레드인 iORBsrv 쓰레드를 생성하여 클라이언트의 요청을 처리할 준비를 마친다. iORBsrv 는 서버가 1 개 이상의 클라이언트와 동시에 통신할 수 있도록 한다.

초기화 작업을 완료한 서버는 BOA 에 클라이언트 요청을 처리할 객체를 등록하고 서버가 준비 완료되었음을 BOA 에게 통보한다.



(그림 4) BOA 에 서버 등록 과정

3.4. 서버 객체 호출 과정



(그림 5) 클라이언트 요청 처리 과정

(그림 5)는 BOA 가 클라이언트의 요청을 접수하여 이를 처리하는 과정을보여준다. 클라이언트의 연결 설정 요청을 접수한 iORBsrv 는 iScoreThread 를 통해 클라이언트와의 실제적인 연결을 설정한다.

클라이언트와 연결을 설정한후, 클라이언트의 요청을 접수한 iScoreThread 는 BOA 를 통해 호출할 객체를 찾아 온다. 그리고 Dispatcher 쓰레드 객체를 생성하여 클라이언트의 요청을 처리할 것을 요구한다. Dispatcher 는 객체 호출을 위해 클라이언트의 요청 정보를 가지고 ServerRequest 객체를 생성한 후, ServerRequest 정보를 가지고 서버 객체를 호출한다. 서버 객체는 클라이언트의 요청을 처리한 결과를 ServerRequest 에 반영한다. 객체 호출을 마친 Dispatcher 는 처리 결과를 ServerRequest 와 ORB 를 통해 클라이언트에게 전달한다.

4. 결론

분산 환경에서 동작하는 분산 응용들은 효율성 측면에서 동시에 여러 응용과 통신하기를 원한다. 다대다 통신은 응용 프로그램이나 분산 환경을 제공하는 미들웨어, 또는 가끔 운영체제에 의해 지원된다.

응용 프로그램에서 다중 통신을 지원하는 방법은 응용 프로그램 작성자가 응용 프로그램의 특성에 적합한 다중 통신 방식을 택할 수 있다는 융통성을 제공한다. 그러나 응용 프로그램 작성에 부담을 준다는 점에서는 바람직하지 않다. 그래서 가장 일반적으로 사용하는 방법이 미들웨어가 다중 통신을 지원하는 것이다. 이 때 응용의 종류에 융통성을 부여하기 위해 응용이 자신이 사용할 다중 통신 방식을 선택할 수 있게 한다.

iORB 는 다중 쓰레드 방식의 BOA 를 통해 서버가 동시에 여러 클라이언트의 요청을 처리할 수 있도록 한다. 이 논문에서 기술한 BOA 는 클라이언트가 전송하는 요청별도 쓰레드를 부여하는 방식을 제공하고 있다. 앞으로 클라이언트와 서버 연결 당 하나의 쓰레드를 부여하는 방식, 서버가 오직 하나의 쓰레드만을 제공하여 동시에 여러 클라이언트와 통신 할 수 없도록 하는 방식 등을 제공하여 응용의 종류에 따라 적합한 방식을 선택하여 사용할 수 있도록 확장할 예정이다.

iORB 의 차기 버전은 OMG CORBA 2.2 규격[5]에서 서버 응용 프로그램의 이식성 등을 제공하기 위해 제시한 Portable Object Adapter (POA)를 지원할 것이다.

참고문헌

[1] Randy Otte, Paul Patrick, and Mark Ray, *Understanding CORBA*, Prentice Hall PTR , Upper Saddle River, NJ, 1996.

[2] Robert Orfali, Dan Harkey, Jeri Edwards, *The Essential Distributed Objects Survival Guide*, John Wiley & Sons, Inc., 1996.

[3] Robert Orfali, Dan Harkey, *Client/Server Programming with Java and CORBA*, John Wiley & Sons, Inc., 1997.

[4] OMG, *The Common Object Request Broker: Architecture and Specification Revision 2.0*, OMG, July 1995.

[5] OMG, *The Common Object Request Broker Architecture and Specification Revision 2.2*, OMG, February 1998.