# 레거시 자바객체를 분산객체로 변환하는 시스템

이상윤
대원과학대학 컴퓨터정보처리과
sylee@daewon.ac.kr

# A Object Converting System for Legacy Java Object to Distributed Object

Sangyun Lee
Dept. of Computer Information Processing, Daewon Science College

## 요 약

분산 컴퓨팅 환경에 적용해야할 응용 프로그램이 점점 많은 비중을 차지하게 됨에 따라, 이를 지원하기 위한 다양한 형태의 분산 프로그래밍 도구들이 제안되어 있다. 그러나, 기존의 도구를 이용하여 분산처리 소프트웨어를 작성하기 위해서는 분산 프로그래밍 도구가 요구하는 부가적인 프로그래밍 지식을 숙지하여야 한다. 부가적인 지식 없이 분산처리를 수행하는 소프트웨어를 개발할 수 있다면, 분산처리에 대한 개발자의 부담을 줄여서, 개발하는 소프트웨어의 자체 기능에 더욱 집중할 수 있다. 본 논문에서는 새로운 형태의 객체 변환 시스템을 제안하고, 이름을 TORB(Transparent Object Request Broker)라고 명명하였다. TORB를 이용하면, 프로그래밍 투명성의 지원을 통하여, 자바로 작성하는 분산처리 소프트웨어를 분산 프로그래밍 도구와 상관 없이 작성하고, TORB의 후처리 도구를 통하여 분산처리에 관여하는 자바 객체를 실제 분산처리를 수행하도록 변환한 후, TORB가 제공하는 분산처리 환경에서 수행할 수 있다.

## 1. 서론

분산 컴퓨팅 환경에 적용해야할 응용 프로그램이 점점 많은 비중을 차지하게 됨에 따라, 이를 지원하기 위한 다양한 형태의 분산 프로그래밍 도구들이 제안되어 왔다. 객체지향 디자인 기법을 적용한 분산 프로그램은, 이식성과 재 사용성을 통하여 복잡한 시스템 구축에 적합하다는 장점 때문에, 널리 사용되고 있는 분산 소프트웨어 개발기법 중의 하나이다[1]. 객체지향 분산 시스템을 지원하기 위한 방안으로써, CORBA, DCOM, DSOM, JAVA RMI, HORB등 여러 가지 도구들이 제안되어 있다[2,3,4,5,6]. 그러나, 이러한 도구를 활용하여 분산처리 소프트웨어를 작성하기 위해서는 분산 프로그래밍 도구가 요구하는 부가적인 프로그래밍 지식을 숙지하여야 한다.

부가적인 지식 없이 분산처리를 수행하는 소프트웨어를 개발할 수 있다면, 분산처리에 대한 개발자의 부담을 줄여서, 개발하고자 하는 소프트웨어의 자체 기능에 더욱 집중할 수 있다. 분산 프로그래밍 도구가 요구하는 부가적인 지식은 소프트웨어의 자체기능과 상관없는 부담이며, 이러한 부담은 기존의 분산 프로그래밍 도구가 프로그래밍 투명성을 만족할 만큼 지원하지 못하기 때문에 나타나는 오버헤드이다. 프로그래밍 투명성이란 프로그래머가 분산환경에서 동작하는 프로그램을 개발할 때, 로컬 환경에서 동작하는 프로그램을 개발하는 것과 동일한 방법을 적용할 수 있도록 지원하는 것이다.

본 논문에서는 프로그래밍 투명성을 지원하는 새로운 분산프로그래밍 도구를 제안하고, 그 이름을 TORB(Transparent Object Request Broker)라고 명명하였다. TORB에서는 프로그래밍 투명성의 지원을 통하여, 자바로 작성하는 분산처리 소프트웨어를 분산 프로그래밍 도구와 상관 없이 작성하여, 객체변환을 수행하는 후처리를 거친 후, TORB가 제공하는 분산처리 환경에서 목적된 분산처리 동작을 수행한다.

본 논문은 객체변환을 수행하는 후처리에 의한 분산 프로그래밍 도구를 제안하기 위하여 다음과 같이 구성된다. 서론에 이어 2장에서는 TORB가 지원하는 분산처리 수행환경의 구조와 동작 메커니즘을 설명한다. 3장에서는 TORB를 구현하기 위한 핵심적인 사항을 제시하고, 4장에서는 결론 및 향후 연구계획을 제시한다.

## 2. 분산처리 수행환경

TORB의 후처리 도구에 의해 변환된 프로그램이 분산처리를 수행하며 동작할 환경이 필요하다. 따라서, TORB는 그림1과 같이 정의된 분산처리 수행환경을 제공하며, 표준 자바 가상기계에서 동작하는 서버와, 변환된 분산처리 프로그램 사이의 연계된 작용에 의해 동작한다.

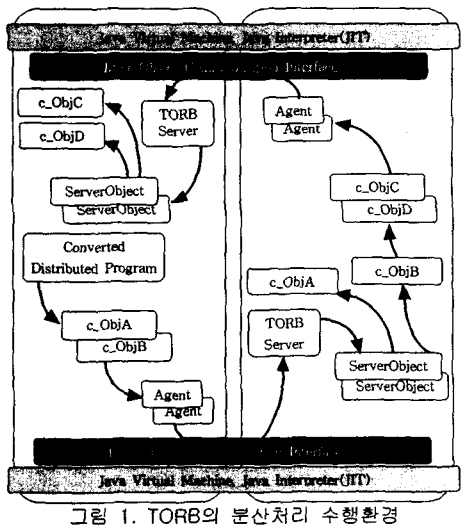


그림 1. TORB의 분산처리 수행환경

다음은 그림1에 나타난 세부적인 동작 메커니즘이다. 프로그래머는 원격객체 ObjA, ObjB와 로컬객체 ObjC, ObjD를 사용하는 프로그램을 일반적인 프로그래밍 방법으로 작성하였고, 이는 후처리 과정을 거쳐 c_ObjA, c_ObjB, c_ObjC, c_ObjD가 생성되며 이들 객체간에 상호 작용을 하는 분산처리 프로그램으로 변환된다.*(이때, c_ObjX는 변환하기 전의 객체이름과 동일한 이름을 가지며, 자바 바이트코드의 내용이 변환되어 원래의 객체를 대체한 것이다.)* 로컬 시스템에서 c_ObjA와 c_ObjB에 대한 접근동작이 일어날 때, TORB에 정의된 객체(Agent)의 지원을 받아 원격 컴퓨터의 "TORB Server"에게 적절한 요청을 한다. 원격 시스템에서는 객체 서비스를 위하여 TORB에 정의된 객체(ServerObject)를 통하여 접근하고자 하

는 객체(c_ObjA, c_ObjB)에 대한 적절한 처리를 수행하고 "Agent"의 요청에 응답한다. 한편, 원격의 c_ObjB객체는 동일한 방법으로 로컬 컴퓨터의 c_ObjC와 c_ObjD를 접근할 수 있다.

"TORB Server"는 원격 호스트에서 동작해야 하는 객체에 대한 서비스를 지원하는 기능을 수행하므로 작성된 프로그램이 실행을 시작하기 전에 동작 중이어야 한다. 따라서, 분산처리에 참여하는 모든 호스트에 "TORB Server"를 기동하는 조치가 필요하다. TORB에서는 "TORB Server"를 기동하기 위한 두 가지 방법을 제공한다. 하나는 사용자가 명령라인에서 직접 서버를 기동하는 방법이고, 다른 하나는, 변환된 분산처리 프로그램에 "TORB Server"를 중복되지 않게 기동하는 코드를 삽입하여, 작성한 프로그램이 실행을 시작할 때 자동으로 기동하도록 하는 방법이다.

## 3. TORB의 설계 및 구현방안

### 3.1 TORB 서버 및 관련된 부속객체

TORB 서버는 임의의 객체를 원격에서 접근하기 위하여 대행객체(_Proxy)의 동작에 의해 네트워크를 통하여 전달되어오는 요청을 분석하여 해당 객체에 존재하는 메쏘드(request)를 수행한 후, 그 결과를 원격에 존재하는 대행객체에게 네트워크를 통하여 응답한다. 이때, 하나의 대행객체에 대해서 각각 별도의 서비스객체(ServerObject)가 대응되도록 구현함으로써 여러 개의 원격객체를 동시에 처리하도록 구현 할 수 있다. 자바의 Reflection 기능을 이용한 메쏘드의 동적호출(Dynamic Invocation)방법을 적용하면, TORB 서버가 임의의 객체에 포함된 메쏘드를 수행하도록 하는 공통적인 프로그램 코드를 작성할 수 있어, 임의의 객체에 대한 서비스를 원격에서 수행하는 통합된 서비스객체(ServerObject)와 임의의 객체에 대한 서비스를 로컬에서 대신 처리할 통합된 대행객체(_Proxy)를 구현할 수 있다. 그 외에 소켓의 연결설정, 쓰레드 관리 등 임의의 객체가 원격 호스트에서 생성되어 동작하고 소멸될 때까지 TORB 서버가 적절한 기능을 수행할 수 있도록 필요한 부속 프로그램을 구현할 수 있다.

### 3.2 대행객체

TORB의 후처리에 의해 변환된 분산프로그램이 적절한 분산처리를 수행하기 위해서는 원격 호스트에서 동작할 객체에 대응할 대행객체(_Proxy)가 마련되어 있어야 한다. TORB에서는 임의의 원격객체에 대한 모든 접근(Access)을 대신할 통합된 자바 객체를 정의하였으며, 이를 통하여 자바 객체간의 분산처리 동작이 이루어진다. 본 연구팀에서는 임의의 객체에 대한 대행객체를 각각 독립적으로 생성하여 분산처리를 수행하는 분산프로그래밍 도구를 발표한 적이 있으나, 통합된 대행객체를 적용함으로써 프로그래밍 투명성의 범위가 넓어지는 결과를 얻을 수 있었다.[7]

#### 3.2.1 대행객체의 내용

TORB를 활용하는 것을 고려하면서 작성하는 분산처리 프로그램은 TORB가 제공하는 후처리를 통하여 변환된 후에야 실제적인 분산처리 역할을 수행한다. 후처리는 원격 호스트에서 동작할 객체를 접근하는 프로그램의 모든 코드를 대행객체를 통해서 접근하는 코드로 변환하는 것이다. 일반적인 자바 객체를 접근하는 방법으로 작성된 프로그램 코드가 대행객체를 통하여 원격객체를 접근하는 코드로 변환되어 분산처리 역할을 수행하는 프로그램이 생성되므로 임의의 객체에 대한 모든 접근동작을 대행객체를 통하여 수행할 수 있어야 한다. 즉, 대행객체를 통한 원격객체의 접근동작이 그 객체를 로컬객체로 취급하여 접근하는 동작을 모두 포함한다면 프로그래밍 투명성을 보장하기 위한 대행객체의 필요한 기능이 모두 갖추어졌다고 말할 수 있다. 자바 프로그램에서 임의의 객체에 대하여 적용할 수 있는 접근조작은 객체 인스턴스의 생성, 메쏘드의 호출, 멤버데이터의 접근, 객체자체를 인수로 취급하여 전달, 상속성 및 다형성에 의한 객체서비스 등이다. 임의의 객체에 대한 모든 접근조작을 대신해서 수행하는 대행객체에는 다음과 같은 내용과 기능이 포함된다.

- ◆ 임의의 객체에 대한 인스턴스를 그 객체에 정의된 모든 생성자를 적용하여 생성할 수 있는 기능. 이는 임의의 원격객체를 생성하는 프로그램 코드가 대행객체를 통하여 원격시스템의 해당객체를 생성하는 코드로 변환되기 때문이다.
- ◆ 임의의 객체에 정의된 모든 메쏘드를 호출할 수 있는 기능. 와 동일한 원형을 가지는 모든 메쏘드. 이는 임의의 원격객체에 정의된 메쏘드를 호출하는 프로그램 코드가 대행객체를 통하여 호출하는 코드로 변환되기 때문이다.
- ◆ 임의의 원격객체가 인수로 취급되어 다른 호스트에 전달되는 기능을 위하여, 대행객체를 인수로 취급하여 대신 전달하여도 동일한 효과를 발휘할 수 있도록 필요한 기능.
- ◆ 임의의 객체에 대한 멤버데이터를 접근하는 조작이 대행객체를 통해서 수행될 수 있도록 지원하는 기능
- ◆ 원격 호스트와의 통신기능을 담당하기 위하여 필요한 기능
- ◆ 원격객체가 생성되어 실행될 호스트의 URL을 실행시간에 지정할 수 있는 기능. 이는 원격객체가 생성되어 동작할 호스트가 실행시간에 결정되는 경우에만 필요한 기능이다.
- ◆ 상속성 과 다형성에 대한 객체서비스는 원격에 존재하는 임의의 객체에 자신에 대한 접근조작을 지원하기만 하면 자연히 처리된다, 왜냐하면, 원격에서 생성되어 원격에서 동작하는 객체는 이미 자바의 모든 객체서비스를 처리할 수 있는 기능을 보유하기 때문이다.

#### 3.2.2 대행객체의 구현방안

- ◆ 임의의 객체에 대한 인스턴스를 생성하는 기능과 정의된 메쏘드를 호출하는 기능은 자바의 Reflection 기법을 활용한 동적 메쏘드 호출기법을 적용하여 구현할 수 있다.
- ◆ 대행객체에는 임의의 객체를 원격 호스트에서 접근하기 위한 모든 방안이 마련되어 있으므로, 원시객체 대신에 대행객체가 인수로 전달되어도 동일한 효과를 발휘하여야 한다. 이를 위하여, 대행객체는 네트워크를 통하여 다른 시스템으로 전송할 수 있도록 구현되어야 한다. 임의의 자바객체를 네트워크를 통하여 전송할 수 있도록 구현하기 위해서는 반드시 자바의 Serialization 기능을 적용하여야 하고, 그렇게 하기 위해서는 대행객체 내에 기계 종속적인 멤버데이터(원격시스템과의 통신기능을 수행하는 소켓 등)가 존재하지 않아야 한다. 따라서, 원격시스템과의 통신기능에 관련된 모든 요소를 멤버데이터를 사용하지 않고 메쏘드 호출에 의해 수행하도록 구현한다. 이로 인하여 매번 원격객체를 접근하는 동작을 수행할 때마다 연결설정이 이루어져야 하는 필연적인 오버헤드를 가진다. 그러나, 한번 설정된 연결이 유지되는 형태의 시스템에서는 네트워크의 단절이 곧 시스템오류가 되어 해당 객체를 더 이상 접근할 수 없게되는 결과를 초래하지만, TORB는 원격 호스트와의 연결이 지속적으로 유지되는 형태가 아니므로, 네트워크가 회복되면 해당객체에 대한 접근을 정상적으로 재시도 할 수 있으므로 오버헤드에 대한 보상을 어느 정도 기대할 수 있다.
- ◆ 멤버데이터에 대한 접근은 _get_x()와 _put_x(int value)와 같은 형태의 메쏘드를 동적으로 호출하는 것으로 해결할 수 있다. 이는 콜래스파일을 변환하는 후처리를 수행할 때, 분산처리에 참여하는 모든 객체에 대하여, 해당 객체의 멤버데이터를 접근하는 _get_x()와 _put_x(int value)와같은 형태의 메쏘드를 강제로 추가하는 절차를 수행하기 때문이다.
- ◆ 원시객체가 생성되어 실행될 호스트의 주소를 실행시간에 동적으로 지정하는 기능은 대행객체의 메쏘드 정의에 URL을 지정하는 메쏘드를 추가해 두고, 프로그램 코드 내에서 이 메쏘드를 호출할 수 있는 기능을 마련해 놓으면 된다.

즉, 임의의 객체에 대하여 URL을 설정하는 메쏘드에 대한 호출코드는 그 객체에 정의된 메쏘드를 호출하는 것이 아니라 대행객체에 정의된 메쏘드를 호출하는 코드로 변환하여 구현할 수 있다.

### 3.3 클래스파일(자바 바이트 코드) 변환

TORB의 후처리 도구에 의해 변환되기 이전의 자바 프로그램은, 원격 호스트에서 생성되어 동작하는 객체를 접근하기 위하여 작성되었지만, 원격객체를 접근하기 위하여 대행객체를 접근하는 자바 바이트 코드가 아니고 로컬에 존재하는 원시객체를 접근하는 자바 바이트 코드로 컴파일되어 있다. TORB의 후처리 과정은 원격객체를 접근하는 바이트 코드를 대행객체를 통하여 접근하는 바이트 코드로 변환하는 작업이다. TORB는 이러한 기능을 구현하기 위해서 BCEL의 jasmin과 JasminVisitor를 보조도구로 활용한다. Jasmin은 자바 어셈블리 코드를 클래스파일로 생성해 주는 기능을 가지는 자바 어셈블러이고, JasminVisitor는 자바 객체의 인스턴스로부터 자바 어셈블리 코드를 생성하는 자바 역어셈블러이다.[8,9] TORB의 후처리 도구가 처리하는 변환작업은 JasminVistor에 의해 역어셈블 되어 생성된 자바어셈블리 코드를 변환하고, jasmin에 의해 변환된 클래스파일을 생성하는 절차로 구성된다. JasminVisitor에 의해 역 어셈블 되어 생성된 자바어셈블리 코드에 대한 변환조작의 주된 내용은 다음과 같다. 여기에 나타나는 분산객체는 분산처리에 참여하는 모든 객체들이고, 대행객체는 TORB에 정의된 TORB.RUNTIME._Proxy이다.

- ◆ 분산객체의 자바 어셈블리코드에 대해서, 분산객체를 생성하는 코드를 TORB에 정의된 대행객체(_Proxy)를 통하여 생성하는 코드로 대치한다. 이로 인하여 해당 객체는 원격 호스트에서 생성되고, 로컬에는 해당 객체에 대한 대행객체가 생성된다. 따라서, 이후에 발생하는 원격 객체에 대한 모든 접근코드는 이때 생성된 대행객체를 통하여 이루어지도록 변환할 수 있다. 대행객체의 생성자에는 TORB 서버를 중복되지 않게 기동하는 기능이 포함되어 있다. 여기서 기동되는 TORB 서버는 로컬 호스트에서 동작하는 것이고, 해당 객체 자신이 직접 혹은 간접으로 다른 분산객체에 인수(parameter)로서 전달되었을 때, 그 분산객체에 대한 원격객체 서비스를 수행하기 위한 것이다.
- ◆ 분산객체의 자바 어셈블리코드에 대해서, 분산객체의 메쏘드를 호출하는 코드를 TORB에 정의된 대행객체(_Proxy)를 통하여 호출하는 코드로 대치한다. 이로 인하여, 원격 메쏘드 호출이 이루어진다.
- ◆ 분산객체에 포함된 모든 멤버데이터에 대하여 _get_x()와 _put_x(int value)와 같은 형태의 이름을 가지는 메쏘드를 강제로 추가하기 위한 자바 어셈블리 코드를 추가한다. 이는, 멤버데이터에 대한 직접접근을 추가된 메쏘드를 통하여 처리하기 위한 것이며, 이는 분산객체의 클래스정의에 멤버데이터를 접근하는 메쏘드를 마련해 두는 역할을 한다.
- ◆ 분산객체의 자바 어셈블리코드에 대해서, 멤버데이터를 접근하는 모든 코드(putfield, getfield)가 분산객체의 멤버데이터를 대상으로 하는 연산이면, 분산객체에 강제로 포함시켜 둔 _get_x()와 _put_x(int value)와 같은 형태의 메쏘드를 호출하여 그 결과를 처리하는 코드로 대치한다. 이는 프로그래밍 투명성을 해치지 않고 멤버데이터에 대한 직접접근을 지원하기 위한 것이다.
- ◆ 분산객체의 자바 어셈블리코드에 대해서, 메쏘드의 원형(Prototype)을 정의하는 부분에 나타난 분산객체의 참조를 대행객체(_Proxy)로 대치한다. 이는 분산객체가 인수로 취급되어 전달되거나 메쏘드의 결과값으로 취급될 때, 대행객체가 그 역할을 대신하도록 처리하기 위한 것이다.
- ◆ 분산객체의 자바 어셈블리코드에 대해서, 자바의 키워드 "this"를 대신하기 위한 멤버데이터(대행객체)에 대한 어셈블리 코드를 삽입한다. 이것은 이 객체 자신이 자바의 키워드 "this"에 의해서 다른 원격 호스트에 인수로서 전달될 때 대치하기 위한 것이다. 이런 경우, 삽입된 멤버데이터가 자바의 키워드 "this"를 대치하도록 어셈블리 코드를 적절히 변환해야 한다.
- ◆ 분산객체의 자바 어셈블리코드에 대해서, 멤버데이터를 정의하는 코드에 나타난 분산객체를 대행객체로 대치한다. 멤버데이터로 활용되는 분산객체에 대한 모든 접근조작은 대행객체를 통하여 이루어져야 하기 때문이다.
- ◆ 이상 제시한 변환조작 외에도 TORB가 지원하는 후처리 도구에 의해 변환된 자바 프로그램이 최초 목적한 분산처리를 수행하도록 적절히 변환하는 단편적인 조작이 필요하다. 이러한 변환조작은 후처리 도구를 구현하기 위한 단순한 코딩 기술이므로 생략한다.

## 4. 결론 및 향후연구

부가적인 지식 없이 분산처리를 수행하는 소프트웨어를 개발할 수 있다면, 분산처리에 대한 개발자의 부담을 줄여서, 개발하는 소프트웨어의 자체 기능에 더욱 집중할 수 있다. 본 논문에서는 분산환경에 적용하기 위하여 프로그램을 작성할 때, 프로그래밍 투명성을 지원하여 상대적으로 쉽게 분산처리 프로그램을 작성할 수 있도록 해주는 분산객체 변환 시스템을 제안하고, 이름을 TORB라고 명명하였다. TORB가 지원하는 프로그래밍 투명성의 범위가 어느 정도이고, 충분한 프로그래밍 투명성을 지원하는가에 대한 정형화된 검증을 대신하여, 여러 가지 단위 프로그램에 대한 테스트를 수행하였으며 상당한 수준의 프로그래밍 투명성을 지원하는 것을 확인하였다. 그 결과, 실제의 분산처리 프로그래밍에 적용하여도 수용 가능한 것으로 평가하였다. 향후, 프로그래밍 투명성에 대한 정형화된 연구와 완전한 프로그래밍 투명성을 지원하기 위한 지속적인 기능 개선이 필요하다.

## 참고문헌


[ 1] J. R. Nicol, C. T. Wilkes and F. A. Manola, "Object Orientation in Heterogeneous Distributed Computing Systems", IEEE Computer, Vol. 26, No. 6, June, 1993.

[ 2] J. Siegel, "CORBA Fundamentals and Programming", John Wiley & Sons, 1996

[ 3] T. L. Thai, A. Oram, "Learing DCOM", O'reilly, April, 1999.

[ 4] IBM, "SOMobjets : A pratical Introduction to SOM and DSOM", International Technical Support Organization, 1994

[ 5] Sun Microsystems, Inc., "Java Remote Method Invocation", Online publishing, URL http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html, 2003.

[ 6] Satoshi Hirano, "HORB : Distributed Execution of Java Programs", In Lecture Notes in Computer Science 1274, Springer, pp.29-42, 1997.

[ 7] 이상윤, "분산프로그래밍 도구의 프로그래밍투명성 지원 방안", 한국정보과학회 2003년 추계학술발표대회 논문집, 제30권 2호, pp. 52-54, 2003.

[ 8] J. Meyer and T. Downing, "Java Virtual Machine". O Reilly, 1997.

[ 9] M. Dahm. "Byte code engineering with the BCEL API". Technical Report B-17-98, Freie Universitat Berlin, Institut fur Informatik, April 2001.