□특집□

자바와 코바를 이용한 분산 어플리케이션

김 홍 윤† 이 재 용†† 황 교 철††† 한 성 택†††† 홍 종 준†††††

◈ 목 차 ◈

1. 서 론

인터넷은 93년에 시작된 웹(Web) 서비스와 94년부터 허용되기 시작한 상업화로 전문가들에게서 일반 사용자들에게 급속히 확산되면서 명실상부한 글로벌 정보 통신망으로 발전하였다. 이러한 웹의 확산은 마침내 정보 시스템 분야에까지 불어와 기존의 정보 시스템을 웹에 연동 시켜 웹 브라우저를 통해 인터넷이 연결되어 있는 곳은 어디서나 이용할 수 있게 되었다[3]. 또한 정보 시스템은 웹의 프로토콜(protocol)인 HTTP(Hyper-Text Transfer Protocol)와 멀티미디어 문서인 HTML (Hyper-Text Markup Language) 그리고 외부 연동 방법인 CGI(Common Gateway Interface) 등의 인터넷 자원을 이용하는 웹 서비스들을 포함하면서 인트라넷 환경으로 발전하고 있다[1].

그러나 웹의 클라이언트-서버(client-server) 구조는 지속적인 상태 정보를 유지하면서 정보를 교환할 수 없으며, 브라우저는 단순히 HTML 문서를 표현해 주기 때문에 동적인 사용자 인터페이스와 지속적인 상태 정보를 요구하는 정보 시스템을 구축할 때 많은 문제가 발생한다. 이러한 문제를 해결하고자 썬마이크로 시스템즈에서는 객체지향 언어인 자바(Java)를 개발하였다. 자바 언어는 객체지향 개념을 지원하며 바이너리 코드가 아니라 중간 코드의 형태의 바이트 코드를 생성하므로 바이트 코드를 실행하는 인터프리터가 있는 곳이면 어디서나 실행될 수 있는 구조를 가지고 있다. 이러한 구조를 적극 활용하여 자바 인터프리터를 웹 브라우저에 삽입하고 자바 응용 프로그램을 브라우저에서 실행시킴으로서 정적인 웹에 동적인 기능을 불어넣고 클라이언트와 서버 사이의 동적인 상태 정보를 유지할 수 있게 해주었다.

웹을 중심으로 한 인터넷과 자바 못지 않게 또 하나의 중요한 환경으로 떠오른 것이 바로 분산

† 정회원 : 한서대학교 전산통계학과 조교수
†† 정회원 : 수원여자전문대학 정보처리과 교수
††† 정회원 : 혜전전문대학 전산과 조교수
†††† 정회원 안산전문대학 전산과 전임강사
††††† 정회원 LG산전연구소 주임 연구원

환경이다. 네트웍의 확산과 아울러 기존의 클라이언트-서버 환경을 적극적으로 발전시켜 여러 시스템을 지리적으로 분산되어 있는 곳에 분산시켜 각기 독립적인 기능을 수행하거나 필요할 때 협력하는 분산 시스템이 보편화되기 시작했다. 이러한 분산 시스템은 초기에 소켓과 RPC(Remote Procedure Call) 기능을 이용하여 개발되었으나 이후에는 DCE(Distributed Computing Environment)등의 통합 분산 환경을 통해 개발되었다.

그러나 기존의 이러한 방법은 분산 환경 하에서 다양한 하드웨어와 응용 소프트웨어, 운영 체계, 데이터베이스 등을 일관되게 연결해 주지 못했다. 이러한 상황을 해결하기 위하여 OMG Object Management Group)에서는 코바(CORBA: Common Object Request Broker)[7]를 발표하였다. 코바는 구현 언어나 개발 환경이라기보다는 객체 기술에 근거한 통합 기술이다. 따라서 실제 구현된, 또는 구현되는 언어에 상관없이 각 시스템들을 연결, 통합할 수 있는 방법을 제공한다.

지금까지 많은 개발자들은 각종 시스템 구축 시 기본 환경으로서 웹을, 웹 환경 하에서 동적인 시스템을 구축하기 위한 개발 언어로서 자바를, 이종의 다양한 시스템을 통합하기 의한 중간 기술로서 코바를 각기 사용해 왔다. 그리고 자바와 코바의 기능을 통합하면, 자바가 제공하는 유연성과 이식성, 코바가 제공하는 분산 객체 인프라를 결합함으로써 인터넷 환경에 웹 서버를 포함한 다양한 서버들과 자바의 이식성 있는 코드를 기반으로 클라이언트를 구축할 수 있는 장점이 있다.

본 고에서는 이와 같은 분산처리 시스템, 자바와 코바의 분산 환경과 이들의 연동 노력에 대하여 다루고자 한다.

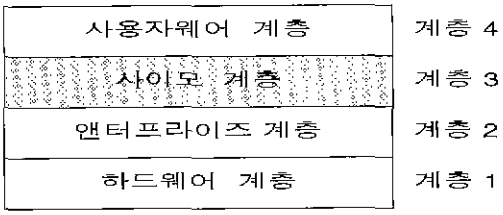
2. 분산 처리 시스템

분산 처리는 일반적으로 분산 컴퓨팅 시스템(distributed computing system)에 의해 제공되는 서비스로 여러 대의 컴퓨터를 네트웍으로 연결하여 컴퓨터 상호간에 업무적 기능들이 네트웍을 통해서 성취되는 형태를 의미한다. 기술적 관점에서는 컴퓨터 상호간에 주기억장치를 공유하지 않는 것으로, 정보 교환은 광역 변수(global variable)를 통해 전달되지 않고 네트웍을 통한 메시지 교환으로 이루어지는 것을 의미한다[4]. 사용자 관점에서의 분산 처리는 컴퓨터 통신망에 연결된 여러 자원들을 마치 사용자 혼자서 사용하는 것처럼 지원해주는 것이며, 사용자들에게 컴퓨터들 사이의 관계가 투명(transparent)하고 무관(irrelevant)하게 보이도록 만드는 것을 말한다.

2.1 클라이언트-서버 시스템

클라이언트-서버 기술은 분산 처리 기술의 특수한 모형으로 통신 망 기술의 발달과 개인용 컴퓨터나 워크스테이션 같이 성능도 높고 규모도 커지는 클라이언트 급의 컴퓨터들이 등장하면서 출현한 시스템 구성상의 새로운 기술이다. 이 기술은 선진 외국에서는 1980년도 말에 정보 공학(information engineering) 분야에서 발표되어 이미 정보 통신 관련 분야에서 활용되고 있다[2].

분산 처리 환경 하에서의 클라이언트-서버 응용 도구는 멀티미디어를 이용한 클라이언트-서버 형태의 응용 프로그램을 지원하며 분산처리 시스템 소프트웨어의 사이모(middleware)로서 기능을 수행한다. 클라이언트-서버 응용 도구에 대한 표준화된 참조 모델(reference model)은 없으나, 분산처리 시스템 소프트웨어 기술 측면에서 살펴보면 (그림 1)과 같은 4 계층으로 분류된다[8].



(그림 1) 클라이언트-서버 응용 개발 도구 참조 모델

이 그림에서 사이모 계층은 클라이언트-서버 응용 개발 도구의 가장 핵심 부분으로서, 데이터 관리 기능과 객체 관리기능, 그리고 통신 관리 기능으로 구성되어 있다. 객체 관리 측면에서는 사용자웨어에서 생성한 개체들을 관리하는 기능, 타 소프트웨어 사이에 데이터를 교환할 수 있는 기능이 있다. 통신 관리 측면에서는 윗 계층인 사용자 계층을 통하여 엔터프라이즈웨어 계층과 하드웨어 계층에 통신 방식으로 접근할 수 있도록 하는 기능이 있고, RPC와 DCE 등의 분산 시스템 소프트웨어를 접속하는 기능이 있다.

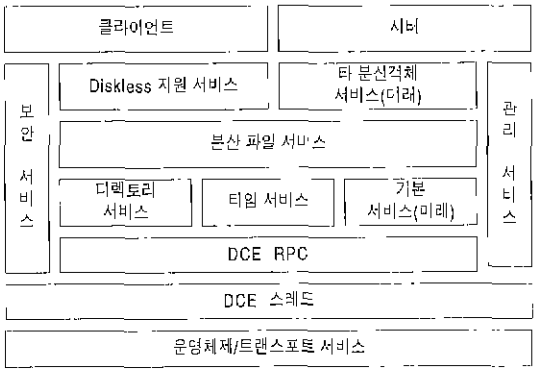
2.2 RPC 통신 사이모

통신 사이모 중 RPC는 가장 광범위하게 사용되었고 지금도 다른 미들웨어 시스템의 하부 구조로 이용되는 시스템이다. RPC는 사용자에게 원격지에 있는 함수를 호출할 수 있는 기능을 제공한다. 사용자는 RPC가 제공하는 IDL(Interface Definition Language)이라는 언어를 통해 원하는 함수를 작성한다. 이 때 IDL 컴파일러는 클라이언트 측에서 사용하는 함수와 서버에서 사용하는 함수를 함께 제공한다. 따라서 클라이언트는 자신에게 주어진 함수를 호출하면 이 호출은 자동으로 RPC 런 타임 라이브러리를 통해 서버에 위치한 함수가 호출된다. 해당 함수의 수행이 끝난 후 그 결과는 다시 클라이언트측에서 호출한 함수의 결과 값으로 전달된다.

2.3 DCE 통신 사이모

DCE는 분산 컴퓨팅 환경을 제공해 주는 사이모로서 이 기종으로 구성된 컴퓨팅 환경에서 분산 응용 시스템을 개발, 이용, 관리, 유지하는데 필요한 여러 가지 도구와 서비스를 제공해 주며 (그림 2)와 같은 구조이다. 이 그림에서는 DCE 구조와 그 기술적 구성 요소들을 보여 주며 응용 프로그램, 기존의 시스템 자원, 향후 기술 등과의 관계도 나타내고 있다. 이 구조에서 DCE의 주요 기능은 다음과 같다.

- 스레드(thread) : 하나의 프로세스내에 여러 개의 스레드를 만들어 관리하고 동기화 제어
- 디렉토리 서비스 : 클라이언트에게 이용 가능한 서버에 관한 정보 제공
- 타임 서비스 : 접속된 컴퓨터들에게 동기화된 시간 보장
- 보안 서비스 : 사용자, 컴퓨터, 서버 등의 등록을 관리하는 등록 서비스, 사용자 인증 서비스(authentication), 특정 객체의 권한 서비스(authorization service) 제공
- 화일 서비스 : 이 기종 컴퓨터들에 분산되어 있는 화일들을 하나의 디스크에 있는 것처럼 사용할 수 있게 지원, 접근 제어 목록(access control list) 유지



(그림 2) DCE 구조

DCE의 장점 중 하나는 서비스들 간의 결합성에 있다. 구성 요소들 자체가 잘 정의된 인터페이스를 가진 모듈이면서 잘 통합되어 있다. DCE의 여러 구성 요소들은 서로 다른 구성 요소들이 제공하는 서비스들을 사용하고 있다.

그러나 이러한 DCE의 RPC는 특정한 함수를 직접 호출하는 구조이므로 코바와 같이 특정 오브젝트의 특정 오퍼레이션을 호출하는 객체 지향적 개념이 결여되어 있어 분산 객체 환경에 적합하지 않다.

3. 자바의 분산 환경

자바는 기존의 프로그래밍 환경을 개선하여 간략화와 객체 지향성, 분산성, 보안성, 독립성, 이식성, 멀티스레딩 지원을 기본 개념으로 썬마이크로 시스템즈에서 개발된 객체지향 프로그래밍 언어이다. 기존의 다른 언어들 보다 이식성과 안정성이 뛰어나며, 또한 상업적 전략이었던 네트스케이프 웹 브라우저와의 결합으로 가장 인기 있는 인터넷 언어로 부상하였다[6].

자바는 기존의 널리 알려진 언어인 C++를 선택하고, 이에 복잡한 개념인 포인터 연산, 오버로딩, 다중 상속 등의 개념을 삭제하였으며, 자동 가베지 수거(garbage collection) 기능을 첨가함으로써 프로그래밍의 간략화를 가져왔다. 또한 객체인 애플렛(applet) 이용으로 재사용성을 높였으며, 네트웍 다운로딩을 통하여 객체의 분산성을 높였다. 또한 자바 가상 머신을 이용함으로써 실시간 컴퓨팅 기능을 향상하였다. 따라서 자바를 통해 클래스나 상속 등 다양한 객체지향 패러다임에 준하는 시스템을 개발할 수 있다. 자바가 갖는 프로그램 적인 명세 외에 자바는 인터프리터 언어로서 운영 체계에 상관없이 수행할 수 있다는 장점을 제공한다. 일단 작성된 자바 코드는 자바 컴파일러에 위해서 자바 바이트 코드로 변환된다. 이 코드는 로컬 컴퓨터나 네트웍을 경유하여 다른 지점에 위치한 컴퓨터에게 전달된다. 전달받은 자바 코드는 바이트 코드 로더에 의해서 로드되고 바이트 코드 검사기에 의해서 검사된 후 인터프리터를 통해 수행된다. 따라서 윈도우즈건 유닉스건 상관없이 바이트 코드를 수행하기 위한 환경이 있는 곳이면 어디서나 수행될 수 있다. 특히 자바 수행 환경을 내장하고 있는 웹 브라우저는 인터넷상의 어디서나 자바 코드를 수행하게 해준다. 바로 이 기능으로 인해 자바는 정적인 웹 환경에 동적인 생명력을 불어넣었으며 많은 개발자들은 운영 체계에 상관없이 개발된 코드를 어디에서나 수행할 수 있다는 장점이 있다.

이러한 자바의 장점을 이용하여 네트스케이프사에서는 웹을 기반으로 하는 분산 어플리케이션(application)을 지원하기 위하여 네트스케이프 ONE Open Network Environment)을 발표하였다. 네트스케이프 ONE은 새로운 것이 아니라 현존하는 인터넷 기술들을 총 망라하여 정리해 놓은 것이다. 클라이언트인 브라우저를 위하여 HTML, 자바스크립트, 자바가 지원된다. 서버와 클라이언트 간의 네트워크 요소가 내포되어 HTTP, SMPT, LDAP 등의 프로토콜이 지원된다. 서버 모듈을 구현하기 위하여 자바, 자바빈즈(Java Beans)가 지원된다.

지금까지 웹 서버에서 수행되는 서버 모듈은 대부분 CGI를 통해서 구현되었다. CGI는 C, C++, Perl 등과 같은 다양한 언어를 이용할 수 있다는 장점이 있지만 사용하기 힘들며 플랫폼(platform)마다 구별된다는 단점이 있다.

자바, 자바빈즈를 이용하여 CGI가 하던 역할을 한다면 앞에서 거론된 단점을 해결할 수 있다. 자바빈즈를 이용함으로써 쉽고 빠르게 원하는 분산 어플리케이션을 개발할 수 있다. 따라서 네트스케

이프 ONE은 대부분 서버 기능을 자바빈즈를 통해 이용할 수 있도록 다양한 컴포넌트들을 제공하고 있다. 데이터베이스, 보안, 디렉토리, 메일 등의 다양한 기능을 이용할 수 있는 컴포넌트들을 제공하여 자바빈즈는 플랫폼 독립적이면서 인터넷 기반의 분산 어플리케이션을 개발할 수 있는 환경을 제공한다.

플랫폼 독립적인 자바의 환경에 비하여 마이크로소프트 사의 액티브 X(Active X) 환경은 풍부한 기술, 수많은 사용자 및 재사용 가능 컴포넌트 등을 무기로 하여 자바의 분산 환경에 강력히 도전하고 있다. 자바의 환경과 액티브 X 환경은 웹 페이지에 포함된 프로그램인 플러그인(plug-in), 컨트롤, 애플릿 등을 자동으로 다운로드하여 클라이언트 시스템에서 실행하며, 소프트웨어 컴포넌트 들을 컨테이너에 넣음으로서 분산 어플리케이션을 구축한다는 측면은 같지만 <표 1>과 같은 차이점이 있다.

<표 1> 자바 환경과 액티브 X 환경 차이점

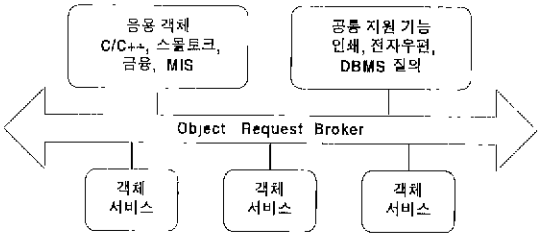
환경 / 비교기준	자바 환경	액티브 X 환경
개발자	썬마이크로 시스템즈	마이크로 소프트
브라우저	Navigator, Explorer, HotJava	Explorer
개발언어	애플릿을 자바 언어로 개발	컨트롤을 C++ 및 기타 언어로 개발
전송 자료 형태	바이트 코드	인텔 바이너리 코드
플랫폼 독립성	독립성 있음	C++, 독립성 없음
응용 수준	애니메이션, 자료 입.출력 수준의 표현	완전한 클라이언트 응용가능

마이크로소프트 액티브 X 환경은 인텔 바이너리 코드 환경에서만 실행되므로 플랫폼 독립적으로 수행되는 자바 환경이 분산 환경에 더 유리하므로 많은 개발자들이 자바 환경을 지원하고 있다.

그러나 자바는 분산되어 있는 애플릿들 간의 연결이 미비하고 데이터베이스에 대한 연결 기능이나 기존의 레거시 시스템/데이터(Legacy system/data)의 수용이 불가하다. 또한 자바 환경 그 자체로는 클라이언트-서버 패러다임을 지원하지 않으므로 자바가 제공하는 수행 환경의 장점에도 불구하고 분산 어플리케이션을 개발하는 데에는 많은 문제점을 안고 있다.

4. 코바의 분산 환경

1989년 4월, 현재 존재하는 기술을 바탕으로 분산 어플리케이션들을 결합하기 위한 객체지향 표준을 제정하기 위해 OMG라는 비영리 단체가 탄생하였다. 이 단체에는 현재 마이크로소프트 사를 포함한 700여 개의 컴퓨터 관련 단체들이 참가하여 객체지향 기술을 기반으로 이종의 분산된 환경에서 어플리케이션들이 서로 통합 할 수 있는 시스템 통합 표준 기술을 탄생시켰다. 이 표준이 바로 객체 관리 구조(OMA: Object Management Architecture)이다. OMA는 어플리케이션 사이의 결합뿐만 아니라 객체의 생성, 소멸에서부터 저장, 트랜잭션 기능에 이르기까지 분산 객체 환경에 필요한 모든 서비스를 총칭하는 것이다. OMA의 특징은 특정 기술을 이용하여 구체화된 제품을 만드는 것이 아니라, 그 기술을 표현한 규격을 제시하여 업체들이 각자 구현하게 한다. OMA는 (그림 3)과 같이 응용 객체, 공통 지원기능(common facilities), 객체 서비스, 객체 요청 중개자(Object Request Broker)로 구성되어 있다.



(그림 3) 객체 관리 구조(OMA) 구성도

- 응용 객체 : OMG에서 정의된 규격으로 작성한 어플리케이션이나 클래스이다. 예를 들면 워드프로세서 등을 말한다.
- 공통 지원기능 : 어플리케이션 사이에 객체를 공유하거나 OMA를 기반으로 하는 어플리케이션을 쉽게 생성하기 위한 구성 요소이다. 예를 들면 객체 사이의 자료 교환, 클립보드(clipboard), 서비스 인터페이스로 복합문서 지원, 객체 연결과 삽입이 있다.
- 객체 서비스 : 모든 객체가 공동으로 유용하게 사용할 수 있게 OMG에서 정의한 인터페이스이다. 예를 들면 객체 이름관리(naming) 서비스, 객체 사이의 사건(event)을 교환하기 위한 사건 서비스, 객체 저장을 위한 서비스 등이 있다.
- 객체 요청 중재자 : 마치 컴퓨터 하드웨어의 시스템 버스처럼 객체 사이의 요청/응답을 전달하는 메커니즘이다.

이들 OMA 기능 중 코바는 컴퓨터 내부의 버스처럼 서로 다른 어플리케이션들 사이의 버스 역할을 하는 모듈로서 OMA의 객체간 통신을 담당한다. 결국 코바는 OMA의 한 부분이고 ORB는 코바의 핵심 기술이다. 현재 OMG는 1990년 OMA를 발표한 이래 지금까지 코바 2.0 스펙을 발표하였으며 3.0을 준비중에 있다.

코바를 이해하는 데 있어 중요한 점은 코바는 서버라는 용어를 사용하지 않는다는 것이다. 왜냐하면 코바는 객체지향 개념을 바탕으로 하여 원격지의 클라이언트가 원격지에 있는 서버를 호출하는 것이 아니라 객체의 메소드를 호출함으로써 서비스가 이루어진다.

따라서 코바에서는 서버가 아니라 구현 객체(Object Implementation)라는 용어를 사용한다. 이때 고려해 볼 점은 사용자가 클라이언트와 구현 객체 사이의 통신 부분을 직접 관여해야 하는가에 대한 여부와 구현시 사용하는 프로그래밍 언어에 따라 각기 다르게 작성해야 하는가 하는 문제이다.

이러한 점을 해결하고자 코바에서는 IDL(Interface Definition Language)이라는 표준 언어를 제공한다. 사용자는 IDL을 사용하여 원하는 어플리케이션을 작성할 수 있다. 일단 작성된 IDL은 코바에서 제공하는 IDL 컴파일러를 통해 컴파일된다. 이 결과 원하는 프로그램 언어로 작성된 클라이언트 코드와 구현 객체 코드를 얻을 수 있다. 이 코드는 C나 C++ 또는 자바 등의 사용자가 원하는 형태로 제공되며, 통신을 위해 필요한 모든 기능이 자동적으로 포함된다[5].

코바의 클라이언트에는 클라이언트 스터브와 어플리케이션 수행시 원하는 메소드를 호출하는 동적 호출(dynamic invocation)기능을 통해 구현 객체를 호출할 수 있다. 클라이언트에 의해서 요청된 서비스는 코바를 통해 원격지의 구현 객체에 전달되고 구현 객체에 의해서 처리된 결과는 다시 클라이언트에게 반환된다. 이때 호출시 전달되는 정보로는 호출시 대상이 되는 객체와 메소드, 전달 인자 등이 있다.

현재 코바는 다양한 비즈니스 분야에서 시스템 통합을 위한 기반 기술로서 각광받고 있다. 특히 코바 IDL이 ISO/IEC를 통해 표준으로 제정되면 더 많은 분야에서 개방형 통합 기술로 채택될 것이다.

5. 자바와 코바의 연동

인터넷, 인트라넷이 보편화된 환경에서 코바를 웹에 연결하기 위한 방법으로 기존의 CGI를 이용하여 게이트웨이를 구성하는 CGI 게이트웨이 방법은 여전히 CGI의 문제를 내포하고 있어 바람직하지 못하다.

다른 방법으로는 코바 IDL 컴파일러를 확장하여 해당 IDL 화일을 웹에 적합한 코드로 생성해 주는 방법이 있다. 이 방법은 코바 IDL을 그대로 사용할 수 있다는 장점이 있다. 이 방법에서 기존의 웹 환경의 클라이언트-서버 사이 통신은 IIOP (Internet Inter ORB Protocol)를 사용한다. IDL은 본래 구현 언어 중립의 인터페이스 정의어이다. 따라서 해당 인터페이스를 자바로 변환하는 과정이 필요하다. 이 과정은 코바에서 자동적으로 제공한다. 또한 클라이언트와 서버는 웹에서 웹의 프로토콜인 HTTP를 사용하지 않고 코바가 제공하는 객체간 통신 프로토콜인 IIOP를 통해서 작동한다.

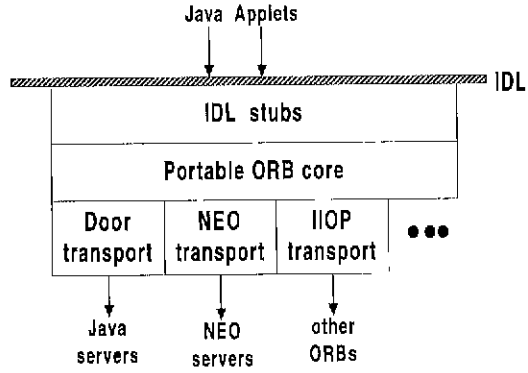
구축 환경은 기존의 웹 환경과 동일하다. 단지 다른 점은 코바 IIOP를 지원하는 웹 브라우저와 코바와 연동 가능한 웹 서버에 있다. 개발된 클라이언트 프로그램은 코바 IIOP를 지원하는 웹 브라우저에 HTTP를 통해 다운로드 된다. 다운로드된 프로그램은 IIOP를 통해 코바와의 서비스를 개시하게 된다.

이 때 유의할 점은 다운로드된 프로그램 역시 자바 애플릿이기 때문에 자바 애플릿이 갖고 있는 보안상의 제약, 특히 다운로드받은 호스트에만 연결을 설정할 수 있는 제약에 지배된다. 따라서 기본적으로 다운로드된 웹 서버를 통해 서비스 연결을 시도한다.

결국 웹 서버는 클라이언트가 요청한 서비스를 네트워크 상의 다른 곳에 위치한 코바에게 전달하기 위한 방법을 갖고 있어야 한다. 이 환경에서는 코바 서비스 외에도 기존의 웹 서비스 CGI 인터페이스와 HTML 문서 서비스를 제공한다. 현재 웹 환경을 지원하는 많은 코바 제품들이 소개되고 있다. 다음은 해당 제품들이다.

5.1 자바 IDL

자바 IDL은 자바소프트에서 개발한 것으로 자바 클라이언트와 응용 서버 사이에 IDL을 사용하여 접속할 수 있는 언어이다. 자바 IDL은 분산 환경에서 C또는 C++, 스몰토크 처럼 언어나 컴파일러 등에 상관없이 코바 응용 프로그램을 작성할 수 있게 해주는 역할을 한다. 즉 IDL로 작성된 코바 응용프로그램은 IDL 컴파일러를 통하여 원하는 프로그래밍 언어에 적합한 코드로 변환된다. 사용자는 이 코드를 이용하여 원하는 어플리케이션을 작성한다. 자바 IDL은 (그림 4)와 같이 포터블(portable) 자바 ORB 코어(core)를 근간으로 하여 만들어진 것으로서 포터블 자바 ORB 코어는 현재 자바 서버들 사이에 연결을 위하여 TCP/IP 프로토콜을 이용한 Door ORB와 다른 ORB와의 연결을 위한 IIOP 프로토콜 모듈, 자바 IDL 시스템과 직접적으로 연결하기 위한 ORB를 개발하였다.



(그림 4) 포터블 자바 ORB 코어

5.2 썬소프트의 네오(NEO)

썬소프트의 제품은 네오 2.0이다. 썬소프트는 네오를 통해 더욱 유연한 네트워크 기반의 기업 컴퓨팅 환경을 구축할 수 있고, 비즈니스 요구에 부응하며, 공유된 서비스를 이용해 비용 절감을 꾀할 수 있게 되다고 홍보하고 있다. 또한 코바 2.0의 기능과 IIOP도 지원된다. 네오는 크게 운영 환경, 개발환경, 시스템 관리도구의 세 개 분야로 나눠진다. 네오에는 기존 제품에서 찾아볼 수 없는 새로운 개념과 특성이 있다. 우선 네트워크 객체라는 개념이 도입됐다. 이는 높은 모듈성, 공유 가능한 소프트웨어 콤포넌트라는 객체의 장점을 네트워크 상에 최대한 활용하는 것으로, 이를 통해 공유 서비스가 가능해진다.

썬소프트는 네오가 인터넷을 기반으로 하고 있다는 점과 이에 맞는 응용프로그램을 쉽게 제작 가능하다는 것을 장점으로 내세우고 있다. 특히 OMG의 표준안인 코바를 지원하며, 코바를 구현하는 IDL을 'Java-to-IDL'로 확장 가능하다는 것, 그리고 기존의 웹 환경을 뛰어넘어 새로운 인터넷 서비스를 개발하도록 하는 Joe 2.0이 주목할 만하다.

Joe에서 강조되는 것은 기존의 웹 환경에 대한 혁신이다. 웹 환경의 혁신은 코바를 통한 분산형 객체 시스템의 구현으로 압축해서 설명할 수 있다. 여기에는 인터액티브한 웹 서비스 구현과 웹을 이용한 클라이언트-서버 환경을 구축하기 위한 세 가지 기능이 포함된다.

우선 양방향성이다. Joe는 CGI 처럼 새롭게 페이지를 구성하지 않고 직접 양방향적으로 동작하도록 한다. 둘째, 클라이언트-서버 시스템을 다시 구축하지 않고 기존의 시스템을 그대로 사용할 수 있도록 한다. 셋째, 클라이언트 쪽의 복잡한 세팅 과정을 없애 사용자도 쉽게 이용할 수 있도록 하였다. Joe의 이러한 특성이 웹 서비스를 더욱 지능화 할 수 있으며, 아주 편리하게 인트라넷을 구축할 수 있도록 한다.

6. 결 론

본 고에서는 최근에 인터넷의 대중화와 더불어 각광 받고 있는 분산처리 시스템, 자바와 코바의 분산 환경과 이들의 연동에 대하여 다루었다. 자바와 코바의 기능이 통합되면, 자바가 제공하는 플랫폼 독립성과 코바가 제공하는 분산 객체 인프라를 결합함으로써 기존의 시스템과 향후 개발될 시스템의 기반 기술로서 중요한 위치를 차지할 것이다. 코바의 경우 여러 회사들이 참여하여 표준을 만들었지만, 자바는 썬의 독점적인 기술이기 때문에 통합 과정에서 문제가 발생할 수 있는 점이 우려된다. 비록 코바에 대응하는 마이크로소프트 액티브 X 환경이 인텔 바이너리 코드 플랫폼에만 국한된다는 제약이 있지만 마이크로소프트의 시장 장악력을 감안하면 액티브 X도 분산 어플리케이션 개발에 많은 영향을 미칠 것으로 예상된다. 이에 따라 코바 표준과 개발사들은 액티브 X의 COM(Component Object Model)/OLE와 연동에 대하여 고민하고 있으며, 이 OLE를 코바 제품들에게 적극적으로 반영하고 있다.

참고문헌

[1] 김용운, "WWW(World Wide Web)의 현재와 미래", 한국통신학회지, 제 14권, 제 4호, pp. 402-414, 1997. 4.

[2] 인소란, "고객-서버 응용 개발 도구", 정보과학회지, 제 14권, 제 1호, pp. 32-39, 1996. 1.

[3] 진영민, "인터넷과 서비스 발전방향", 한국통신학회지, 제 14권, 제 4호, pp. 396-401, 1997. 4.

[4] Amjad Umar, Distributed Computing, Prentice-

Hall, 1993.

[5] Eric Evans, Daniel Rogers, "Using Java Applets and CORBA for Multi-User Distributed Applications", IEEE Internet Computing, Vol. 1, No. 3, pp. 43-55, May-June 1997.

[6] James Gosling, Henry McGilton, The Java Language Environment: A White Paper, Technical Report, Sun Microsystems, October 1995.

[7] Object Management Group, "The Common Object Request Broker: Architecture and Specification-Revision 2.0", (OMG 96-08-04), Object Management Group, Framingham, Mass., 1996; http:// www. omg.org/corba/corbiiop.htm.

[8] Paul E. Renaud, Introduction to Client-Server System, John Wiley & Sons, Inc., pp. 66-94, 1993.

김 홍 윤

1982년 인하대학교 전자계산학과 (이학사)
1984년 인하대학교 전자계산학과 (이학석사)
1996년 인하대학교 전자계산학과 (이학박사)

1995년-현재 한서대학교 전산통계학과 조교수
관심분야 : 컴퓨터 네트워크, 한국어 정보처리

이 재 용

1985년 인하대학교 전자계산학과 (이학사)
1990년 인하대학교 전자계산학과 (이학석사)
1995년-현재 인하대학교 전자계산학과 박사과정

1993년-현재 수원 여자 전문대학 정보처리과 교수
관심분야 : 망관리, 분산 시스템

황 교 철

1988년 인하대학교 전자계산학과 (이학사)
1991년 인하대학교 전자계산공학과 (공학석사)
1995년-현재 인하대학교 전자계산학과 박사과정

1991년-1994년 LG산전연구소 주임 연구원
1994년-현재 혜전전문대학 전산과 조교수
관심분야 : 컴퓨터 네트워크, 인터넷 보안, 분산 시스템

한 성 택

1988년 인하대학교 전자계산학과 (이학사)
1991년 인하대학교 전자계산공학과 (공학석사)
1995년-현재 인하대학교 전자계산학과 박사과정

1991년-1994년 LG산전연구소 주임 연구원
1995년-현재 안산전문대학 전산과 전임강사
관심분야 : 지능망 설계, 분산 시스템

홍 종 준

1991년 인하대학교 전자계산공학과 (공학사)
1993년 인하대학교 전자계산공학과 (공학석사)
1997년-현재 인하대학교 전자계산학과 박사과정

1993년-현재 LG산전연구소 주임 연구원
관심분야 : 망관리, 인터넷 보안, 분산 시스템