웹 환경에서 확장된 객체지향형 미들웨어(CORBA/JAVA)를 이용한 멀티미디어 정보 검색 시스템 구현

이 원 중† · 안 길 수† · 주 수 종††

요 약

최근, 웹(World Wide Web) 환경에서 효율적인 인터넷 서비스를 위해서 정적 서비스보다는 동적 서비스에 대한 다양한 요구사항들을 만족시키고자 많은 연구자들의 노력이 시도되고 있다. 웹 상에 있는 서버들로부터 정보를 얻기 위하여, CGI는 분산 응용들 간의 상호연동의 한 방법으로 제안되었다. 이 방법에서 서버는 서비스 객체들간의 정적 바인딩 인한 과중한 처리 부담 및 네트워크 트래픽 부하를 극복할 수 없는 문제점들을 가지고 있다. 이러한 이유에서 본 논문에서는 기존의 CGI를 사용하는 대신에 확장된 객체 지향형 미들웨어를 사용하므로 웹 상에서 분산자원을 효율적으로 지원할 수 있도록 한다. 목표 시스템으로 CORBA와 자바의 매핑을 위한 기능들을 확장하고, 분산객체의 재사용성을 채택한 멀티미디어 정보 검색 시스템을 웹 상에서 구현하였다. 여기에서 클라이언트 모듈들은 임의의 서버에게 정보를 요청하기 위한 CORBA 미들웨어 접속용으로 서버 모듈은 클라이언트의 요청을 CORBA/JAVA 미들웨어를 통해 받아 웹 서버내의 멀티미디어 정보를 검색하도록 각각 구현되었다.

An Implementation of The Multimedia Information Searching System Using Extended Object-Oriented Middleware(CORBA/JAVA) on Web Environments

Won-Jung Lee† · Gil-Sou Ahn† · Su-Chong Joo††

ABSTRACT

Recently, many researchers have tried to satisfy with various requirements for dynamic services rather than static services for the efficient internet services on Web environments. The existing CGI(Common Gateway Interface) approach had proposed as a method of the interactions between distributed applications for obtaining information from servers on Web. However, this method has problems which servers can not overcome system and traffic overheads due to static binding between service objects. For this reason, it is for our paper to effectively support distributed resources on Web by using the extended object-oriented middleware(CORBA/ JAVA) instead of the existing CGI.

As our target system, we implemented the multimedia information searching system is adapted with reusability of distributed objects and extended functions for CORBA/JAVA mapping on Web. Here, client modules can be connected to CORBA middleware for requesting information to an arbitrary server, and server modules are implemented for accessing multimedia information located in Web server via CORBA/JAVA middleware from clients' requests

※ 본 논문은 '97년도 원광대학교의 교내연구비의 지원에 의해서 연구됨.

† 준 회 원 : 원광대학교 대학원 컴퓨터공학과
†† 준 회 원 : 원광대학교 컴퓨터공학과 부교수

논문접수 : 1998년 2월 17일, 심사완료 : 1998년 5월 4일

1. 서 론

웹은 인터넷 상에서 분산된 정보와 서비스를 사용자에게 편리하고 쉽게 접근 가능하도록 제공함으로 써, 웹에 대한 수요와 관심은 놀라울 정도로 증가해 왔다. 이에 따라 웹 기술의 변화는 초기에 웹 서버에 의해 이기종의 자원을 HTML의 형식으로 전송하는 형태였고, 이후에 정적 HTML 문서 외에 외부자원을 웹 브라우저에게 지원하기 위해서 CGI(Common Gate-way Interface)가 이용되어왔다. 그러나 CGI는 서버의 과중한 부하와 오버헤드가 많고, 보안 유지가 어려우며, 대화형 프로그램의 개발이 어렵다는 단점이 있다. 이러한 문제점을 해결하기 위해 썬 마이크로시스템즈는 객체지향 프로그램 언어로 써, 어떠한 플랫폼에서도 실행할 수 있는 자바 애플릿(JAVA Applet)을 웹 브라우저에 내장시켜 대화형 응용 프로그램 작성을 용이하게 하는 자바를 내놓았다[4, 8, 9, 11].

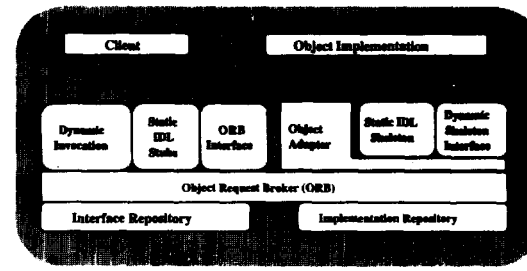
한편 분산 시스템 분야에서는 최근 소프트웨어와 하드웨어의 이질성을 해결하고 객체 지향형 서비스를 지원하는 미들웨어에 대한 연구가 OMG(Object Management Group)를 중심으로 하여 활발히 진행되고 있다. OMG에서는 분산 객체 컴퓨팅 표준 구조인 OMA (Object Management Architecture)를 제안하고, CORBA(Common Object Broker Architecture) 사양을 발표하였으며, 또한 1994년에 발표된 CORBA 2.0에서는 다른 ORB상의 상호운용성(Interoperability)을 위해 IIOP(Internet Inter-ORB Protocol)를 제시하였다[2, 12].

본 논문은 IIOP와 HTTP를 통해 CORBA와 웹을 접목함으로 써 웹 상에서 구현객체를 요구하여 구현 객체로부터 서비스를 받을 수 있는 CORBA의 원리를 이용하고, 자바가 실행 가능한 웹 브라우저에서 구현객체(멀티미디어 정보검색)를 사용하여 멀티미디어 정보를 검색하는 시스템을 설계 및 구현하는데 목적이 있다.

본 논문의 제2장에서는 기본적인 CORBA의 구조와 기능에 대해서 살펴보고, 웹 상에서 분산 객체들간의 상호운용성에 대해 알아본다. 제3장은 기존의 CORBA와 웹의 연동방법들을 기술하며, 제4장은 자바를 이용한 CORBA와 웹의 연동방법을 확장하여 적용한 멀티미디어 정보 검색 시스템을 구현한다. 마지막으로 제5장에서 결론과 추후 연구 방향을 기술한다.

2. CORBA 구조와 기능

CORBA는 분산 객체 환경에서 응용 프로그램의 상호운용성을 위하여 정의한 객체 관리를 위한 구조로서 분산 객체 구조들의 위치와 구현 방법에 관계없이 엑세스가 가능하도록 완전한 투명성을 제공한다. 또한 단일 시스템 또는 분산 시스템들 내에 구현된 객체들간의 바인딩 기능과 요구에 대하여 적절한 응답을 서로 주고받을 수 있는 peer-to-peer 기능을 제공한다. 아래 (그림 1)은 분산 환경을 지원하는 객체지향 미들웨어인 CORBA의 핵심(Core)과 ORB(Object Request Broker)와의 인터페이스 관계를 나타낸다[2].



(그림 1) CORBA 구조
(Fig. 1) The CORBA Architecture

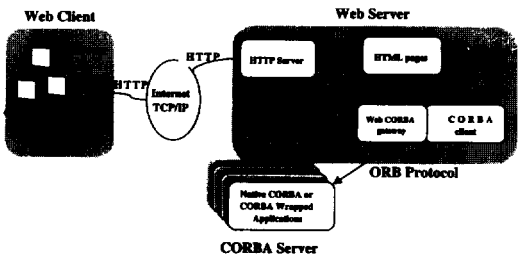
CORBA의 클라이언트는 클라이언트 스터브와 응용 프로그램 수행 시, 원하는 메소드를 호출하는 동적 호출(Dynamic Invocation) 기능을 통해 구현객체를 호출할 수 있다. 일단 클라이언트의 호출이 발생하면 통신을 담당하는 ORB를 통해 구현객체에게 전달된다. 구현객체(Object Implementation)쪽의 객체 어댑터(Object Adapter)는 기본적으로 해당 객체를 호출 가능하게 생성하고 원하는 호출에 해당 메소드를 실행시킨다. 즉, 클라이언트로부터 요청된 서비스는 ORB를 통해 원격지의 구현객체에 전달되고 구현객체에 의해서 처리된 결과는 다시 클라이언트에게 반환된다. 이때 호출 시 전달되는 정보로는 대상 객체, 메소드, 파라미터 등이 있다. 또한, CORBA의 표준언어인 IDL (Interface Definition Language)[4, 6]은 CORBA 객체의 인터페이스를 정의하는데 사용된다. IDL로 객체의 인터페이스를 정의함으로써 객체 구현언어, 객체 위치, 운영체제, 컴퓨터 구조, 네트워크 구조로부터 독립된 인터페이스를 갖는다. 실제 IDL에서 가장 기본이

되는 단위는 인터페이스이다[3]. 이 인터페이스는 애트리뷰트와 메소드로 구성된다. 대부분의 문법은 자바 구문과 비슷하나 파라미터_타입의 경우 클라이언트와 구현객체 사이의 이동방향에 따라 in(요청에서 서비스), out(서비스에서 요청), inout(양방향)으로 선언하여 사용할 수 있다. IDL로 작성된 모든 인터페이스 정보는 인터페이스 저장소에 저장되고, 저장된 정보를 사용하여 클라이언트가 요구한 서비스 요청에 대한 인터페이스 정보를 조회할 수 있으며, 해당요청이 정당한지 검사할 수 있다. 또한 IDL 인터페이스는 IDL 컴파일러를 통해 컴파일 될 때, 인터페이스와 구현정보가 각각 인터페이스 저장소와 구현저장소에 저장된다. 이렇게 저장된 정보는 각 클라이언트와 구현객체가 실행될 때 이용된다. IDL을 이용하므로 써 기존 CGI와 달리 인터페이스의 다중 상속성의 지원 및 동적 호출 메카니즘을 제공할 수 있다.

OMG에서는 웹 상에서 서로 다른 ORB 간의 프로토콜 관계를 제안한 것으로 필수적인 요구 사항으로 GIOP(General Inter-ORB Protocol)와 TCP/IP를, 선택 사항으로는 ESIOP(Environment Specific Inter-ORB Protocol)를 제안했다[17]. GIOP는 ORB들 사이의 통신을 위한 일련의 메시지 형태와 공통의 데이터 표현을 명시한다. GIOP로 명시된 클라이언트의 요구사항은 TCP/IP를 비롯해 임의 네트워크를 공유해 다른 ORB에게 바로 전달된다[7, 15]. CORBA 2.0에서 정의한 TCP/IP를 IIOP(Internet Inter-ORB Protocol)라고 기술하였다. IIOP는 GIOP로 정의된 메시지가 TCP/IP 상에서 어떻게 전달되는지를 사양을 정의하고 있다. 인터넷 상에서 ORB의 기본 네트워크 프로토콜인 IIOP를 사용함으로써 웹 상에서 멀티미디어 정보 검색을 제공하는데, 이때 서비스 객체는 특정 서버 시스템의 위치에 관계없이 서비스를 받을 수 있다. 이러한 특성을 이용하여 본 논문에서는 IIOP와 2층(2 tiers) 구조의 검색 시스템을 구성하였고, 검색 서비스 기능들을 구현한 객체구현(Implementation Object)을 웹 서버 상에 설치하여 멀티미디어 정보 검색 시스템을 구현하였다.

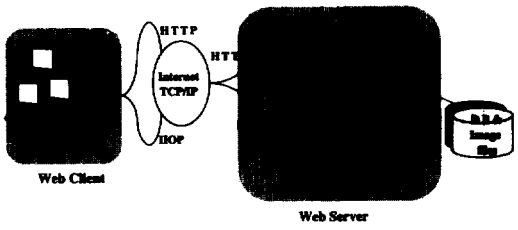
3. CORBA와 웹의 연동 방법

지금까지 객체지향형 미들웨어인 CORBA를 이용한 웹과의 연동 방법에는 CGI를 이용하는 방법과 자바를 이용하는 방법으로 2 가지가 있다[14]. 먼저 CGI를 이용하는 방법은 (그림 2)와 같이 CGI 기반의 게이트웨이를 통한 웹과 CORBA의 일반적인 연동 구조이다. 클라이언트의 요구가 있을 때, HTTP 서버에 의해 호출된 게이트웨이는 ORB를 통하여 CORBA 서버 객체를 호출하여 서비스를 받는다. 이때 CGI의 병목현상은 물론, 이러한 방법은 CORBA 객체를 직접 접근이 불가능하므로 HTML 문서를 동적으로 검색 서비스를 하거나 HTTP 프로토콜 상태유지를 위한 추가적인 처리가 요구되어야하는 단점이 있다[8].



(그림 2) CGI를 이용한 CORBA와 웹의 연동
(Fig. 2) The Interactions Between CORBA And Web Using CGI

그러나 자바를 이용한 CORBA와 웹과의 연동 방법은 다음 (그림 3)과 같이 웹 클라이언트와 웹 서버간에는 HTTP와 IIOP를 통하여 서로 통신하며, 웹 서버에서 제공하는 HTML 문서나 CORBA 객체를 사용할 수 있다. 이 방법은 CGI에 의한 CORBA의 연동 시 발생되는 문제점들을 해결할 수 있는 다음과 같은 장점을 제공한다. 첫째, 자바를 이용한 방법에서는 웹 클라이언트가 직접 웹 서버를 동적으로 호출이 가능하므로 기존의 CGI에서 정적 호출에 의해 발생되는 트래픽 부하의 병목 현상을 회피할 수 있다. 둘째, CORBA에서는 ORB를 통해 객체들간에 통신이 이루어지므로 써 하나 또는 그 이상의 객체들은 여러 서버들 상에서 서로 공유되고 실행될 수 있다. 셋째, 현재의 자바 애플릿은 객체를 호출하기 어려우나 자바를 이용한 CORBA와 웹의 연동 방법에서는 자바 애플릿이 네트워크를 통하여 다른 언어로 구현된 다른 객체를 호출할 수 있다[14]. 위와 같은 장점을 채택하기 위해서, 본 논문에서는 CGI를 이용하는 방법보다 자바를 이용한 CORBA와 웹의 연동 방법을 제 4장에서 확장시켜 본 시스템의 객체지향형 미들웨어로 사용한다.



(그림 3) 자바를 이용한 CORBA와 웹의 연동
(Fig. 3) The Interaction Between CORBA And Web Using JAVA

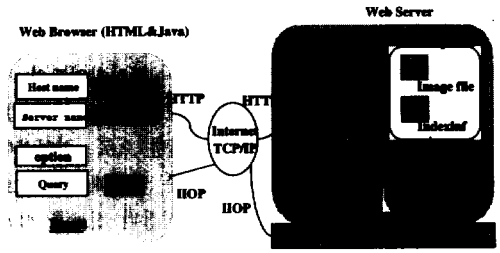
4. 멀티미디어 정보 검색 시스템의 구현

4.1 설계 및 구현환경

본 논문에서 IIOP와 HTTP를 통하여 멀티미디어 정보 검색을 하는데 있어 CORBA의 구현객체를 사용함으로서 객체 구현언어, 객체위치, 운영체제, 컴퓨터구조, 네트워크 구조로부터 독립된 인터페이스를 갖는 점을 활용하고, 기존의 널리 사용하는 웹을 이질적인 운영체제 와 플랫폼에서 사용할 수 있는 CORBA의 장점을 접목하여 설계한 웹 상에서의 멀티미디어 정보 검색 시스템 환경은 (그림 4)와 같으며 검색을 위한 구현객체가 CORBA 서버를 통해서 외부의 화일 시스템에 접근하도록 설계하였다. 화일 시스템은 웹 상에서 지원될 멀티미디어 정보를 보관하도록 하였다. 이 정보를 얻기 위해서 검색용 객체를 생성시키므로 써, 웹 브라우저는 해당 객체에 접근하여 멀티미디어 정보를 요청할 수 있도록 하여 요청에 대한 결과인 멀티미디어 정보를 웹 상에서 서비스를 받을 수 있게 설계하였다.

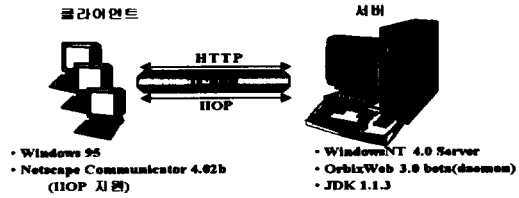
구현환경은 (그림 6)과 같이 웹 브라우저는 자바가 실행 가능한 브라우저로(Netscape Communicator 4.2b〈IIOP 지원〉)를 사용했고, 웹 서버는 일반적인 웹 자원과 이미지에 대한 정보를 검색하는 객체에 대한 요청과 웹 브라우저의 이벤트에 관한 처리 및 결과를 애플릿으로 처리한다. CORBA 서버는 웹 서버같이 수행상태에 있으면서 요청되는 서비스의 응답결과를 IIOP를 통하여 웹 브라우저에 상에 보내고 웹 브라우저부터 요청을 받는 기능을 수행한다.

여기에서 우리는 클라이언트와 서버의 접속을 위한 미들웨어인 CORBA를 IONA사의 OrbixWeb 3.0을 이용하였다. OrbixWeb은 사용한 이유는 OMG에서 제안한 CORBA 표준안을 따르고 있으며, 인트라네트 또는 인터넷상에서 응용 개발 및 통합 시, 미들웨어로서 웹



(그림 4) 설계 환경
(Fig. 4) The Design Environment

환경 및 클라이언트/서버 환경을 지원 할 뿐 아니라 CORBA의 기능들과 자바 애플릿들을 이용하여 객체들 간의 동적 바인딩으로 시스템들간에 유지부하를 최소화 할 수 있다. 또한, 웹 브라우저에게 자바 크래스들을 제공하여 동적으로 다운로딩할 수 있기 때문이다. 웹 브라우저 상에서 결과를 보이기 위하여 GUI(Graphic User Interface) 화면과 구현객체들에 대한 인터페이스들과 구현객체들의 컴파일은 JDK 1.1.3을 사용했다[11].



(그림 5) 구현 환경
(Fig. 5) The Implementation Environment

구현객체는 웹의 HTTP 와 IIOP를 기본 프로토콜로 사용하며, CORBA IDL로 작성된 인터페이스를 자바로 매핑하기 위해 IONA사의 OrbixWeb 3.0을 사용하였다. 본 멀티미디어 정보 검색 시스템에서 검색 서비스를 제공하기 위해서는 서버객체와 인터페이스는 OrbixWeb에서 제공하고 있는 IDL을 이용해서 (그림 6)와 같이 정의하였다.

```
// In file index.idl
interface index {
    typedef sequence<string> path_name;
    path_name image_indexing
            ( in string query_one,
              in string query_two,
              out long result_count ) ;
};
```

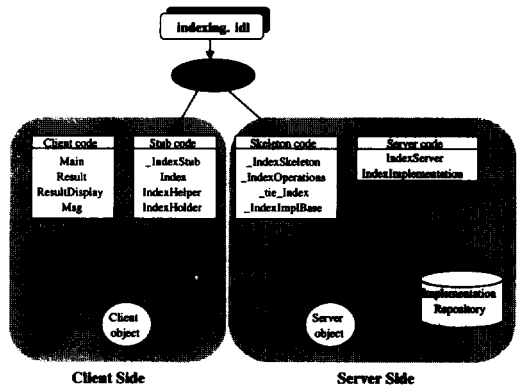
(그림 6) 서버객체의 IDL 작성
(Fig. 6) The IDL Coding of Server Object

위 그림에서 기술한 IDL에서 path_name은 검색이 완료된 후, 질의에 대한 결과 값(즉, 화일의 경로)을 반환하기 위한 타입을 나타내고, 이를 sequence 구조로 선언하여 재 정의하였다. sequence 구조는 IDL 컴파일 시, 자바 배열구조로 생성된다. 실제적으로 멀티미디어 정보 검색 서비스를 제공할 multimedia_indexing 오퍼레이션의 query_one 파라미터는 private key를 넘겨주며, query_two는 검색하고자 하는 content 정보를 넘겨준다. IDL에서는 함수 반환 값 외에 다수의 결과를 받아볼 수 있도록 out형의 파라미터 기능을 제공하고 있는데, 검색 결과물의 수를 클라이언트 측으로 반환해 주는 result_count 파라미터를 out 타입으로 선언하였다. 결과 값으로 오퍼레이션 수행 후 검색된 멀티미디어 정보 값이 path_name에 넘겨진 후, 웹 브라우저에서 클라이언트 애플릿을 통해 가공 멀티미디어 정보를 보여진다.

(그림 7)은 IDL 작성 후, 클라이언트와 서버용 객체들을 생성하기 위한 컴파일 단계를 나타낸다. IDL 컴파일 하면 구현객체와 바인딩에 필요한 자바 화일들이 생성되고, 웹 브라우저에서 구현객체와 연결이 설정되고 결과를 보여주는 GUI 애플릿을 구현한다. (그림 7)에서 나타내고 있는 부분에서 자동으로 생성된 스터브 코드와 스켈레톤 코드 내에 있는 화일들의 역할은 다음과 같다. 클라이언트 측과 매핑으로 Index는 IDL 인터페이스의 자바 클라이언트 측을 정의한 오퍼레이션를 가지고 있는 자바 인터페이스이고, _IndexStub는 인터페이스 Index에 정의된 오퍼레이션들이 구현된 자바 클래스이다.

이 클래스는 클래스가 서버 측으로 오퍼레이션 호출을 할 수 있도록 하는 기능을 제공한다. 또한 서버 측과 매핑으로 _IndexSkeleton은 서버에 들어오는 요청들을 분배시키기 위해서 OrbixWeb에 의해 내부적으로 사용되는 자바 클래스는 구현객체에 요구한다. 응용 개발자들은 이 클래스를 이해할 필요는 없다.

_IndexOperations는 자바 오퍼레이션 내에서 IDL이 정의하는 속성과 오퍼레이션을 매핑하는 자바 인터페이스이고, 이들 오퍼레이션들은 TIE 접근법으로 서버에서 클래스에 의해 구현되어야 한다. _tie_Index는 자바 클래스는 인터페이스 구현에서 TIE 접근을 사용함으로써 Index 인터페이스를 서버 측 개발자에게 구현하도록 허용한다. _IndexImplBase는 추상 자바 클래스는 ImplBase 접근을 사용하여 Index 인터페이스를 구현하도록 서버 측 개발자에게 허용한다.



(그림 7) 클라이언트와 서버객체의 생성과정
(Fig. 7) The Generation Procedures of Objects for Client and Server

Client Object의 MainApplet은 클라이언트측의 MainApplet으로 서버객체의 바인딩, 검색 질의 입력 및 오퍼레이션을 요청하고 결과값을 ResultApplet에게 반환, 검색정보에 대한결과를ResultApplet에게 넘겨주고, ResultApplet은 MainApplet로부터 검색 결과를 이용 이미지를 표현하고 Event를 처리한다. 또한 ResultDisplay는 ResultApplet의 Event에 대한 실제 이미지의 표현하는 애플릿이다. MsgDialog는 바인딩 검색결과의 처리 과정에 대한 Message를 처리한다. Server Object는 IndexServer와 Index Implementation으로 나눌 수 있는데 IndexServer는 CORBA의 Server Management에 실제로 등록되는 구현 객체로서 서비스 오퍼레이션인 IndexImplementation 클래스의 인스턴스를 생성시키고, 구현객체를 Ready 상태의 (그림 8)로 만든다.

```
// IndexServer
package Index_Searching;
import java.io.*;
import IE.Iona.OrbixWeb._CORBA;

public class IndexServer {
  public static void main(String args[])
  { Index  Index_var;
   org.omg.CORBA.ORB orb=org.omg.CORBA.ORB.init();
   try {  Index_var  =  new _tie_Index
                        (new IndexImplementation());
         _CORBA.Orbix.impl_is_ready("Index_server");
   try {  Thread.sleep(50000);
       } catch (InterruptedException ex) {}
```

```
    System.out.println("Shutting down server...");
    orb.disconnect(Index_var);
    }
    catch (org.omg.CORBA.SystemException se)
    {
    System.out.println("Exception raised during creation of
        Indexing_Implementation" + settoString());
    System.exit(1);
   }
    System.out.println ("Server is now exiting....");
   }
}
```

(그림 8) 구현객체의 READY 상태
(Fig. 8) The READY Status of Object Implementation

다음으로 IndexImplementation은 서버 측에서 실제로 이미지 검색을 실행하고 결과를 반환하는 오퍼레이션 클래스로 (그림 9)와 같다.

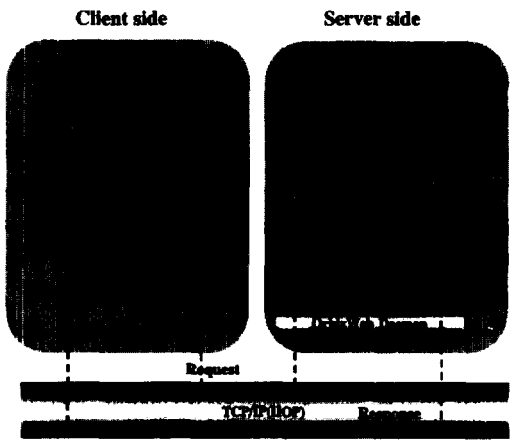
```
// IndexImplementation
class IndexImplementation implements _IndexOperations
{   String[]  result_return_string;
//  path_name  result_returning;

// implementation of the set operation
  public String  image_indexing
  ( String query_one,String
   query_two,org.omg.CORBA.IntHolder result_count)[]
       .....
  }
```

(그림 9) 구현객체의 오퍼레이션
(Fig. 9) The Operation of Object Implementation

4.2 수행 과정

위의 설계 과정을 통하여 서버를 등록한 후, 웹 브라우저와 웹 서버간에는 다음과 같은 과정으로 수행이 이루어진다. 웹 브라우저와 웹 서버간에는 (그림 10)과 같은 과정으로 수행이 이루어진다.

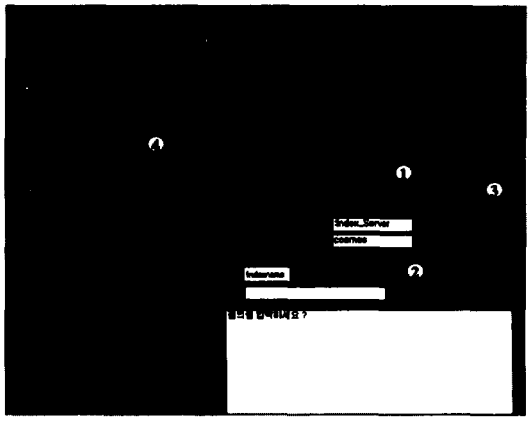
웹 서버가 웹 상에서 CORBA 객체를 사용할 수 있는 HTML을 넘겨주고, 웹 브라우저는 이를 해석한다. 웹 브라우저는 애플릿 태그를 인식하여 자바 인터프리터에게 전송하고, 참조된 최상위 클래스를 실행시킨다.(MainApplet.class) 그리고 자바 인터프리터가 최상위의 코드를 처리함에 따라, 웹 브라우저는 IDL 컴파일에 의해 생성된 코드들과 OrbixWeb 클래스들을 전송 받아 애플릿을 수행하고, IIOP를 통해 웹 서버에 있는 구현객체인 IndexServer.class로부터 검색 서비스를 지원 받는다.



(그림 10) 멀티미디어정보 검색시스템 내의 수행과정
(Fig. 10) The Processing in Multimedia Information Searching System

4.3 구현 결과화면

멀티미디어 정보 검색 서비스의 초기화면은 (그림 11)과 같이 구성되어 있으며, 먼저 구현객체의 바인딩이 이루어진 후에 검색 서비스를 받게된다.



(그림 11) 구현객체(멀티미디어 정보 검색)의 초기화면
(Fig. 11) The Initial Input Screen of Implementation Object for Multimedia Information Searching

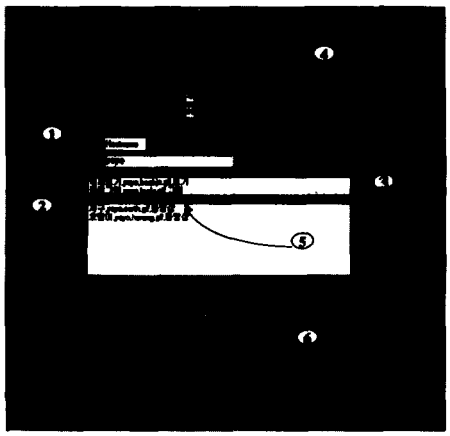
멀티미디어 정보 검색을 위한 구현객체들의 바인딩 절차는 다음과 같다

① 웹 서버 내의 OrbixWeb server Management의 구현객체(멀티미디어 정보검색) 이름을 입력한다. (:Index_Server)

② 구현객체가 등록된 웹서버 내의 Host name을 입력한다. (:cosmos)
③ Host_name 과 Server_name으로 구현객체에 대한 바인딩을 처리한다
④ 바인딩 결과를 메시지 형태로 표현한다.

멀티미디어 정보 검색 결과의 화면에 제공하는 구현 객체들에 대한 바인딩이 이루어진 후 실질적인 검색 서비스가 이루어진다. 검색질의 및 결과에 대한 화면은 (그림 12)과 같이 구성된다. 검색질의 및 검색절차는 다음과 같다

① 질의의 선택옵션을 갖는다.
(Indexname : 인덱스에 대한 옵션을 갖는다(예 : 고려자기, 백제자기,...).)
(Hostname : Hostname에 대한 옵션을 갖는다(예 : yoyo, cosmos...).)
(Explain : Explain에 대한 옵션을 갖는다(예 : avi....).)
② 선택옵션에 대한 실제적인 질의를 입력하는 필드이다.
③ ①~② 입력필드의 질의를 검색 오퍼레이션의 파라미터로 호출 결과를 받아 표현하는 이벤트를 발생시킨다.
④ 검색 결과를 메시지 형태로 표현한다
⑤ 질의 결과에 대한 멀티미디어 정보를 보고 원하는 멀티미디어 정보를 보여주는 이벤트이다.
⑥ 검색 멀티미디어 정보로 그림을 보여준다.



(그림 12) 구현객체(멀티미디어 정보 검색)의 결과화면
(Fig. 12) The Output Screen of Implementation Object for Multimedia Information Searching

5. 결 론

인터넷 환경의 발달로 세계의 많은 컴퓨터들이 연결되어 세계는 하나라는 정보 시대로 도래하게 되었다. 이로 인해 더욱 다양한 서비스에 대한 요구사항을 수용할 수 있는 방법으로 웹 기반의 분산 응용 처리 구조로 CGI를 이용한 방법과 자바기반의 웹 응용 기술을 들 수 가 있다.

CGI를 이용한 방법은 분산응용과의 상호동작을 위해 제안되었지만 서버의 과중한 부하와 다른 서버들로부터 분산 지원 서비스를 받지 못하는 단점을 가지고 있으며, 반면 자바는 객체지향 언어로써 서버의 연산을 분담하여 효율성과 응답성을 높이고, 불필요한 네트워킹의 트래픽을 줄일 뿐만 아니라 가상의 머신에서 실행될 수 있는 기계 중립적인 언어로 플랫폼에 독립적인 장점이 있어서 CGI의 단점을 해결할 수 있다. 그러나 이 기종 분산환경에서 분산 객체들의 재사용 및 상호운용성, 분산 서비스 등을 위한 투명성을 제공하는데 에는 문제점을 가지고 있다[8, 14]. 이러한 문제점을 객체지향 미들웨어의 확장 방법으로 자바와 CORBA의 매핑을 통하여 해결했고, 이를 기존의 멀티미디어 정보 검색 시스템과는 달리 객체지향 기반의 분산환경에 적용하여 시스템을 구현하였다.

따라서 본 논문의 결과는 CORBA구조를 기반으로 한 웹 연동 방법에서 CGI를 사용함으로 발생되는 병목현상과 상태유지의 단절성과 같은 문제점을 GUI 애플릿을 통해 해결하였다. 또한 이 기종의 분산환경에서 기존의 CORBA 객체들과 자바 응용들을 연동시킬 수 있는 자바 ORB기반의 웹과 연동 방법으로 확장하여 웹 자원과 CORBA의 구현 객체를 동시 사용할 수 있도록 하여 웹 브라우저나 일반 클라이언트에게 위치 투명성을 제공하도록 하였다.

향후의 연구내용은 객체지향형 멀티미디어 데이터베이스와의 연동을 위한 DBA(Database Adapter)개발과 다양한 구현객체들의 구현하여 멀티미디어 서비스(텍스트, 이미지, 지리정보등)의 질을 향상시켜야 할 것이다.

참 고 문 헌

[1] Philippe Merle. et al., "CorbaWeb: A generic object navigator" Computer Networks and ISDN Systems 28 (1996) 1269-1281.

[2] Jon Siegel, "CORBA Fundamental and Programming", Object Management Group, 1996.

[3] Robert Orfali · Dan Harkey · Jeri Edwards, "The Essential Distributed Objects Survival Guide", John Wiley & Sons, Inc , 1996

[4] ANDREAS VOGEL, KEITH DUDDY, "Java Programming with CORBA", pp.85~115, 1997.

[5] IONA Technologies PLC, "OrbixWeb 3.0 Beta Programmer's Guide", IONA, 1997.

[6] Object Management Group, " The Common Object Request Broker: Architecture and Specification" Revision 2.0 July 1995.

[7] Orfali Harkey, "Client/Server Programming with JAVA and CORBA" Wiley.

[8] Eric Evans & Daniel Rogers, "USING JAVA APPLETS AND CORBA for Multi-User Distributed Applications", IEEE Internet Computing, pp.43~55, MAY 1997.

[9] "JAVA Programming" Sun microsystems.

[10] http://www.iona.com/Products/Orbix/OrbixWeb/index.html

[11] http://java.sun.com/Products/Jdk/1.1/

[12] http://www.omg.org/corba/corbiiop.htm

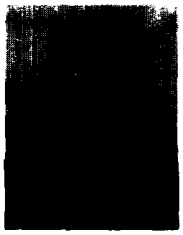
[13] http://www.acl.lanl.gov/CORBA/

[14] 마경훈, 이상호, "웹과 CORBA의 연동 기술에 관한 연구" 정보과학회 데이터베이스연구회지, 13권 2호, 1997.

[15] 박재현, 한상만, " ObjectWeb 기반의 시스템 통합 기술".

[16] 이병대 외, "CORBA를 이용한 데이터베이스 브로커의 개발" 정보과학회 봄 학술발표논문집 Vol.24, No.1, 1997.

[17] H. Onozawa,"분산 오브젝트 지향기술 CORBA" 홍릉과학 출판사, 1997.



이 원 중

1990년 전북대학교 자연대학 전산·통계학과(이학사)

1998년 원광대학교 교육대학원 전자계산교육전공(교육학석사)

1998년 원광대학교 대학원 컴퓨터공학과 박사과정

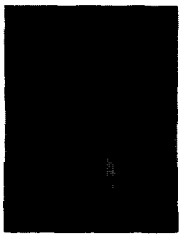
관심분야 : 멀티미디어 데이터베이스, 멀티미디어 정보검색

안 길 수

1997년 원광대학교 공과대학 컴퓨터공학과(공학사)

1997년 원광대학교 대학원 컴퓨터공학과 석사과정

관심분야 : 멀티미디어 데이터베이스, 분산 데이터베이스, 멀티미디어 정보검색



주 수 종

1986년 원광대학교 전자계산공학과(공학사)

1988년 중앙대학교 대학원 컴퓨터공학과(공학석사)

1992년 중앙대학교 대학원 컴퓨터공학과(공학박사)

1993년~1994년 미국 Univ. of Massachusetts at Amherst, 전기 및 컴퓨터공학과, Post-Doc.

1990년~현재 원광대학교 공과대학 컴퓨터공학과 부교수

관심분야 : 멀티미디어 데이터베이스, 분산 실시간 컴퓨팅, 분산객체모델, 시스템 최적화