분산객체기반 Web서버 시스템의 설계 및 구현

김광수* · 박규석 **

1. 서 론

사용자들은 웹에서 쉽고 편리하게 서비스를 이용할 수 있게 되었지만 참조 무결성, 이동 투명성 그리고 유지 · 보수의 어려움과 같은 웹의 문제점을 인식하게 되었다[1,2,3]. 또한 웹서비스의 수요가 급속히 증대해가면서 서비스와 응용 프로그램들도 점점 커져가고 있다. 하지만 향상된 서비스를 제공하기 위해서 기존의 시스템 및 정보 등을 웹에 통합한 응용 프로그램을 개발하는데 어려움을 지니고 있다. 또한 웹 시스템은 클라이언트/서버 모델에 기반을 두고 있으며 통신 때 사용되는 프로토콜인 HTTP(Hyper Text Transfer Protocol)는 상태 정보를 유지하지 않는(stateless)특성을 지니고 있다. 이는 서버의 부하를 줄여주는 장점은 가지고 있으나 올바른 자원의 참조가 이루어지지 않을 위험이 있다.

본 논문에서는 이러한 문제점을 개선하기 위하여 분산 객체 기술을 이용한 웹 지원 시스템인 CWS(Corba Web Server)를 설계 및 구현하였다. 분산 객체 기술을 이용하기 위해서 CORBA(Common Object Request Broker Architecture) 및 Java를 이용한다.

본 시스템은 객체 지향 웹서버를 이용하여 자원과 응용 프로그램의 상태성을 유지하며, 객체 단위로 자원의 관리가 이루어진다. 따라서 대규모화되는 웹 자원의 유지 · 보수도 객체 지향 기반으로 효율적으로 이루어지며, 이러한 객체 단위의 관리는 더 높은 수준의 유연성과 확장성을 제공할 수 있다.

*경남대학교 정보통신 공학부 컴퓨터공학전공 석사 졸
**경남대학교 정보통신 공학부 교수

2. 관련 연구

2.1 웹의 특성

웹은 인터넷에 분산 되어있는 다양하고 광범위한 정보를 GUI환경에서 편리하고 쉽게 접근 가능하도록 한 분산 하이퍼미디어 정보 시스템이다. 즉, 웹은 하이퍼텍스트 기법으로 쉽게 웹 자원의 접근과 활용이 가능하지만 자원의 작은 변화에도 웹 환경에 큰 영향을 미칠 수 있다[1,2,3].

2.1.1 참조 무결성

저장되어 있는 정보와 참조하고있는 정보가 정확성을 가져야 한다. 만약 허용되지 않는 값이나 부정확한 데이터가 참조된다면 참조 무결성이 유지된다고 할 수 없다. 만약 시스템이 자원 참조의 정보와 참조의 수를 인식할 수 있다면 참조 무결성을 유지할 수 있지만 기존의 웹에서 사용되는 HTTP 1.0 프로토콜은 상태성을 가지지 않으며 서버도 클라이언트에서 자원의 참조가 이루어질 때 참조 정보와 참조의 수를 인식하지 않기 때문

에 참조 무결성을 유지하기가 어렵다. 즉, 임의의 자원에 대한 클라이언트의 요구가 처음 요청한 것인지 다른 클라이언트에서 요청하여 사용중인 자원인지 알지 못한다. 만약 참조중인 자원에 대한 자원의 변경이나 수정이 가해지면 올바른 자원의 참조가 이루어질 수 없다. 반면에 항상 올바른 참조가 이루어지도록 서버가 가지고 있는 모든 자원들이 변경이나 수정 없이 계속 유지되기를 바라는 것도 비현실적이다. 따라서 유지·보수를 위해서 자원의 변경은 필수 불가피하다고 할 수 있으며, 이것은 부작용(side-effect)의 잠재성을 내재하게 된다.

2.1.2 이동 투명성

이동투명성이란 사용자나 응용 프로그램의 수행연산에 어떠한 영향도 미치지 않고 자원의 이동이 가능한 것을 의미한다. 하지만 기존의 웹에서 자원의 이동은 URLs의 변경을 야기한다. 즉, 자원의 이동이 발생했을 경우에 이 자원에 링크된 자원들은 이동된 URLs로 변경작업을 해야한다. 이러한 작업은 관리자의 실수나 정보 제공자의 실수로 하이퍼텍스트 링크가 끊기는 잠재성을 가진다. 따라서 자원의 이동이 발생했어도 URLs의 변경 없이 자원의 참조가 가능한 방법이 요구된다.

2.2 인터넷 네트워크 기술

2.2.1 TCP/IP

TCP/IP(Transmission Control Protocol/Internet Protocol)는 OSI(Open Systems Interconnection) 모델을 직접 따르지 않으며, 그 계층은 Application Layer, Transport Layer, Internet Layer, Physical Layer로 구성된다.

1) Application Layer

이 계층은 네트워크를 실제로 사용하는 응용프로그램(FTP, Telnet, SMTP등)으로 이루어진다. OSI 모델에서 보면 애플리케이션 계층과 프리젠테이션 계층에 해당한다.

2) Transport Layer

이 계층의 역할은 도착하고자 하는 시스템까지 데이터를 전송하는 것이며, OSI 모델에서 보면 Session 계층과 Transport 계층에 해당하고, TCP/IP소켓부분이 OSI 모델의 Session 계층에 해당한다. TCP/IP에서는 시스템의 주소(address)와 포트(port)를 가지고 각 프로세스를 연결하여 통신한다. OSI 모델의 Transport에 해당하는 부분은 TCP/IP프로토콜의 TCP프로토콜에 해당한다.

3) Internet Layer

이 계층의 역할은 데이타그램을 정의하고 데이타그램을 라우팅 시키는 일을 담당한다. 데이타그램이 가지고 있는 자료는 보낸 주소(source address), 받을 주소(destination address), 보내는 데이터, 그 외 몇 가지 제어 필드(control field)를 가지고 있다. OSI 모델에서는 네트워크 계층과 데이터 링크 계층에 해당한다.

4) Physical Layer

물리적 계층에 대한 TCP/IP 프로토콜에서 따로 정의한 내용은 없고, IEEE에서 정해놓은 하드웨어 표준에 따른다.

컴퓨터 시스템에서의 서로 상이한 언어와 기호체계의 사용으로 발생하는 문제점을 극복하기 위해 고안된 것이 컴퓨터 통신의 표준 프로토콜(protocol)이다. 오늘날의 인터넷을 가능하게 한 것도 바로 TCP/IP이다. 가장 중요한 기술적 특징은 인터넷용 프로토콜인 TCP/IP 프로토콜이 지니고 있는 특성에서 나온다고 할 수 있다.

2.2.2 WWW(World Wide Web)

HTTP(Hyper Text Transfer Protocol)는 분산

환경 및 정보서비스 제공을 목적으로 설계된 응용계층의 프로토콜로서 웹(World Wide Web)에서의 하이퍼텍스트 문서의 전송을 위해 쓰이는 프로토콜이다. 또한 요구 명령어의 추가를 통해 네임서버나 분산 객체 관리 시스템 등과 같은 여러 가지 일에도 사용할 수 있는 객체 지향 프로토콜이며, MIME(Multipurpose Internet Mail Extension)에 의해 정의될 수 있는 모든 문서 형식을 전송할 수 있다[4].

2.3 WWW(World Wide Web) 구성 형식

2.3.1 Hypermedia

하이퍼미디어(hypermedia)란 하이퍼텍스트(hypertext)의 커다란 집합체라고 말 할 수 있다. 하이퍼텍스트는 일반적인 텍스트와 별 다른 차이가 없지만 하이퍼텍스트 링크 즉, 하이퍼링크(hyperlink)라는 다른 문서로의 연결고리를 가진다는 큰 특징을 가지고 있다. 이는 다시 말해, 다른 문서 또는 텍스트로의 하이퍼링크를 가지는 하이퍼텍스트 문서이면서, 연결되어 있는 문서의 형태가 단순한 텍스트뿐만 아니라 여러 형태 즉, 음성 또는 영상(정지영상, 동영상) 등의 멀티미디어 데이터라는 것이다[4].

2.3.2 URL(Uniform Resource Locator)

URL은 WWW 시스템에서 인지할 수 있는 주소 지정 문법이라 할 수 있는데, WWW 시스템은 이 URL을 이용하여 하이퍼미디어 링크를 표현한다. 따라서 인터넷상의 모든 화일과 망 서비스가 URL을 통해 연결 및 사용 가능하게 된다.

2.3.3 HTML(HyperText Markup Language)

HTML이란 하이퍼미디어 문서를 생성하고 인식하기 위해 Web이 사용하는 표준 언어이다. 이것은 SGML(Standard Generalized Markup Language)이라고 하는 문서 형성 언어의 부분집합이라고 할 수 있으며, 현재 IETF에서 제시한 HTML 2.0이 폭넓게 사용되고 있다.

현재 널리 사용되고 있는 Mosaic이나 Netscape 같은 브라우저는 HTML+(3.0)의 많은 기능을 제공하기 못하고 있다. 반면에 ARENA 브라우저는 HTML+(3.0)의 거의 모든 기능이 지원되고 있다[6].

2.3.4 HTTP(Hyper Text Transfer Protocol)

HTTP란 WWW 클라이언트와 서버가 서로 통신하기 위한 일종의 응용 프로토콜이다. WWW의 서버와 클라이언트는 HTTP를 이용하여 서로 하이퍼미디어 문서를 주고받을 수 있으며, 서버와 클라이언트간에 전송되는 데이터 특성에 상관없이 HTTP를 이용하여 데이터 표현 상의 절충이 가능하다.

현재, HTTP 버전 1.0으로 아직, 드래프트 단계이며 또 한편으로는 HTTP-NG(Http Next Generation)가 개발되고 있다.

2.4 Web

분산 객체 기술을 응용한 연구는 크게 세 가지로 분류된다. 첫째, 게이트웨이를 이용한 방법으로서 대표적인 예로 ANSA(Advanced Networked Systems Architecture)에서 연구한 CORBA의 자원을 활용하는 방법이며 둘째, 웹에서 분산 객체 기술을 적용하기 위해서 웹의 하부 시스템을 설계하는 방법으로서 Ajuna의 W3Object가 그 대표적인 예이다[1-3]. 마지막으로 CGI(Common Gateway Interface)를 이용해서 분산 객체 시스템을 연동하는 Corba Web방법을 들 수 있다.

게이트웨이를 구성하는 방법으로서는 기존의 CGI를 이용하거나 HTTP 프로토콜을 IIOP(Internet Inter-Orb Protocol)로 변경해주는 방법이 있

다. 그리고 Java와 CORBA를 매핑하는 방법이 있으며, HTTP-to-IIOP의 경우는 ANSA와 오라클에서 사용하고 있다[7]. 이 방법은 CORBA가 HTTP를 처리하는 역할을 갖는 것을 의미한다.

시스템의 변형을 통해 연동하는 방법은 CORBA IDL(Interface Definition Language)컴파일러를 확장하여 해당 IDL 파일을 WWW에 적합한 코드로 생성해 주는 방법이다. 이 방법은 CORBA IDL를 그대로 사용할 수 있다는 장점이 있으나 CORBA 인터페이스마다 개별적인 FORM 문을 생성하기 때문에 대규모 시스템 개발에는 부적합하다.

그 외에 HTTP와 CORBA를 공존시키는 방법으로는 웹 클라이언트는 CORBA에 서비스를 요청하기 위한 ORB기능을 가지고 실행 시에 서버 측에 자바로 작성된 CORBA 응용 프로그램을 다운 로드 받은 후 이를 수행시키는 방법이 있다. 이 방법은 현재 네스케이프와 VisiBroker, 선사의 NEO등에서 사용하고 있다[7].

지금까지의 CORBA와 WWW에 관한 연구들은 기존의 웹 기능을 확장하는데 치중하여 확장에 따르는 문제점을 해결하지 못하고 있다.

3. 시스템 설계

본 논문에서 제안하는 CWS(Corba Web Server)를 구현하기 위해 순수한 Java로 작성된 웹 서버 데몬 및 서버의 확장된 모듈 그리고 CORBA의 기능을 제공하는 Java ORB를 사용한다. Java로 작성된 객체 지향 웹 서버는 기존의 많은 웹 서버 데몬과 같은 역할을 하고 또한, 확장된 모듈은 웹 문제점을 개선하며 CORBA 자원과 서비스의 활용을 가능하게 한다.

Java ORB는 Java로 작성한 응용 프로그램에서 CORBA 자원으로 직접 접근을 가능하게 하여 유연성과 확장성을 향상시킨다. CWS 시스템은 그림 1에서처럼 두 부분으로 구성된다. 첫째는 객체 지향 웹 서버이다. 객체 지향 웹 서버는 객체 단위로 자원을 관리하며 객체 관리자가 그 기능을 담당하게 된다. 둘째는 웹의 최하위 구조로서 CORBA의 기능을 완벽하게 제공하는 Java ORB이다. Java ORB는 CORBA의 자원 및 이름 서비스를 제공한다.

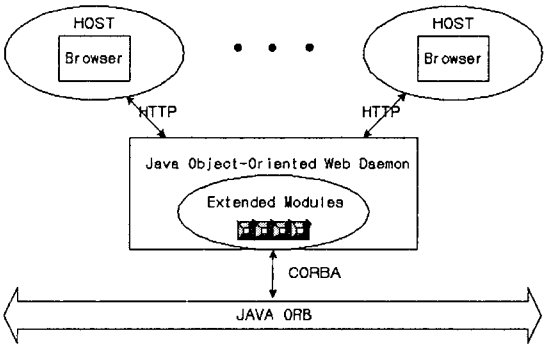


그림 1. 시스템 구조

3.1 CWS 구성

시스템의 전체 구성도는 다수의 클라이언트와 클라이언트의 요구에 응답할 Daemon으로 구성된다.

CWS 시스템의 구성은 그림 2와 같으며 Application인 Frm Web Serber, 각 모듈을 관리하는 Object Manager, 그리고 각 모듈을 활성화시키는

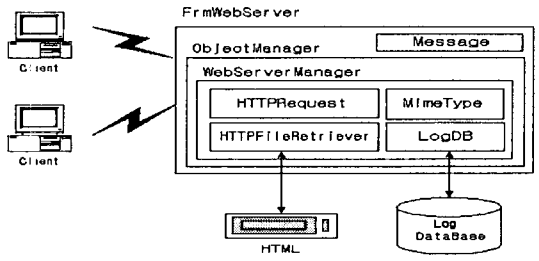


그림 2. CWS 시스템 구성

Web Server Manager, 클라이언트 요구에 서비스할 HTTP Request, HTTP File Retriever, MIME Type을 관리하게 되는 MIME Type Manager, 클라이언트 정보를 저장할 LogB Manager로 구성되어 클라이언트에게 서비스를 제공한다. 그리고 구현객체에서 서비스하는 내용을 보여주기 위한 Message 모듈로 구성되어 있다.

CWS시스템의 전체 시나리오는 그림 3과 같다. Web Server Manager가 클라이언트의 요구를 받게 되면 응답할 모듈인 HTTP Request를 활성화시키기 위해 Process Request 이벤트를 생성한다. 그리고 로컬에 저장된 HTML 문서와 MIME Type의 내용을 클라이언트에게 전송하기 위한 HTTP File Retriever을 활성화할 Process Request를 생성한다. HTTP File Retriever이 클라이언트에게 HTML 문서를 전송할 때 MIME Type Manager로 등록된 MIME Type을 전송 받게되고 클라이언트에 대한 정보를 Log DB Manager에게 전송하여 결과를 Data Base에 저장한다.

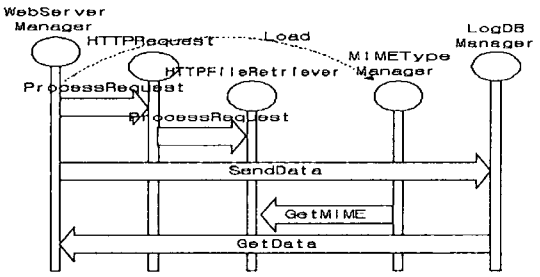


그림 3. CWS 시스템의 시나리오

3.2 모듈별 기능

CWS의 각 모듈별 기능은 다음과 같다.

3.2.1 Object Manager

구현객체와 연결하게 되는 모듈이다. 그리고, 향후 객체의 추가나 기타 다른 객체의 사용을 용이하도록 하기 위한 역할을 수행하게 된다.

3.2.2 Web Server Manager

Web Server Manager는 클라이언트 요구를 감시하는 이종의 Daemon역활을 하게되는 모듈이다. 이 모듈의 역할은 클라이언트의 요구가 들어왔을 경우 HTTP Request 모듈을 활성화시키고 MIME Type로부터 설정된 MIME Type의 내용을 전송 받게 된다. HTTP Request가 활성화된 후 HTTP File Retriever 모듈은 로컬에 있는 HTML 문서를 클라이언트에게 서비스한다. 또한, 접속된 클라이언트에 대한 정보를 Log DB Manager에게 그 정보를 전송하게 된다. 또한 현재 상태를 확인하기 위해 클라이언트가 요구한 파일의 이름을 보여주기 위해 Message 모듈에게 그 내용을 전송하여 화면상에 보여주는 기능도 하게 된다.

● 기본 알고리즘을 이용한 Web Server

① 클라이언트가 요구한 Socket을 Open 하여 클라이언트와 연결한다.

② 클라이언트가 요구한 파일, 디렉토리 그리고 Method를 확인한다.

③ 로컬에 있는 HTML 문서를 파싱한다.

④ Open된 Socket으로 파싱한 HTML문서를 전송한다.

⑤ Socket을 Close한다.

Web Server가 동작하게 되는 알고리즘을 도식화하면 그림 4와 같다.

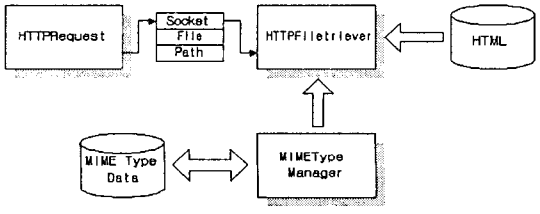


그림 4. HTTP File Retriever 데이터 흐름도

3.2.3 HTTP Request

HTTP Request는 Web Server Manager로부터 클라이언트의 요구에 활성화되는 모듈로서, 모듈을 초기화한 후 클라이언트가 요구한 디렉토리의 위치, 파일의 이름, Host 이름, Method 방식 등을 지정하거나 확인하게 된다. 그리고 구성된 FORM Method가 POST또는 GET 방식에 따라서 클라이언트에게 서비스 할 수 있도록 확인하는 모듈이다. 클라이언트에게 응답할 Server Socket을 Open하고 이상유무를 체크한 후 전송하게 되는 파일의 이름과 내용을 확인하고 그 내용을 HTTP File Retriever 모듈에게 전송한다.

Method는 POST와 GET 방식의 가장 큰 차이점은 "?" 의 존재 여부로 구분할 수 있다. GET 방식은 "?" 뒤에 전송할 코드가 함께 넘겨지게 된다. 이에 대해 POST 방식은 일반 HTML 문서와 같은 형태로 넘겨지게 되는 것이다. 이에 대한 Java 소스는 그림 5와 같다.

```
try {
   if (getMethod().equals("GET")) {
        formVariables = getRequestedObject();
        int i = formVariables.indexOf("?");
        if (i == -1) throw new Exception("no form variables");
        formVariables = formVariables.substring(i + 1,
        formVariables.length());
        i = formVariables.indexOf(" ");
        if (i > 0) formVariables = formVariables.substring(0, i);

   } else if (getMethod().equals("POST")) {
        int i = requestStr.indexOf("WnWn");
        formVariables = requestStr.substring(i+4,
                                requestStr.length()).trim();
   } else formVariables = new String("Unknown Method");

} catch (Exception e) {
   formVariables = "";
}
```

그림 5. Web Server 기본 동작 알고리즘

3.2.4 HTTP File Retriever

HTTP File Retriever 모듈은 HTTP Request 모듈로부터 전송 받은 파일을 다시 확인하고 로컬에 저장되어 있는 HTML문서를 파싱해서 그 내용을 클라이언트에게 전송하게 된다. 그리고 전송 시 MIME Type 모듈로부터 이미 설정되어 있는 MIME Type에 따라 서비스하는 모듈이다.

그림 6은 HTTP File Triever 모듈을 중심으로 데이터 흐름을 나타낸 것이다. HTTP Request 모듈은 클라이언트와 연결한 Socket 및 전송 요구를 받은 파일 이름, 파일이 있는 경로를 나타내는 Path 등 3개의 데이터를 HTTP File Triever 모듈에게 전송한다. 이러한 작업이 정상적으로 이루어지면 HTTP File Triever 모듈은 로컬에 저장된 HTML 문서를 파싱해서 클라이언트와 연결된 Socket으로 전송할 준비를 하게 된다. 이때 MIME Type 모듈은 로컬에 저장된 MIME Ttype 데이터를 읽어들여 HTTP File Triever 모듈에게 전송하여 클라이언트가 요구하는 대로 처리한다.

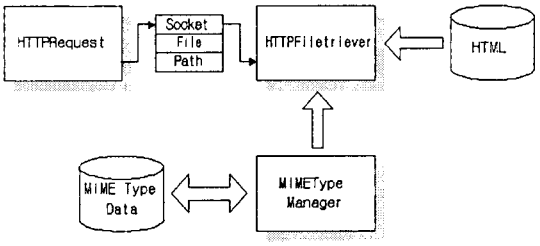


그림 6. HTTP File Retriever 데이터 흐름도

3.2.5 MIME Type

MIME Type의 내용을 Network Working Group에서는 MIME Type을 RFC 822에서 정의하고 있다. 이러한 구성요소들은 확장 가능한 메카니즘과 다양한 표현형식으로 전송할 수 있도록 하기 위해서이다. 그러나 RFC 1521 은 메일에 대한 설명을 하고 있으나 HTTP RFC 1521에 설명되어 있는 것과 약간 다른 기능을 갖고 있기도 하다. 이러한 차이점들은 이진모드 전송 연결 상

에 있어 성능을 최적화하고, 새로운 미디어의 사용을 훨씬 더 자유롭게 하며, 날짜 비교를 더욱 쉽게 하여 Web 서버와 클라이언트의 활용성을 알리기 위한 목적으로 선택된 것들이다.

MIME Type 모듈은 로컬에 저장된 MIME Type 파일을 관리하게 된다. Web Server Manager 모듈에서 확인하는 것은 로컬에 저장된 파일이 정상적인지를 확인하는 것이며, 실제적으로 MIME 데이터를 불러들이는 것은 HTTP File Retriever 모듈에서 처리하게 된다. HTTP File Retriever 모듈이 로컬에 저장된 HTML 문서에 MIME Type이 있는지 확인하고 있으면 이에 대한 처리를 MIME Type 모듈에게 넘겨주어 처리하도록 처리한다. 또한 HTTP File Retriever 모듈이 설정된 MIME 요구 시 get MIME 이벤트로 자료를 전송하여 클라이언트 요구에 대한 서비스를 담당한다. 그림 7에서 MIME Type 설정부분을 보여주고 있다.

```
ObjectOutputStream out = new ObjectOutputStream(
                        FileOutputStream("mimetypes.dat"));

java.util.Hashtable dict = new java.util.Hashtable();

for (int i=0; i<mimeTable.length; i++) {
  System.out.println("Adding: " + mimeTable[i][0] + ", " +
                     mimeTable[i][1]);
  dict.put(mimeTable[i][0], mimeTable[i][1]);
}

out.writeObject(dict);
out.close();
```

그림 7. MIME 설정

표 1에서는 CWS에서 기본적으로 설정한 MIME Type의 내용을 나타내고 있다.

3.2.6 Log DB Manager

웹을 통해 많은 수의 클라이언트 접속이 이루어지며, 이에 따라 서버가 받게되는 부하라든가, 빠른 처리시간의 요구, 다량의 데이터 전송 등을 위한 DBMS에의 접근을 위해 ODBC에 연결하여 질의를 처리하였으며, 이때 구현객체의 ODBC 연결에는 JDBC(Java DataBase Connectivity)를 사용하였다.

표 1. MIME Type 설정

Extension	MIME Type
ai	application/postscript
aif	audio/aiff
aifc	audio/aiff
avi	video/avi
css	text/css
doc	application/msword
enc	video/mpeg
eps	application/postscript
exe	application/x-msdownload
fif	application/fractals
gif	image/gif
gz	application/x-gzip
hqx	application/mac-binhex40
htm	text/html
. . .	. . .
zip	application/x-zip-compressed

그림 8은 Log DB Manager의 클라이어트 정보 저장에 대한 DB 구성도를 나타낸 것이다.

Log DB Manger 모듈은 Web Server Manager 모듈로부터 전송된 클라이언트 정보를 DB에 저장하고 관리하는 모듈이다. 저장되는 내용은 클라

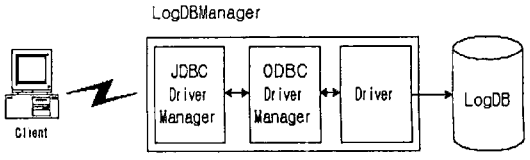


그림 8. DataBase 구성도

이언트의 IP 주소, IP 주소에 대한 Host이름, 클라이언트가 접속한 날짜와 시간, 그리고 HTML 전송시 사용된 Method와 전송하는 HTML 파일이름을 저장하고 관리한다. 표 2는 LogDB의 각 필드를 나타낸 것이다.

표 2. Log 데이타베이스

Field Name	Field Type	설 명
IP	String(15)	클라이언트 IP주소
HOST	String(20)	클라이언트 HOST 이름
Date	Date	접속한 날짜
Time	Date	접속한 시간
Method	String(5)	클라이언트 요구 방식
FileName	String(10)	클라이언트가 요구한 파일

3.3 알고리즘

그림 9의 알고리즘에서 서버가 클라이언트의 요구를 대기하다가 접속이 이루어지면 사용자의 요구에 맞는 서비스를 행하게 됨을 알 수 있다.

또한 웹서버에서 기본적으로 지원해야 될 MIME을 구성하게 되고 사용자가 GET 혹은 POST중 어느 방식을 사용하였는지 확인한 후 그에 맞는 서비스를 수행하도록 되어 있다.

```
public void openServerSocket()
{
    Socket Open
}

public void run()
{
    // IMPLEMENT: Operation
    Socket requestSock;
    while (!isTerminated()) {
        상용자 서버스요규 대기 및 서비스 시작
    }
    try { closeSocket(serverSock); }
    catch (Exception e) { }
}
void processRequest(Socket requestSocket) throws IOException
{
    try {
        if (getMethod().equals("GET")) {
            GET방식으로 넘어온 데아타 Parsing
        }
        else if (getMethod().equals("POST")) {
            POST방식으로 넘어온 데아타 Parsing
        }
        else formVariables = new String("Unknown Method");
    } catch (Exception e) {
    formVariables = "";
}
public static void load() throws IOException
{
   try {
      ObjectOutputStream out = new ObjectOutputStream(
    FileOutputStream("mimetypes.dat"));
       Default MIME TYPE구성
    } catch(Exception e) {}
    out.writeObject(dict);
    out.close();
}

public void connect(java.lang.String url)
{
    ODBC연결
}

public void insert(Cws.UserInfoDef userInfo)
{
    사용자정보저장
}
```

그림 9. 알고리즘

4. 구현 및 평가

4.1 구현

본 논문에서 제안한 CWS 시스템의 설치 및 운용에 대한 구현 환경은 표 3에서 나타낸 것과 같다.

표 3. 구현환경

분류	CWS 시스템	클라이언트
사용OS	Windows NT 4.0	Windows 95
프로세서	펜티엄 200	펜티엄 166
RAM크기	64M	40M
개발도구/브라우저	Visual Cafe 3.0 Inprise의 VisiBroker	Netscape
사용 DB	Access97	

제안 시스템은 환경설정에 등록된 디렉토리에 HTML 문서를 작성해서 CWS 시스템을 동작시킨다.

그림 10은 클라이언트가 접속한 내용에 대한 정보를 보고자 할 때의 처리 화면이며, 클라이언트 주소, Host 이름, 접속 날짜, 시간, Method, 파일 이름 등을 볼 수 있다.

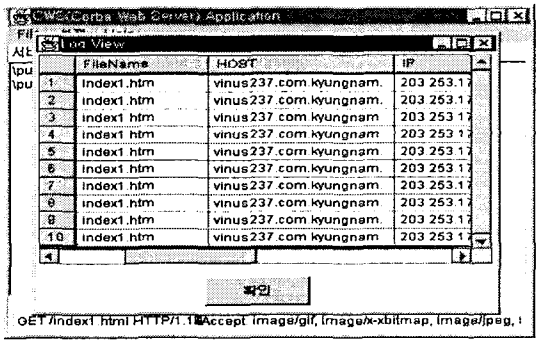


그림 10. Log 화면

그림 11은 CWS의 환경 설정 화면이다. 관리자가 원하는 기본문서, 디렉토리 설정, 그리고 클라이언트에게 OPEN할 port를 설정할 수 있으며, 관리자가 필요에 따라서 환경을 바꾸어 사용할 수 있다.

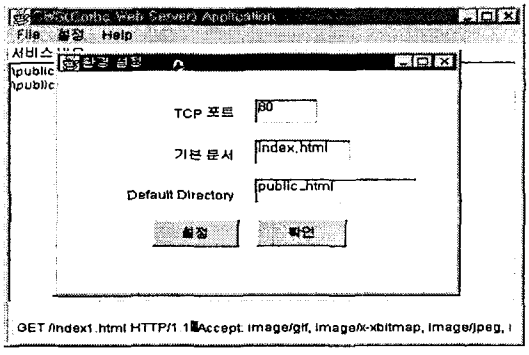


그림 11. 환경 설정 화면

그림 12는 MIME Type을 설정하는 화면으로써 화면에 보이는 내용들은 시스템에서 추가된 MIME Type이다. 관리자가 필요에 따라서 그 내용을 추가하거나 수정할 수 있다.

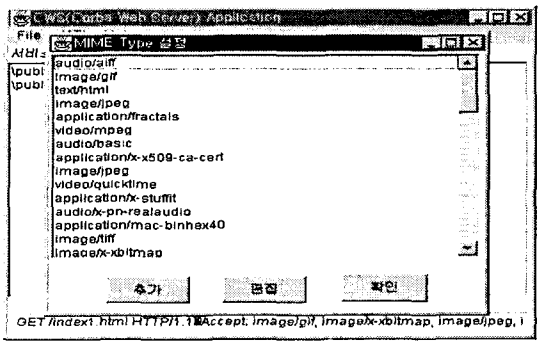


그림 12. MIME Type 화면

4.2 평가 및 분석

본 논문에서 설계한 CWS 시스템과 Sun 사에서 발표한 Java Web Server와 수행 성능을 비교하였으며, 표 4는 각각에 대한 수행 환경을 나타내고 있다.

표 4. 수행 환경

분 류	CWS 시스템	Java Web Server	클라이언트
사용OS	Windows NT 4.0	Windows NT 4.0	Windows NT 4.0
프로세서	펜티엄 200	펜티엄 200	펜티엄 200
RAM	64M	64M	64M
DB			Access97

수행을 위한 구성 요소는 다음과 같다.

① 클라이언트 : 네트웍을 통해 연결된 다수의 클라이언트들은 임의의 시간에 Server에게 데이터를 요구한다.

② 서버 : 클라이언트 요구에 대해 로컬에 저장된 HTML문서를 서버에게 전송한다.

다수의 클라이언트환경을 만들어서 테스트하

기는 어려운 특성이 있기 때문에 본 성능평가를 위한 수행 환경에서는 이러한 상황을 가정하고, 시스템이 설정할 수 있는 클라이언트 수를 증가하면서 서버에서 데이터를 전송하는 시간을 데이터베이스에 저장하였다. 그림 13은 클라이언트 설정화면을 보여주고 있다.

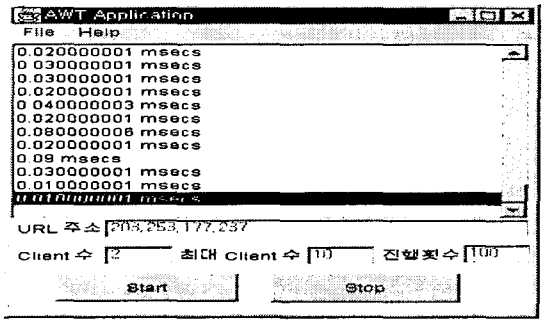


그림 13. 클라이언트 설정화면

그림 14와 그림 15는 클라이언트 수에 대한 응답시간을 나타내고 있다. 클라이언트 수가 1인 경우에는 Java Web Server의 응답지연시간이 짧은 것을 알 수가 있다. 클라이언트 수가 늘어날수록 응답지연시간이 그 만큼 길어지고 있지만, CWS 시스템의 경우 사용 클라이언트의 수가 늘어남에도 불구하고 응답시간이 짧아지는 것을 알 수 있다.

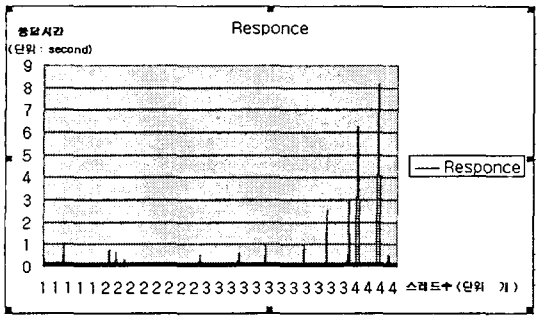


그림 14. Java Web Server 응답시간

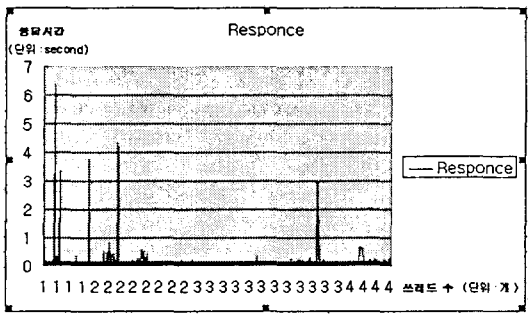


그림 15. CWS 시스템 응답시간

다수의 사용자 접속에 대한 응답시간은 시스템 자원을 얼마나 많이 사용하고 있는 가를 나타낸다고 할 수 있으며, 따라서 CWS 시스템은 적은 자원을 사용하여 클라이언트에 대한 서비스를 하고 있음을 알 수 있다.

5. 결 론

기존의 웹은 참조 무결성, 이동 투명성 그리고 유지·보수의 어려움과 같은 문제점들을 지니고 있으며, 응용 프로그램 개발을 위한 유연성과 확장성을 제공하지 못한다.

본 논문에서는 웹의 상기 문제점들을 보완하고 유연성과 확장성을 제공할 수 있는 CWS 시스템을 설계하고 구현하였다. 본 시스템을 이용할 경우, 객체 지향을 기반으로 한 자원의 관리가 가능하게 되며, 웹 서버의 확장된 내부 모듈로 기존의 웹 문제점들을 개선하였다. 또한 인터페이스 환경을 향상시키고 응용 프로그램에 대한 광범위한 자원 참조가 가능하도록 하는 강력한 분산 객체 운영 처리가 가능하다.

향후 기존의 웹에 Proxy 서버 기능을 추가하여 좀더 빠른 결과와 보안문제 해결을 위한 계속적인 연구가 필요하다.

참 고 문 헌

[1] D. B. Ingham, M. C. Little. S. J. Caughey and S. K. Shrivastava, W3Objects : Bring Object-

Oriented Technology to the Web, World Wide Web Joural, Issue 1, pp. 89-105 : Proceeding of the Fourth International World Wide Web Conference, Boston, Mass., USA, 1995.

[2] D. B. Ingham, S. J. Caughey and and M. C. Little, Supporting Highly Manag7eable Web Services, Proceeding of the Sixthe International World Wide Web Conference, Santa Clara, USA 7-11 April 1977

[3] D. B. Ingham, S. J. Caughey, and M. C. Little, Fixing the Broken-Link Problem : W3Objects Approach, Computing Network & IDSN System, Vol. 28 No. 7-11, pp. 1255-1268 : Proceeding of the Fifth International World Wide Web Conference, Paris, France, 6-10 May 1996.

[4] Roy Fielding, "Hypertext Transfer Protocol HT TP/1.0", RFC 1945, IETE HTTP WG, May 1996.

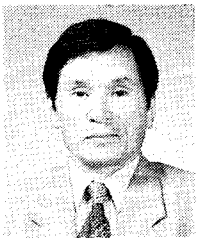
[5] Fielding, R., "Relative Uniform Resource Locators", RFC 1808, UC Irine, June 1995.

[6] 건국대학교 전자계산학과, 한선영, http://sharon.comeng.chungnam.ac.kr/~dolphin/ws2/content/TT1/

[7] Edwards, N., Object Wrapping for WWW-The Key to Integrated Services, Document Identifier APM. 1464, Architecture Project Management Ltd., Cambridge, 1995.

김 광 수

• 경남대학교 컴퓨터 공학과 졸업(공학사)
• 경남대학교 대학원 컴퓨터 공학과 졸업(공학석사)
• Siri infonet/개발팀장



박 규 석

• 중앙대학교 전자계산학과 졸(이학 석, 박사)
• 한국 정보 과학회 영남지부장/이사
• 한국 멀티미디어 학회/ 부회장
• 現 경남대학교 정보통신 공학부/교수
• 現 경남대학교 정보통신 연구소/소장