XML-RPC 기반의 분산환경 문서관리 시스템 모델

고혁준* · 김정희** · 곽호영***

DEDMS : Distributed Environment Document Management System Model based on the XML-RPC

Hyuck-Jun Ko* · Jeong-Hee Kim** · Ho-Young Kwak***

요 약

웹 서버에서 제공하는 문서 자원들은 URL/URI 형식으로 표현되고 있지만, 동적인 서버 환경의 변화로 인해 반드시 해당 자원이 서버에 존재하고 있는 것을 보장할 수 없다. 따라서 본 논문에서는 자원에 대한 신뢰성을 보장하고, 동적인 서버 자원 관리 및 클라이언트의 요청을 처리하기위해 XML-RPC를 이용한 통합 문서 관리 시스템을 제안하고 모델링 한다. 제안한 시스템은 동적인 서버 자원을 관리하는 미들웨어 시스템과 클라이언트가 서버에 저장시킨 문서에 대한 갱신 정보를 서버에서 미들웨어 시스템으로 통보하는 서버 시스템으로 구성된다. 모델링 결과, 분산된 서버에 있는 동적으로 변하는 문서들을 효과적으로 저장 관리할 수 있었으며, 현재 운영되고 있는 웹 서버에 적용시킬 수 있어 새로운 웹 서버 구축비용을 절감할 수 있고, XML-RPC 프로토콜을 사용하기 때문에 플랫폼 독립적이고 또한 데이터 관리가 효율적임을 알 수 있었다.

ABSTRACT

Even the document resources offered from web server can be represented in the form of URL/URI, it can not necessarily be guaranteed that corresponding resources exist due to a dynamic change of server environment In this paper, integrated document administration system is therefore proposed and modeled using the XML-RPC technology which guarantees the reliance of resources, and handles a dynamic server resource management and request of clients. The proposed system is composed of middleware and server systems. The former system manages dynamic server resources, and the latter reports the updated information of documentations stored in server by client from the server to middleware system. As a result, effective storing management of dynamic resource in distributed server could be archived and building cost of a new web server could be reduced due to an applicability to current web server. In addition, platform independent and efficient data management was obtained by using the XML-RPC protocol.

키워드

XML-RPC, 문서 관리, 분산 환경

Ⅰ. 서 론

인터넷의 급격한 발전으로 WWW(World Wide Web)은 우리 생활 깊숙이 파고들어 많은 웹 기반 프로그램들을 개발하였으며, 또한 웹 브라우저를 통해 임의의 플랫폼에서도 프로그램에 접근하여 정보공유와 경제활동을 하는 비즈니스 부분으로 그 활용분야를 넓혀가고 있다.

그러나 현재 비즈니스 관련 웹 개발환경은 서버/클라이언트 구조를 가지는 2-Tier환경에서 많

* 제주대학교 대학원 컴퓨터공학과 석사과정
** 제주대학교 대학원 컴퓨터공학과 박사과정
*** 제주대학교 통신컴퓨터공학부 컴퓨터전공 교수
접수일자 : 2004. 1. 2

은 사용자와 대용량의 데이터를 처리하기 위한 Business Logic과 Application Logic을 담당하는 미들웨어(Thin Server)를 갖춘 3-Tier환경으로 변해왔으며 지속적인 서비스를 위한 가용성과 성능향상 그리고 용량을 초과하는 서비스들을 처리하기 위해서 확장성이 요구되어 n-tier환경으로 변해가고 있다[1][2].

이러한 흐름 속에서 미들웨어 시스템은 분산환경의 자원들을 표현하고 통합 및 표준화하여 클라이언트들에게 서비스를 제공해 줄 수 있는 다양한 방식[3,4,5,6,7]으로 연구되고 있지만, 현재 분산환경의 통신수단으로 활용되고 있는 MicroSoft사의 COM/DCOM, OMG사의 CORBA, Sun사의 RMI/JavaBeans 등은 서로 호환성을 가지고 있지 않으며, 또한 객체기반 분산 컴퓨팅 환경에서 클라이언트는 ORB(Object Request Broker) 소프트웨어에 의존하는 형식으로 서버와 데이터를 교환하고 있어서 상호운영성의 결여 등이 문제점으로 지적되고 있다[8,9].

따라서 현재 분산 환경의 서버 상에 존재하는 동적인 문서들을 효율적으로 통합 및 표준화하여 클라이언트들에게 제공하는데 어려움을 가지고 있으며, 이에 따라 분산된 문서들을 효과적으로 저장하고 관리하는 서비스의 필요성이 제기되고 있다.

이 중에서 분산된 웹 환경의 문서들에 대해 플랫폼에 독립적으로 특정한 공간에 저장 및 관리하고 효과적으로 공급하기 위해 특정 자료로부터 추출한 정보를 검색에 사용될 수 있는 형태로 가공한 정보, 즉, 메타정보(Meta-Information)들을 미들웨어(Middleware)를 통해 활용하는 연구가 대두되고 있다[10].

현재 XML[11]이라는 데이터 표준 데이터 포맷을 이용하여 각각의 웹 환경의 자원들을 플랫폼 독립적으로 전달하기 위하여 XML 프로토콜인 XML-RPC[12,13,14], SOAP[15], WDDX등이 활용되고 있으며, 응용프로그램간의 단순한 데이터 교환뿐만 아니라 그에 적합한 처리를 할 수 있는 응용프로그램까지 전달 할 수 있으며 상호운용성도 우수하다[16].

본 논문에서는 다양한 XML 프로토콜[17]중에서 구조적 데이터를 표현할 수 있고, 분산 환경에서 통신수단으로 활용할 수 있는 XML-RPC 프로토콜을 사용하여 분산된 웹 서버에 있는 동적인 문서들의 정보를 관리하고 효과적으로 클라이언트들에게 제공하기 위한 문서 관리 시스템 모델을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 연구의 이론적 배경에 대해 알아보며, 3장에서는 제안하는 문서관리 시스템 모델에 대한 시스템 분석 및 설계를 설명하고, 4장에서는 시스템 구현 결과 및 고찰을 그리고 5장에서는 결론 및 향후 연구 방향을 제시한다.

II. 관련 연구

2.1 분산 시스템

분산 시스템 환경은 여러 개의 컴퓨터 환경과 통신망 속에서 시스템과 응용 프로그램들이 분산되어 제반 자원의 공유와 제어를 행하는 시스템 형태로 널리 이용되고 있다.

그러나 이러한 분산 시스템의 환경에서는 자원의 공유나 제어를 통한 시스템의 성능을 향상시키는 것에 반하여 분산된 중복 데이터 처리에 따라 통신 처리의 오버헤드 문제 해결, 효율적인 메커니즘의 활용이 주요한 문제점으로 대두되고 있다. 이러한 문제점을 해결하기 위하여 원격 프로시저 호출(RPC : Remote Procedure Call)이 제시되었는데 RPC는 분산처리 시스템과 네트워크 사이를 프로세스간의 통신 메커니즘 화하여 통신망의 과부하를 줄이고 단순성(Simplicity), 투명성(Trans-parency)의 특성을 갖는 분산 시스템 개발에 광범위하게 이용되고 있다.

분산 시스템의 구성 중 클라이언트/서버 모델은 분산 처리의 일반적인 모델로서, 서비스를 제공하는 원격 서버와 서비스를 요구하는 클라이언트로 구성된다. 클라이언트/서버 모델은 이미 X Window System, NFS(Network File System), NIS(Network Information Service), 데이터베이스 관리 시스템 등 다양한 분야에서 사용되고 있다.

이런 경향에 따라 중앙집중식으로 개발되었던 애플리케이션들이 클라이언트/서버 구조의 분산 형태로 다시 설계, 개발되고 있으며 새로 개발되는 애플리케이션들도 클라이언트/서버 형태로 개발되어지고 있다. 이러한 클라이언트/서버 형태의 애플리케이션 작성은 소켓(Socket)과 같은 트랜스포트 레벨의 통신 API 사용과 통신의 세부적인 내용을 애플리케이션 개발자가 정확히 알고 다루어야 한다는 점이 기존 중앙집중식 애플리케이션보다 개발의 생산성을 떨어뜨리지만 이런 문제를 해결하기 위해 RPC를 기존의 프로그래밍 언어에서 이용할 경우 중앙집중식 프로그램의 작성과 거의 동일한 형태로 제공함으로서 생산성 저하를 줄여주고 있다. 하지만, 분산기술은 웹과의 직접적인 호환성을 갖지 못하므로 이를 위해 XML을 이용하고자 하는 많은 노력을 하고 있다.

2.2 RPC (Remote Procedure Call)

RPC는 분산형 컴퓨팅의 클라이언트 서버 모형을 수행하기 위하여 가장 많이 쓰이는 패러다임이다. 또한 RPC는 한 프로그램이 네트워크상의 다른 컴퓨터에 위치하고 있는 애플리케이션에 서비스를 요청하는 경우에 사용되는 프로토콜로서, 이때 서비스를 요청하는 애플리케이션은 네트워크에 대한 상세 내용을 알 필요가 없으며, 그리고 RPC는 클라이언트/서버 모델을 사용하며 요청한 처리 결과가 반환될 때까지 대기하는 동기 운영형태를 가진다.

그러나 가벼운 프로세스의 사용이나, 같은 주소공간을 공유하는 스레드 등은 여러 개의 RPC들이 동시에 수행될 수 있도록 허용한다. RPC를 사용하는 프로그램 문장들이 실행 프로그램으로 컴파일 될 때, 컴파일 된 코드 내에 RPC의 대리인처럼 동작하는 스텁(Stub)이 포함된다. 이러한 스텁 프로시저는 원격 프로시저가 클라이언트에 의해 호출될 때 생성되며 로컬 프로시저를 원격 프로시저 콜로 서버 측에 변환해 주는 것이 주목적이다. 그림 1은 RPC 호출 메카니즘을 보여준다.

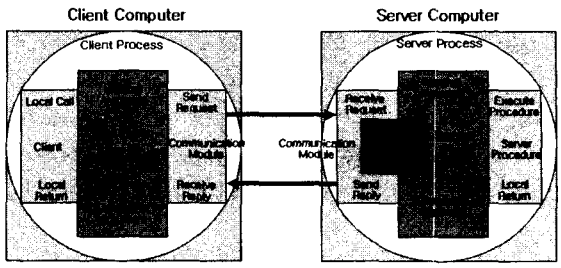


그림 4. RPC 호출 메카니즘
Fig. 1 RPC Call Mechanism

2.3 CORBA 와 XML

최근 분산 컴포넌트 기술인 OMG(Object Management Group)의 CORBA(Common Object Request Broker Architecture)와 웹과의 연동을 위한 SOAP(Simple Object Access Protocol)을 이용한 SCOAP(Simple Corba Object Access Protocol)이 제안 되어 있는 상태이다. SOAP 프로토콜은 Microsoft사에 의해 제안되었으며, 플랫폼 독립적으로 인터넷에 분산된 서비스, 오브젝트, 또는 서버에 접근할 수 있도록 HTTP 혹은 SMTP상에서 단순히 서비스 요청/응답에 대하여 표준 XML 문서 구조를 정의하여 서비스 제공업자와 사용자 간에 상호 교환하도록 구성되어 있다.

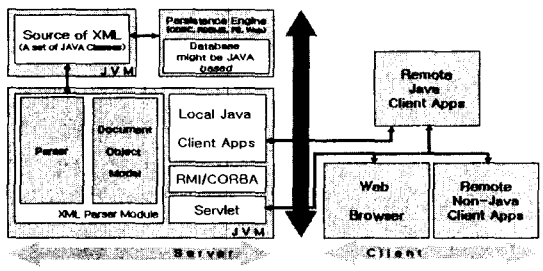


그림 5. Web API를 이용한 CORBA와 XML의 연동 구조
Fig. 2 Connection Structure of CORBA and XML using the Web API

그림 2는 JVM(Java Virtual Machine) 기반의 Web API 를 이용한 경우의 CORBA와 XML의 연동구조를 나타내고 있다. 이 경우 XML과 CORBA의 연동에서 XML 문서의 처리방법은 CORBA 클라이언트와 CORBA 서버 측에서

XML 문서를 직렬화 형태로 전송하여 CORBA 클라이언트와 서버 측에서 이 XML 문서를 처리하는 방법과, CORBA 클라이언트가 CORBA 서버 측에 요청하여 받은 응답을 XML로 변형하는 방법이 있다. 그러나 이 구조는 기존의 CORBA 환경에서 XML 문서를 처리해야 하기 때문에 CORBA 클라이언트객체와, CORBA 서버객체, IDL 등을 수정해야만 하는 문제점이 있다.

메타 데이터에는 XMI(XML Metadata Interchange), W3C의 RDF(Resource Description Framework), OMG의 MOF(Meta Object Facility), SMIF(Stream based Model Inter- change Format)등이 있다. UML기반의 메타모델을 정의하여 분산 환경에서 메타데이터의 교환 및 전송을 위한 XML 형식의 메타데이터를 생성하여 데이터베이스에 저장한다. 이 XML 형식의 메타데이터를 이용하는 것은 인터넷상에서 자료전달 과 교환 시 텍스트 양식으로 전달하므로 플랫폼에 따라 발생할 수 있는 호환성 문제를 해결할 수 있다. 그러나 이 구조는 서로 다른 메타데이터들 간의 호환성에 문제가 제기되고 있다.

2.4 XML-RPC

XML-RPC는 가장 단순하며 간단하게 웹 서비스에 접근 할 수 있는 방법 중 하나이며, 컴퓨터가 다른 컴퓨터에 있는 프로시저를 쉽게 호출할 수 있도록 해준다. XML-RPC는 이전 통신 수단을 재사용하여 컴퓨터에 있는 각종 애플리케이션간의 통신을 지원하게 된다. 확장 마크업 언어인 XML은 RPC를 기술하는 어휘를 제공하며, 이것은 컴퓨터들 사이에서 HTTP를 이용하여 전달된다. XML-RPC는 개발과정을 매우 간단하게 해주며, 서로 다른 종류의 컴퓨터들이 쉽게 통신할 수 있도록 해준다.

가장 기본적인 레벨로는 XML-RPC를 통해서 네트워크를 오고 가는 '함수 호출'을 만들 수 있다. RPC 아키텍처를 XML과 HTTP 기술에 혼합한 XML-RPC를 이용하여 컴퓨터들이 네트워크에 있는 자원을 쉽게 공유하도록 할 수 있다. 이는 새로운 것이 진행되는 중에 이미 만들어 놓은 시스템을 단순히 읽고 재사용하는 것이 아니라, 최선의 효과를 낼 수 있도록 프로그램들을 서로 혼합하고 재배치하는 것을 의미하는 것으로서, 사용자들이 원하는 정보에 직접 접근하도록 할 수 있다.

웹 프로토콜과 기반 구조 재사용하기 위해 XML-RPC는 전송 프로토콜을 재사용한다. HTTP 프로토콜은 웹 서버에 적합한 환경에서부터 프로그램 내부에서 직접 사용하는 마이크로서버까지 많은 개발 환경에서 만들어졌으며 전송을 위해 수많은 문서를 조합하는 과정을 수행해왔으며, 웹 서버와 웹 환경의 방화벽 등을 수년 동안 지원해왔다.

호출된 메소드의 식별자를 열어서 매개변수를 제공하고, 그 메소드가 어떤 값을 리턴해야 할지 결정하는 것 등 여러 가지 측면을 볼 때 HTTP는 RPC 기반의 프로토콜이다. HTTP는 서로 다른 종류의 컨텐츠를 일치시키고 인코딩하는 표준 규약인 MIME (Multipurpose Internet Mail Extensions)에 기초한 비교적 개방적인 접근 방법이다. 따라서 웹 사이트에 필요한 많은 종류의 컨텐츠를 유연하게 다룰 수 있다. 이러한 유연성은 RPC 프로토콜 요청에 따른 다양한 응답 방법들을 수행하기에 충분하다.

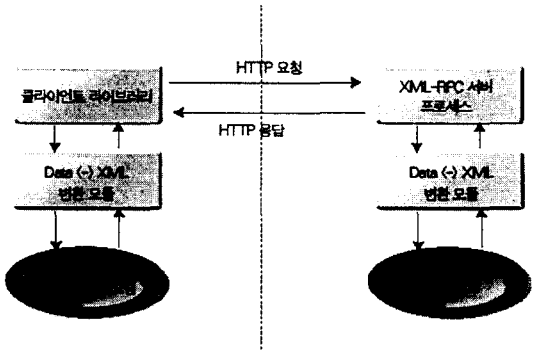


그림 3. XML-RPC 전송방식
Fig. 3 Transmission method of XML-RPC

XML-RPC는 기존의 HTTP 프로토콜과 기반 구조의 장점을 유지하면서 RPC 접근 방법을 실제로 구현할 수 있도록 한다. HTTP는 모든 종류의 개발환경과 운영체제에서 동작할 수 있고, XML 파서는 필수품과도 같기 때문에 어떠한 환경에서

도 XML-RPC 툴킷을 사용해서 시스템을 구성할 수 있다.

III. 시스템 분석 및 설계

3.1 제안한 전체 시스템 모델 구조

본 논문에서 제안하는 시스템 모델은 각각의 분산된 서버에 존재하거나 등록할 문서에 대해 효율성과 신뢰성을 보장하며 클라이언트들에게 정보를 제공하기 위하여 문서정보의 등록/검색/삭제 등의 작업들을 미들웨어와 XML-RPC 프로토콜을 도입하여 문서 정보를 웹 애플리케이션을 통해 제공 및 저장하고 메타정보들을 데이터베이스에 저장하기 위한 시스템을 모델링한다.

클라이언트들은 분산되어 있는 서버에 상관없이 웹 브라우저를 통해 문서에 대한 메타정보를 제공 및 등록하는 웹 애플리케이션 서버에 접근하여 자신이 원하는 문서 정보들에 대하여 등록/검색/삭제 등의 작업을 하게 되고, 그에 대한 결과값을 데이터베이스에 저장하여 신뢰할 수 있는 메타 정보를 미들웨어를 통해 제공받는다. 제공받은 정보를 가지고 클라이언트들은 직접 문서가 저장된 분산된 서버로 접근하여 문서를 얻어 올 수 있게 된다.

웹 애플리케이션 서버와 미들웨어 그리고 데이터베이스는 각각 분리되어 있으며, 각각의 시스템들은 XML-RPC 프로토콜을 이용해 통신을 하게 된다.

웹 애플리케이션 서버와 분산된 서버에서 이벤트가 발생할 때마다 미들웨어는 이벤트 프로세스를 통해 마샬링된 Documents()객체를 전송받아 언마샬링하여 관계형 데이터베이스 매핑 과정을 거쳐 문서정보들을 저장하게 된다.

저장된 메타정보는 클라이언트들의 문서에 대한 관리에 사용되게 되며, 제안하는 시스템은 크게 웹 애플리케이션 서버, 미들웨어 시스템, 분산 서버 시스템으로 구분된다.

그림 4의 제안 시스템 모델 구조를 살펴보면 클라이언트들은 각각의 분산된 서버에 대해서 알 필요가 없게 됨으로서 위치 투명성이 보장된다. 기존의 웹 애플리케이션 시스템에서 자료를 처리하던 비즈니스 로직을 미들웨어로 분리하고 오직 보여주는 프리젠테이션 부분에 집중할 수 있게 되어 프리젠테이션 부분과 비즈니스로직 부분에 대한 처리에 유연성이 생겨 추가적으로 웹 애플리케이션 서버와 미들웨어의 확장 가능성도 유연하다.

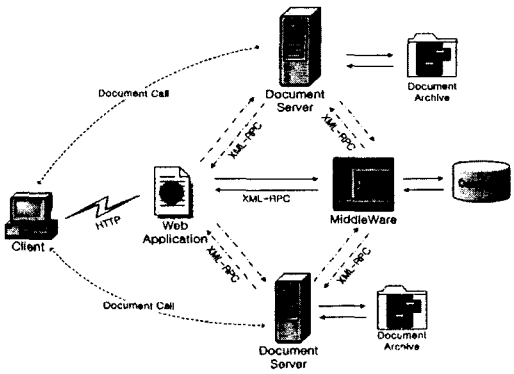


그림 7. 제안한 시스템 모델 구조
Fig. 4 Model structure of proposed system

그림 5는 문서 등록과정에 대한 데이터 흐름과 클라이언트의 요청에 대한 반응을 설명한다.

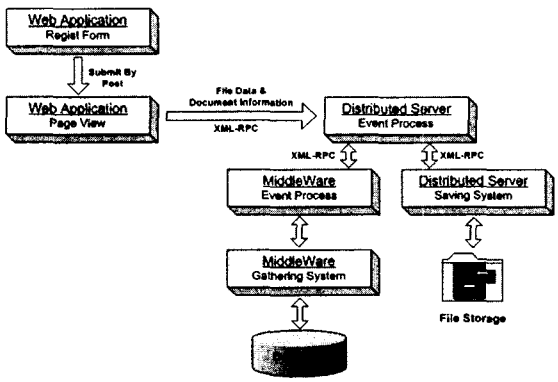


그림 8. 문서 등록 흐름
Fig. Registry flow of document

클라이언트는 웹 애플리케이션 서버로 접근하여 등록 폼을 통해 선택한 서버와 문서 그리고 저자에 대한 값을 입력하고 등록하게 되면, XML-RPC를 통해 해당 분산 서버로 문서 정보들과 파일이 전송되고, 파일은 분산 서버의 특정 디렉토리에 저장 되면, 등록된 문서 정보들은 다시 미들

웨어로 XML-RPC를 통해 전송되어 이벤트 프로세스에 의해 Gathering 시스템으로 넘겨주고 데이터베이스 매핑 과정을 통해 등록 정보가 저장되게 된다.

그림 6에서 보듯이 웹 애플리케이션 서버는 클라이언트가 검색하는 정보를 미들웨어에 요청하고 결과 값만 보여주기 위해 가공하여 클라이언트에게 되돌려 주고, 데이터베이스에 접근하여 문서에 대한 메타정보와 위치 정보를 얻어내는 일은 미들웨어가 맡는다. 이 때, 각 로직 사이에 통신은 최대한 간결하게 이루어지고 각 로직이 맡은 일에만 집중할 수 있도록 도와주기 때문에 가용성을 높일 수 있다.

이러한 과정을 거쳐 최종적으로 클라이언트가 어느 서버에 접근을 해야 원하는 문서를 얻을 수 있는지를 알 수 있게 되는데, 웹 애플리케이션의 구현 정도에 따라 서버에 대한 정보도 클라이언트가 알 필요 없도록 가려 투명성을 확보할 수 있다.

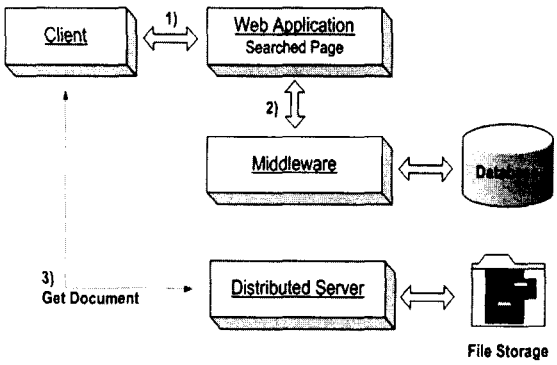


그림 9. 문서 검색 흐름 [순서 1) - 2) - 3)]
Fig. 6 Search flow of document

그림 7에서 보면 문서를 삭제하기 위해서 클라이언트는 삭제할 문서를 검색된 페이지를 통해 얻게 되고 삭제를 웹 애플리케이션을 통해 요청하게 되면 미들웨어는 삭제에 대한 이벤트를 받아 분산된 서버와 Gathering 시스템에게 알리게 된다.

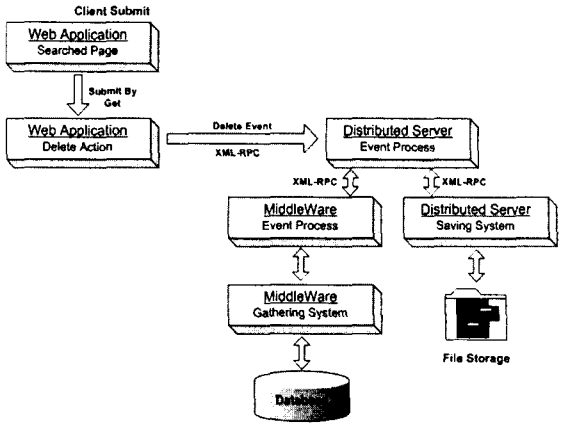


그림 10. 문서 삭제 흐름
Fig. 7 Delete flow of document

실질적인 문서파일은 분산된 서버에 저장된 File Storage에서 삭제되고, 동시에 데이터베이스에 저장된 메타 정보가 삭제되어, 신뢰 할 수 있는 정보를 유지하게 된다.

그림 8는 문서 수정에 대한 흐름을 보여주고 있다. 문서 갱신 과정은 문서 등록 과정을 Registry Module로, 문서 삭제 과정을 수행하는 것을 Delete Module로 놓고 두 과정을 이어서 진행함으로서, 독자적인 모듈보다는 기존 모듈을 이용함으로서 구현이 가능하다.

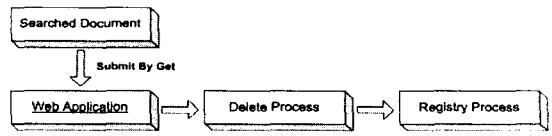


그림 11. 문서 수정 흐름
Fig. 8 Modify flow of document

3.2 미들웨어와 분산된 서버와의 연동구조

웹 애플리케이션 서버는 등록/검색/삭제/수정 등의 작업에 대한 이벤트 발생시 XML-RPC를 통해 분산되어 있는 서버로 문서정보 및 파일을 보내게 되며 파일은 각각의 지정된 서버에 저장이 되며, 문서에 대한 정보는 미들웨어와 분산된 서버와의 XML-RPC 통신을 통해 Document객체로 전송하면, 미들웨어는 전송된 정보를 데이터베이스 매핑 시스템을 통해 데이터베이스에 저장하고

클라이언트의 문서정보 요청 시 웹 애플리케이션 서버에 결과를 보내준다.

미들웨어를 통해 모든 문서 정보가 수합되고 데이터베이스에 저장이 되며, 각각의 분산 서버들은 이벤트 처리에 대한 결과 값을 실시간으로 미들웨어에 전달하게 된다. 이러한 과정을 통해 클라이언트는 항상 신뢰할 수 있는 실시간 문서변환 정보 및 파일을 사용할 있게 된다. 그림 9는 미들웨어와 분산된 서버와의 연동구조를 나타낸다.

3.3 웹 애플리케이션 서버시스템

웹 애플리케이션 서버는 클라이언트가 익숙한 웹 인터페이스를 제공한다. 클라이언트들은 제공되는 서버들의 목록을 지정하고 저자/문서에 대한 간단한 값을 입력하고 문서파일과 문서정보를 저장할 수 있는 폼과 문서를 검색하는 폼을 제공한다. 웹 애플리케이션 서버시스템은 모든 비즈니스 로직 부분은 미들웨어에 위임하고, 프리젠테이션 부분만 관여하기 때문에 시스템이 유연하고, 미들웨어에 대한 정보 값만 알고 있으면 되므로 미들웨어 확장이나 웹 애플리케이션 확장의 편의성을 제공 받게 된다.

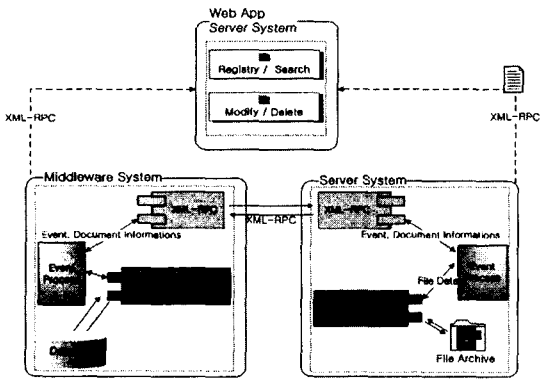


그림 12. 미들웨어와 분산된 서버와의 연동구조
Fig. 9 Connection structure of middleware and distributed server

따라서 다양한 웹 기반의 서비스와 맞물려 웹 기반의 문서관리 시스템을 클라이언트에게 제공하기 쉽고, 운영 중인 서버에도 바로 연동 할 수 있어 보다 효율적이라 할 수 있다.

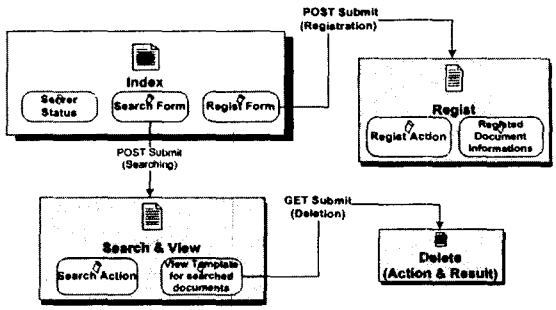


그림 13. 웹 애플리케이션 이벤트 흐름
Fig. 10 Event flow of Web application

3.4 미들웨어 시스템

미들웨어는 웹 애플리케이션 서버에 의해 실시간으로 발생하는 문서 이벤트들에 대한 정보들을 XML-RPC 프로토콜을 사용해 각각의 분산된 서버의 이벤트 프로세스로 전달하게 되고, 이벤트 처리 결과를 미들웨어 시스템에 실시간으로 통보하게 된다. 처리된 이벤트들을 Gathering System으로 넘겨주며 관계형 데이터베이스에 맞게 매핑하여 저장하게 된다. 또한 미들웨어 시스템은 분산되어 있는 서버와 웹 애플리케이션 서버에 대한 정보들을 설정파일로 가지고 있으면서 실시간으로 변하는 이벤트들을 처리하게 되며, 이 시스템의 가장 큰 역할은 문서 정보들을 수집하고 저장하여 클라이언트에게 신뢰 할 수 있는 메타정보를 제공하기 위한 비즈니스 로직이 구현되는 부분이다.

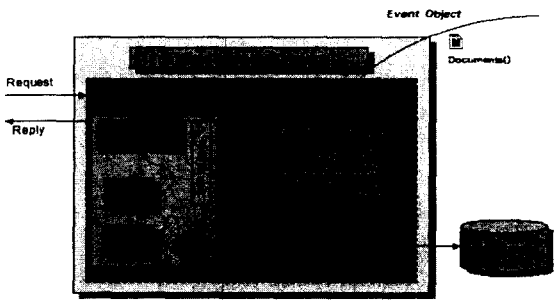


그림 14. 미들웨어 시스템 구조
Fig. 11 Structure of middleware system

3.4.1 이벤트 처리 시스템

동적으로 변화하는 문서 정보를 유지하기 위한

수단으로 분산된 서버 시스템에서 처리한 이벤트 처리 결과를 받아서 Gathering 시스템으로 전달하는 처리를 하게 된다. Document객체를 전달받아 파싱과정을 거치고, 언마샬링하여 문서 객체 정보에서 문서정보를 수집한다. 또한 클라이언트들이 웹 애플리케이션을 통해 문서정보인 메타정보들을 검색할 때, 데이터베이스에 저장되어 있는 문서 정보를 제공하는 역할도 하게 된다.

3.4.2 Gathering 시스템

이벤트 프로세스에 의해 넘겨받은 문서정보들을 수집하여 데이터베이스에 매핑 시키는 역할을 하는 시스템이다. 또한, 웹 애플리케이션에서 검색에 대한 이벤트들을 처리하여 결과 값을 이벤트 프로세스에게 반환해 주는 역할을 한다.

데이터베이스의 주된 역할은 문서정보에 대한 검색정보이기 때문에 본 논문에서는 관계형 데이터베이스를 사용하게 된다.

표 1은 데이터베이스의 테이블 구조를 나타내며, 각 Field는 다음과 같은 내용을 저장한다.

표 1. 데이터베이스 스키마
Table. 1 Database schema

Filed	Type	Null	Key	Default	Extra
id	int unsigned		Primary	notnull	auto increment
name	varchar			notnull	
server	varchar			notnull	
author	varchar	yes			
mimetype	varchar			notnull	
regdate	double	yes			

- id : DEDMS table의 문서정보에 대한 index를 지원하며, Primary key 값.
- name : 문서 이름에 대한 값.
- server : 등록된 서버에 대한 이름값.
- author : 문서 저자에 대한 값.
- mimetype : 문서형식에 대한 값.
- regdate : 문서 등록 / 수정 / 삭제 등에 대한 시간 값.

그림 12는 이벤트 발생 시 XML-RPC를 통해 이벤트 프로세스가 받는 객체의 XML 파일의 구조이다.

```
<params>
  <param>
    <value><array><data><value><struct>
      <member>
        <name>mimetype</name>
        <value><string>image/jpeg</string></value>
      </member>
      <member>
        <name>author</name>
        <value><string>jjcole</string></value>
      </member>
      <member>
        <name>name</name>
        <value>
          <string>CIMG0553.JPG</string>
        </value>
      </member>
      <member>
        <name>regdate</name>
        <value>
          <double>1067792581.84</double>
        </value>
      </member>
      <member>
        <name>server</name>
        <value><string>jerimo</string></value>
      </member>
    </struct></value></data></array></value>
  </param>
</params>
```

그림 12. Documents()객체 XML파일의 구조
Fig. 12 Documents() Object XML File Structure

3.5 분산된 서버 시스템

각각의 분산된 서버들은 미들웨어에 대한 정보를 가지고 XML-RPC 프로토콜을 통해 통신을 하게 된다. 웹 애플리케이션 서버로부터 문서정보와 파일을 받아 저장/검색/삭제/수정 등의 이벤트 처리를 하고 미들웨어로 결과에 대한 정보를 보내주는 시스템으로 XML-RPC Server Module, XML-RPC Client Module, 이벤트 프로세스, Saving 시스템으로 구성된다.

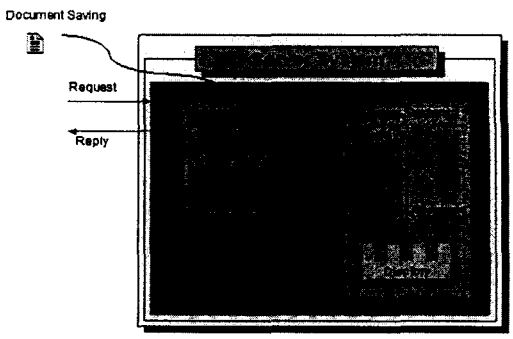


그림 13. 서버 시스템 구조
Fig. 13 Structure of server system

3.5.1 이벤트 처리 시스템

웹 애플리케이션에서 발생한 저장/삭제/수정 등의 이벤트들을 처리하는 시스템으로 미들웨어에서 전달받은 Documents( ) 객체 정보들과 서버 상태, 문

서에 대한 처리 상태 등을 Saving 시스템과 미들웨어로 보내고 받는 처리를 하게 된다.

현재 서버의 상태 및 갱신 정보들을 실시간으로 알려주는 역할을 하게 되며, 미들웨어의 이벤트 프로세스와 마찬가지로 XML-RPC를 통해 통신하며 미들웨어에 대한 정보를 가지고 있기 때문에 플랫폼 독립적이며 확장이 용이하다.

3.5.2 Saving 시스템

이벤트 프로세스를 통해 전달 받은 문서(파일)을 지정된 공간에 저장하는 시스템이며, 처리 결과를 이벤트 프로세스를 이용하여 미들웨어로 통보하며, 문서 삭제 /수정의 요청들을 처리한다.

IV. 시스템 구현 결과

4.1 개발 환경

- ▶ System : 펜티엄 IV 2.0GHz 256RAM / 펜티엄 II 300MHz 96RAM
- ▶ Server : Apache or any HTTPD server
- ▶ PHP 4.1 or higher
- ▶ Python 2.3 or higher
- ▶ MySQLdb (mysql-python)
- ▶ MySQL Server
- ▶ XML-RPC Library
- ▶ Any Web Browser

4.2 구현 결과

4.2.1 웹 애플리케이션 시스템

웹 애플리케이션 시스템은 HTTPD Server 기반의 CGI와 PHP를 가지고 구현되었으며, 사용자를 위한 UI(User Interface) 부분을 담당한다.

UI는 크게 Documents에 대한 검색부분과 등록부분 그리고 삭제부분으로 나누어지며, 등록과 검색 시 분산된 서버를 선택하는 형식으로 구현하였다. 문서 등록 과정에서는 선택된 서버로 문서(파일)와 문서 정보가 이벤트 객체로 XML-RPC를 통해 통신하게 된다.

웹 애플리케이션 서버는 분산된 서버의 현재 상태를 나타내주게 되므로, 클라이언트는 서버의 상태를 눈으로 확인 할 수 있게 된다. 그림 15는 웹 애플리케이션 서버의 등록/검색/삭제 등의 메인화면을 나타낸다.

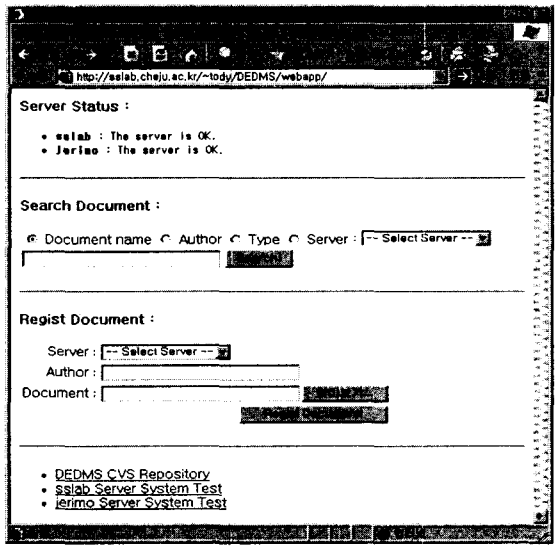


그림 14. 웹 애플리케이션 시스템 메인화면
Fig. 14 Main window of Web application system

웹 애플리케이션 서버는 미들웨어를 통해 실시간으로 변화하는 문서정보를 얻어오기 때문에 미들웨어에 대한 설정정보가 필요하다.

```
<?php
/*
 * Configurations of Web App. System
 */

$middleware_host = 'sslab.cheju.ac.kr';
$middleware_port = 80;
$middleware_path = '/~tody/DEDMS/middleware/middleware.cgi';
?>
```

그림 15. 미들웨어에 대한 설정정보
Fig. 15 Configuration Information about the middleware

그림 16은 등록 과정에서 Documents( )객체로 전송되는 XML 구조를 덤프한 화면이다. 문서를 등록하게 되면 처리된 결과를 그림 17과 같이 나타내주며, 클라이언트가 등록하는 문서(파일)는 64바이트로 인코딩되어 분산된 서버로 전송되며, 분산된 서버에서는 64바이트로 디코딩하여 지정

된 File Storage에 저장하게 된다. 처리 결과는 미들웨어로 통보하여 실시간으로 저장하게 된다.


```
<params>
  <param>
    <value><struct>
      <member>
        <name>filedata</name>
        <value><base64>
          VGtsTFJTQzl4Ym5mSU1XcHNlTEhwUTBL
          Zkh3Z0tHMXRLUUI4ZkNBeU16QWdmSHdn
          TWpRd01IeDhJREkxTUNCOGZDQXI0akFn
          ZkH3ZwOKTWpjd01IeDhJREk0TUNCOGZDQ
          X1PVEFnZkh3Z0I6QXdJSHg4SURNeE1DQjh
          mQOF6TWpBZ2ZId05Dbng4SUUxbGJuTWdm
          SHdnZkh3ZwOKZkh3Z1ZWTWd0eUI4ZkNC

          .....(생략).....

          Z055QjhmQOJWVX1BNElIeDhJR1ZUSURrZw0
          KZkh3Z1ZWTWdNVEFnZkh3Z1ZWTWdNVE
          VnZkh3Z2Z2Id2dmSHdnZkh3Z2Z2Id05DZz09DQ
          o=
        </base64></value>
      </member>
      <member>
        <name>name</name>
        <value>
          <string>nike_footware_size.txt</string>
        </value>
      </member>
      <member>
        <name>author</name>
        <value><string>jjoole</string></value>
      </member>
    </struct></value>
  </param>
</params>
```


그림 16. 객체의 XML 구조(등록)
Fig. 16 XML structure of object(Registry)

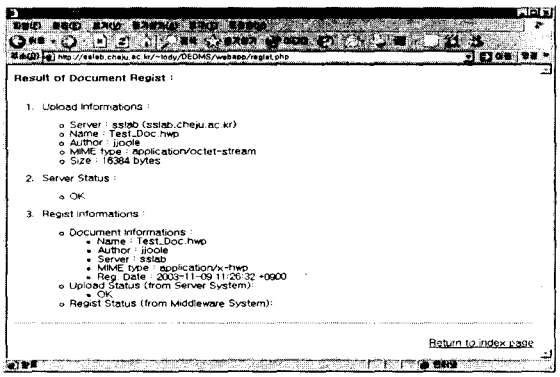


그림 17. 문서 등록 결과 화면
Fig. 17 Result window of document registry

클라이언트는 검색과정을 통해 문서를 얻어 올 수 있고, 삭제도 가능하다. 모든 이벤트에 대한 처리 결과는 미들웨어에서 제공받기 때문에 실시간으로 변화하는 정보에 대한 신뢰성을 갖게 된다.

문서 검색은 문서이름, 저자, 문서타입, 서버 의 형태로 이루어지며, 검색된 결과는 그림 18과 같고, 하나의 서버에 저장된 파일을 모두 검색한 화면이다.

클라이언트는 문서 정보를 확인 후 문서(파일)를 요청 가능하며, 문서(파일)는 하이퍼링크를 따라 지정된 서버로 직접 요청하고 다운로드 할 수 있다.

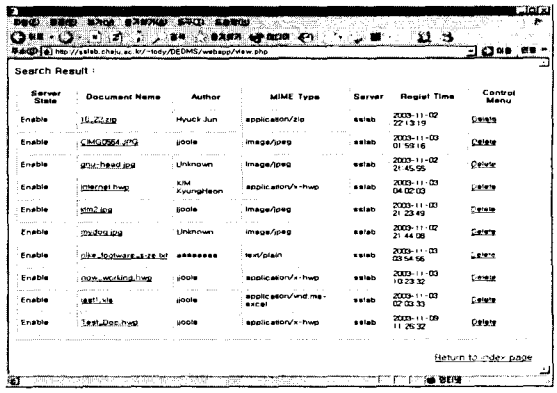


그림 18. 문서 검색 결과 화면
Fig. 18 Result window of document search


```
<params><param><value><array>
  <data><value><struct>
    <member>
      <name>mimetype</name>
      <value>
        <string>image/jpeg</string>
      </value>
    </member>
    <member>
      <name>author</name>
      <value><string>jjoole</string></value>
    </member>
    <member>
      <name>name</name>
      <value>
        <string>CIMG0553.JPG</string>
      </value>
    </member>
    <member>
      <name>regdate</name>
      <value>
        <double>1067792581.84</double>
      </value>
    </member>
    <member>
      <name>server</name>
      <value><string>jerimo</string></value>
    </member>
  </struct></value>
  <value>..... (생략).....</value>
  </data>
</array></value></param></params>
```


그림 19. 객체의 XML 구조(검색)
Fig. 19 XML structure of object(Search

클라이언트는 검색된 화면에서 삭제과정도 바로 수행할 수 있으며 Delete링크를 선 클릭하게 되면, 분산된 서버의 File Storage에 저장된 문서가 바로 삭제되고, 미들웨어에 그 정보를 실시간으로 알려주게 된다. 그림 20은 삭제(Delete) 한

결과 화면이고, 그림 21는 문서 삭제 시 객체의 XML 구조를 나타낸다.

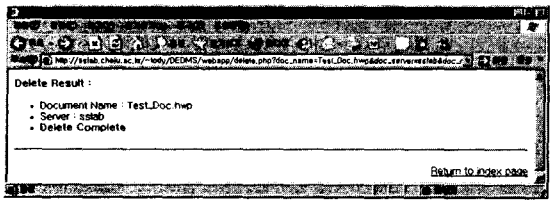


그림 20. 문서 삭제 결과 화면
Fig. 20 Result window of document delete

```
<params>
  <param>
    <value>
      <string>nike_footware_size.txt</string>
    </value>
  </param>
  <param>
    <value>
      <string>sslab</string>
    </value>
  </param>
</params>
```

그림 21. 객체의 XML 구조(삭제)
Fig. 21 XML structure of object(Delete)

4.2.2 미들웨어 시스템

미들웨어 시스템은 클라이언트의 문서관리에 대한 비즈니스 로직 부분을 담당하게 되며, 문서의 등록/삭제 등의 이벤트 발생 시 분산된 서버로부터 처리된 결과를 받아 데이터베이스에 저장하고 클라이언트의 문서 검색에 대하여 신뢰할 수 있는 정보를 제공해준다. 연결되어있는 데이터베이스와 분산된 서버의 정보를 가지고 실시간으로 발생되는 이벤트에 반응하게 된다.

4.2.2.1 이벤트 처리 시스템

분산된 서버에서 처리되는 결과 값을 XML-RPC 통신을 통해 Document( )객체로 전달받고, ServerList를 불러와 그 값을 Gathering 시스템을 통해 데이터베이스에 매핑하고 저장하게 된다. 또한, 클라이언트의 문서 검색 시 결과 값을 XML-RPC 통신을 통해 웹 애플리케이션 서버로 전달해 준다.

4.2.2.2 Gathering 시스템

이벤트 프로세스에서 전달받은 정보를 수집하고 데이터베이스에 대해 삽입 / 삭제 / 검색 등의 작업을 처리한다.

4.2.3 분산 서버 시스템

웹 서버 환경으로 구성된 시스템이거나 독자적인 XML-RPC Server Module이 서비스 되면 미들웨어와 웹 애플리케이션과의 연동이 가능한 구조이다.

XML-RPC는 HTTPD(웹 서버)를 기반으로 통신하지만, XML-RPC Server Module과 XML-RPC Client Module을 자체적으로 지원하기 때문에 보다 더 확장 가능한 구조이다. 웹 서버 기반 구조로 응용되는 부분은 XML-RPC 자체 모듈은 보안이나, 기타 인증 부분을 웹 서버 환경에 위임함으로서 보다 더 효과적인 서비스가 가능하기 때문이다.

분산 서버 시스템은 크게 이벤트 처리 시스템과 Saving 시스템으로 구분되며, 웹 애플리케이션을 통해 저장되는 문서(파일)는 File Storage에 저장되고, 클라이언트의 요청 시 파일을 직접 전해주게 된다.

분산된 서버 시스템에는 데이터베이스가 구축되지 않을 수도 있기 때문에, 구현 시 File Storage를 사용하여 특정 위치에 문서(파일)를 저장하게 된다.

```
# 서버 이름. uniq
server_name = 'sslab'

# base_dir은 웹에서 접근이 가능해야 함.
base_dir = '/home/tody/public_html'
base_uri = '/~tody'
document_dir = base_dir + '/documents'

# Middleware 주소
# cgi면 http에 알맞는 url로, 따로 돌아가는 서버라면 '
주소:포트'로.
middleware_url ='http://sslab.cheju.ac.kr/
                 tody/DEDMS/middleware/middleware.cgi'
```

그림 25. 분산 서버 설정 환경
Fig. 25 Configuration environment of distributed server

4.2.3.1 이벤트 처리 시스템

미들웨어의 이벤트 처리 시스템과 비슷한 역할을 하게 된다. 웹 애플리케이션에서 발생한 이벤트(저장/삭제/수정)들을 XML-RPC 통신을 통해 Documents( )객체로 전달받고, Saving 시스템에 전달하고 처리한 이벤트 결과들을 전달하는 역할을 한다.

4.2.3.2 Saving 시스템

이벤트 처리 시스템에서 넘겨받은 정보에서 Base_dir, Base_uri, Document_dir 들을 체크하고 저장/삭제 등의 작업을 하게 된다. 웹 애플리케이션 서버 시스템에서 64바이트로 인코딩 된 파일을 디코딩하여 지정된 디렉토리에 저장하게 된다.

V. 결 론

본 논문에서는 분산 환경에 있는 문서 자원들의 정보를 저장/관리하고 클라이언트에게 신뢰할 수 있는 정보들을 효과적으로 제공하기 위하여 XML-RPC 기반의 분산 환경 문서관리 시스템을 설계하고 구현하였다.

XML-RPC를 사용함으로서 다양한 웹 환경하의 특정 플랫폼 환경에 따른 프로토콜을 개발하지 않아도 되었으며, 웹 서버와 연동되어 운용되므로 네트워크의 세세한 부분과 보안을 웹 서버에 위임할 수 있어 보안 및 관리를 보다 간단하게 할 수 있으며, 실제 서비스되고 있는 다양한 웹 서버환경에 바로 적용시킬 수 있어 새로운 서버 구축비용을 절감 할 수 있었다.

또한, XML-RPC의 전송은 HTTP를 이용하고 인코딩은 XML을 사용하여 분산 환경에서 메소드의 매개변수가 인코딩 가능한 것이라면 XML-RPC로 그 메소드를 호출할 수 있으므로 분산된 시스템에 있는 클라이언트가 네트워크의 또 다른 부분에 있는 서버에서 실행될 수 있는 작업을 요청할 수 있다.

동적으로 변화하는 분산 환경의 웹 서버 문서들을 실시간으로 획득하기 위해 이벤트 처리 방식을 사용하고, 문서교환은 웹 애플리케이션 서버와 클라이언트가 직접통신하게 함으로서 미들웨어의 부담을 줄일 수 있었고, 획득한 정보는 데이터베이스에 저장하여 관리되므로 신뢰할 수 있는 메타정보를 검색하여 요청한 클라이언트에게 제공하고 할 수 있었다.

또한 현재 서비스 되고 있는 웹 서버뿐만 아니라 자체적으로 내장한 서버/클라이언트 모듈을 사용해 서비스가 가능하고, java, perl, php, python, asp등 많은 프로그래밍언어에서 XML-RPC에 대한 API를 제공하고 있기 때문에, 플랫폼 독립적이며 원격 프로시저 호출에서 다른 원격지 서버에 안정적으로 접근할 수 있었다.

향후 연구 과제로는 메시지 암호화를 통한 안전한 메시지 교환이 이루어져야 하고, 인증과 권한 부분에 대한 보안 연구가 필요하다. 또한 미들웨어 서버의 다운 시 이벤트 처리를 유지하기 위한 다중 서버의 클러스터링 지원을 위해 Server-side cashing이 필요하겠다.

참고 문헌

[1] BIOMEDICAL ENGINEERING APPLICATIONS BASIS COMMUNICATIONS, A Middleware Architecture to Improve the Efficiency of Web-Based Tel-emedicine Application , vol. 12 No.3, [2002]

[2] 이달상, 김태화, 반상우, "분산 환경에서 자바빈즈를 이용한 전자상거래 프레임워크의 설계" (産業技術硏究誌, Vol.15 No.1, [2001]

[3] CORBA Specification, Ver 2.4.2, OMG, http://www.omg.org"

[4] 진병률, 정지문, 최성, 우성구 , "이질 환경을 위한 XML 미들웨어 시스템 연구" 반도체장비 학술심포지움, [2001]

[5] 황준, 김영신 "실시간 분산 컴퓨팅을 위한 미들웨어 설계 및 구현" 자연과학논문집, Vol.13 No. 1, [2001]

[6] 조성연, 장주만, 전병태 "분산 컴퓨팅 환경과 미들웨어" JOURNAL OF COMPUTER SCI-E-ENCE & ENGINEERING TECHNOLOGY, V-ol.1 No.1, [1999]

[7] 이재완, 전병인 "CORBA를 기반으로 한 XML 정보검색 시스템 통합 구현", 情報通信技術硏究論文集, Vol.4 No.1, [2000]

[8] 오길호, Distributed Object Oriented Software System에 관한 연구(A Study on a Distribut-ed Object Oriented Software System), 産業技 術開發硏究, Vol.16 No.1, [2000]

[9] A CORBA extension for intelligent software environments /Filman, R.E. Advances in engineering software, Vol.31 No.8-9, [2000]

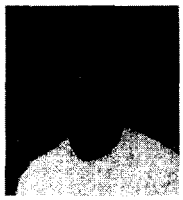
[10] Hypermedia Document Management, A

Met-adata and Meta-Information System, Suh, W. Journal of database management, Vol.12 No.2, [2001]
[11] XML(eXtensible Markup Language), W3C. http://www.w3.org/XML"
[12] Simon St. Laurent, Joe Johnston, EddDumbi-ll, "Programming Web Services with XML-R-PC, O'reilly, 2001. 6
[13] Dave Winer, UserLand SoftWare Inc, "XM-L-RPC Specification" http://www.xml-rpc.co-m/spec"
[14] XML-RPC HOWTO. http://classic.helma.at/hannes/xmlrpc
[15] W3C, SOAP version 1.2 Spec. "http://www.-w3.org/TR/2002/CR-soap12-part1-2002"
[16] Batch control meets the Internet Improvem-ents in the XML protocol for data exchange-and remote Web-based and wireless monito-ring, are enhancing batch control for the pr-ocess plant / (Chemical engineering, Vol.108-No.5, [2001])
[17] W3C, XML Protocol Abstract Model. "http-://www.w3.org/TR/2003/WD-xml-am-200302-20"
[18] W3C, Document Object Model(DOM). "http-://www.w3c.org/DOM"
[19] 김노환, 정충교 "XML DOM을 이용한 웹 문서 검색 알고리즘" VOL. 02 NO. 06 pp. 0775 ~ 0782. 2001. 06.

저자 소개

고혁준 (Hyuck-Jun Ko)

2002년 제주대학교 청정 화학 공학과 졸업
2003~현재 제주대학교 대학원 컴퓨터 공학과 석사과정
※ 관심분야 : Web Service, XML, DataBase, OS 등



김정희(Jeong-Hee Kim)

1994년 제주대학교 정보공학과 졸업(학사)
1997년 제주대학교 대학원 정보공학과 졸업(석사)
2002년 제주대학교 대학원 정보공학과 박사과정
1998~현재 제주산업정보대학 컴퓨터정보계열 겸임교수
※ 관심분야 : XML, 데이터베이스, 차세대 인터넷 기술, Semantic Web 등

곽호영(Ho-Young Kwak)

1983년 홍익대학교 전자계산학과 졸업(학사)
1985년 홍익대학교 대학원 전자계산학과 졸업(석사)
1991년 홍익대학교 대학원 전자계산학과 졸업(박사)
1990~현재 제주대학교 통신컴퓨터공학부 교수
※ 관심분야 : 객체지향 프로그래밍, 프로그래밍 언어론, GIS 등