웹 서비스의 서버 구조 단순화를 통한 웹 2.0 웹서비스 성능향상

김 용 태[†] · 박 길 철[††] · 김 석 수[††] · 이 상 호[†††]

요 약

본 논문은 웹서비스의 성능 향상을 위해 Servlet 컨테이너를 제거함으로써 전형적인 웹 서비스 구현의 필수 구성요소로 요구되는 SOAP 처리를 향상시킨 모바일 웹 서비스 구조(SMSP)를 제안한다. HTTP상에서 SOAP을 존재하게 하여 구현 실험에 의거한 지연 수행 평가를 보여주고, 웹2.0 표준 웹서비스 시스템의 성능 평가를 하였다. 본 논문에서 제안하는 모델은 웹 2.0의 웹서비스 시스템의 통신 오버헤드와 메시지를 처리하는 시간, 서버 오버헤드를 감소시키면서, 표준 웹 서비스 프로토콜을 완전하게 지원할 수 있다.

키워드 : SOAP, Web 2.0, 웹서비스, WSDL, 서블릿 컨테이너

Performance Enhancing of Web service by Simplifying Architecture in Web 2.0

Yong-Tae Kim[†] · Gil-Cheol Park[††] · SeokSoo Kim[††] · Sang-Ho Lee[†††]

ABSTRACT

In this paper, we are to overcome the latency problem of current Web 2.0 Web Services system. To overcome the standard mobile web services implementation, a SOAP message processing system (SMSP) was proposed in which the SOAP request and response messages are directly processed without using Servlet engine. Our system can completely support standard Web Services protocol, reducing communication overhead, message processing time, and server overload.

Key Words : SOAP, Web 2.0, Web service, WSDL, Servlet container

1. 서 론

정보통신 기술이 빠르게 발전하고 있으며 수많은 개념이나 용어들이 빠르게 발전하고 있다. 그 중에서도 e-business나 유비쿼터스 정보기술 환경을 논의할 때 어김없이 등장하는 것이 웹 서비스(Web service)이다[1-3].

글로벌하게 정보를 공유하고 배포하기 위한 하나의 방법으로서 그리고 하나의 거대한 자율적인 분산 콘텐츠 저장소로서 역할을 해온 웹은 점차 그 영역을 확장함으로써 더욱 복잡한 비즈니스 대 비즈니스 상호작용 등과 같은 클라이언트와 서버간의 더욱 고도화된 상호작용의 형식을 가능하게 하고 있다[2]. 기존의 분산 컴퓨팅 기술들인 CORBA, DCOM과 차이점은 느슨한 연결(loose coupling), 이진 부호화가 아닌 XML 유니코드를 사용하며, 객체 지향적이기보다는 메시지 지향적이다[11,12]. 웹 서비스는 컴퓨터와 인간의 상호작용(월드와이드웹)과는 달리 컴퓨터와 컴퓨터 상호작용을 위한 시스템이다[4-9].

이에 따라 웹 기반의 어플리케이션 대 어플리케이션 상호작용을 체계적으로 지원하는 적절한 모형과 이러한 새로운 분산 전산 플랫폼을 기존 환경들과 효율적으로 통합할 수 있는 방안에 대한 요구가 발생했다. 즉, 모든 사람들은 동일한 운영체제, 프로그램 언어, 분산 객체시스템, 데이터베이스 등을 사용하지 않는다. 웹 서비스는 이러한 요구에 대한 해답을 제공하기 위한 하나의 시도로 발생하였다[2].

웹 서비스는 기존의 웹 환경에 비해 다음과 같은 여러 이점을 가지고 있으며, 이에 따라 기존의 웹 어플리케이션을 이용한 서비스와는 분명한 차이를 보인다[6,8,11].

- 단순성 : 웹 서비스는 XML 기반이므로 기존의 분산 컴퓨팅 모델에 비해 보다 단순하고 확장이 용이한 모델을 제공한다.
- 상호 운용성 : 웹 서비스는 교환 메시지를 포함한 모든 정보 표현에 있어 XML을 이용하며, 기존의 웹 환경

※ 본 연구는 산업자원부 지역혁신센터 사업인 민군겸용 보안공학 연구센터 지원으로 수행되었음.
† 정 회 원 : 충북대학교 대학원 전자계산학과 박사과정
†† 종신회원 : 한남대학교 멀티미디어학부 교수
††† 종신회원 : 충북대학교 전기전자 컴퓨터공학부 교수
논문접수 : 2007년 3월 15일, 심사완료 : 2007년 5월 29일

위에 바로 구현할 수 있는 이점을 제공함으로써 이기종 시스템간의 상호 운용성을 극대화하였다.

- 표준 기반 : 웹 서비스는 XML 기술을 기반으로 한 개방형 표준들의 지원을 받는다.
- 빠른 발전 및 업계의 지원 : 웹 서비스는 Microsoft, IBM, Oracle을 비롯한 주요 IT 업체들의 지원을 받으며, 빠른 발전 속도와 함께 다양한 개발 도구의 지원을 받고 있다.

웹2.0의 웹 서비스는 기존의 월드와이드웹에서의 웹서비스와는 달리 다음과 같은 특징을 가지고 있다[3].

- 분산 환경이 플랫폼을 잘 지원한다.
- 참여구조로서 집단지성을 활용한다.
- 소프트웨어 배포주가 없다.
- 프로그래밍이 가볍고 단순하다.
- 다양한 디바이스로 풍부한 애플리케이션 설계를 추구한다.

앞으로의 웹 서비스 환경은 개방형 플랫폼화 되어 서비스와 기술을 접목시킬 수 있게 발전하고 사용자간의 공유를 통해 지혜를 모으고 데이터를 축적한다. 축적된 자료는 여러 종류의 디바이스로 접근할 수 있다. 따라서 웹 2.0의 출현으로 웹서비스 기술은 한층 더 발전할 전망이다. 이러한 웹2.0의 기술적 요소는 ①웹 표준(XHTML/CSS), ②브라우저 지원(Firefox, Safari), ③논리적 주소체계(Logical URI), ④콘텐트 신디케이션(RSS/Atom, RDF), ⑤오픈 API(REST, SOAP, Web Services), ⑥집단지성(Folksnomy, Tag), ⑦가벼운 서비스프레임워크(Python, Ruby on Rails), ⑧풍부한 사용자경험(Ajax, Flex), ⑨확장가능(Firefox Extensions, Widget)을 들 수 있다[4]. 본 논문은 기술적 요소들 중 오픈 API를 위한 것이다.

웹 서비스의 실행 환경은 <표 1>과 같다. 웹 서비스를 위해서는 기본적으로 웹서비스와 톰캣, AXIS가 필요하며, 웹 2.0은 톰캣 4.0 이상이 필요하고, Firefox 확장이 필요하다. Servlet 번역기인 톰캣과 SOAP 메시지를 위한 AXIS (Apache eXtensible Interaction System)가 설치되어 있으면 웹 서비스 시스템을 구축할 수 있다[16].

본 논문의 구성은 제 2장에서 관련연구에 대하여 요약/정리하고, 3장에서 본 논문이 제안하는 시스템의 구성에 대하여 설명한다. 제 4장에서는 SOAP 메시지 서비스 프로세서(SMSP)의 구현 방법과 실험 과정, 그 결과에 대하여 설명하고, 제 5장에서는 결론을 제시하고 향후 연구 방향에 대하여 기술한다.

〈표 1〉 웹 서비스의 실행환경

구 분		웹 1.0 (현재의 웹)	웹 2.0
클라이언트 사이드	브라우저	IE(Internet Explorer)	IE, Firefox or Widget
서버사이드	웹서버	Apache	Apache
	웹서비스	Tomcat, AXIS	Tomcat4.0이상, AXIS
통신(데이터전송)		HTTP	HTTP
통신주체		브라우저<->서버	브라우저(Ajax엔진)<->서버
HTML+CSS해석		서버에서수행	클라이언트에서수행
공통점		- 클라이언트와 서버로 나눔 - 웹서버 가짐 - 서버에서 애플리케이션을 가짐	

2. 관련 연구

본 논문은 아파치 웹서비스를 기반으로 연구 실험되었다. 따라서 2장에서는 아파치 웹서비스 시스템 위주로 작성되어 있다[16].

현재 웹 서비스 기술에 관련된 대표적인 프로토콜 스펙으로는 SOAP(Simple Object Access Protocol), WSDL(Web Services Description Language), UDDI(Universal Description, Discovery and Integration of Business for Web) 등으로 이루어진다. 모든 메시징에 XML이 사용되어 상호 운용성이 높다.

이외에도 비즈니스 활용을 지원하기 위한 상위표준들의 표준화 작업이 국제표준화 기구들(W3C, OASIS, WS-I 등)에서 활발하게 진행 중에 있다[5,6,7].

SOAP은 세 부분으로 구성되는 XML 기반의 프로토콜로 envelope는 메시지와 어떻게 처리할 것인가를 설명하기 위한 기본틀을 정의하고, 정의된 애플리케이션의 데이터 타입의 사례를 표현하기 위한 encoding 규칙의 집합, 그리고 떨어져 있는 프로시듀어 호출과 응답을 나타내기 위한 규정들을 정의한다[9,13].

아파치 톰캣은 servlet과 JSP를 지원하는 servlet 컨테이너이다. 수반하는 Tomcat Jasper 컴파일러는 JSP를 servlet으로 컴파일 한다. 톰캣 servlet 엔진은 Apache HTTP Server나 다른 웹 서버와 함께 결합하여 사용된다. 톰캣은 또한 독립된 웹 서버로 기능을 할 수 있다. 초기에 개발한 톰캣은 속도와 트랜잭션 조작에 대한 최소의 요구들을 가지는 다른 환경이며, 단지 적당한 개발 환경을 위해 독립적으로 존재되는 것으로 인식했다. 톰캣은 고 소통량과 고유용성 환경에 독립적인 웹 서버로 사용된다[16].

2.1 Apache SOAP

SOAP Apache SOAP는 Apache Software Foundation과 IBM alphaWorks에 의해 개발되어 SOAP를 제공한다. Apache Group은 Jakarta Tomcat 서버를 가지고 수행하는 서비스와 같은 "SOAP" 표준을 구현했다. Apache SOAP는 Apache Tomcat 응용 서버를 지원한다. 그것은 Java Servlet으로 구현되어 진다.

SOAP 메시지는 특정한 URL과 Tomcat 서버에게 보내진다. 서버에서 SOAP 서비스는 메시지를 해석하고 적당한 방법을 호출하고 다른 SOAP 메시지로 그 결과를 되돌려 준다.

SOAP 서비스의 서버 구현은 모든 자바 애플리케이션과 같다. 이것은 적당한 매개변수와 HTML 또는 XML 형식에서의 가져온 리턴을 가지고 위치를 알아내는 서비스이다. 그 차이는 표준이 되는 SOAP 메시지를 사용하는 SOAP 서

비스와 원격 호출하는 공식화된 방법을 제공한다[7,14].

클래스 연결, 쉬운 사용 그리고 실행을 위해 그것들이 제공하는 다른 여러 가지의 Web Services 구현 방법이 있다.[9] 그것들 중에서, Tomcat을 가지는 Apache Axis (Apache eXtensible Interaction System)는 인기 있는 구현 방법이다. Axis는 HTTP SOAP 요구/응답 세대, SOAP 메시지 모니터링, 동적인 요청, 웹 서비스 전개, 그리고 웹 서비스에 대해 자동적인 WSDL 생성을 제공한다[10].

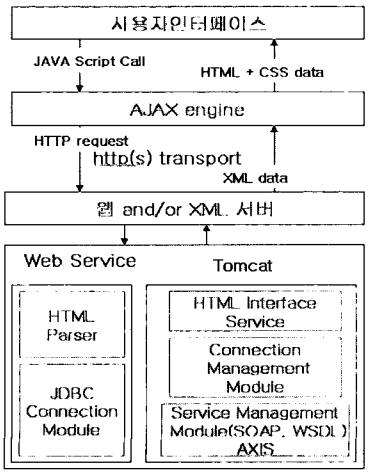
2.2 웹 2.0에서의 웹서비스 실행 흐름도

(그림 1)은 웹 2.0에서의 웹서비스 실행 흐름도이다. 웹 서비스를 위한 요구사항은 기존의 웹과 웹 2.0과 동일한 구조를 가진다. 차이점은 웹 2.0 시스템에서 Ajax 엔진이 추가로 필요하다는 점이다.

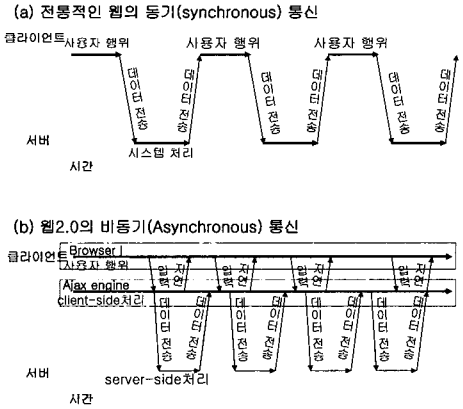
사용자 측면에서 서버에 요청할 때 전통적인 웹은 HTTP request를 하는데 웹 2.0은 JavaScript call을 하고 Ajax 엔진이 HTTP request로 변환하여 서버에 전송한다. 또한 사용자에게 전송시에도 서버는 XML data를 Ajax 엔진에 전송하고 Ajax 엔진이 HTML+CSS data로 변환하여 브라우저로 전송한다[15].

웹서비스를 위해서는 톰캣이 필요하고, Axis 라이브러리를 사용한다. 우선 HTML 메시지를 문법적으로 분석하여 WSDL로 변환하기위하여 파서가 필요하다. 하위 데이터베이스와 연결을 위한 JDBC 모듈이 있으며, 연결설정 관리 모듈 WSDL을 SOAP 메시지로 엔캡슐레이션을 하는 AXIS가 있다[15].

SOAP 메시지를 생성하는 과정에서 톰캣의 역할을 간략히 함으로서, 가벼운 시스템 구조를 가질 수 있고, 또 SOAP 및 WSDL을 생성하는 방법에 따라 시스템의 부하를 감소시킬 수 있다. 이것은 Web Servlet 엔진을 위한 부가적인 처리와 통신 포트 사용을 제거하기 때문에 효율적인 서비스 시스템을 구축할 수 있다 따라서, 우리는 Web Servlet 엔진을 사용하지 않고 SOAP 메시지를 처리할 수 있는 시스템을 제안한다. 이 구현은 3절에서 논의될 것이다.



(그림 1) AXIS와 Tomcat이 있는 웹2.0 서비스 구조



(그림 2) 전통적인 웹과 웹 2.0의 통신방식

2.3 전통적인 웹과 웹 2.0의 통신방식

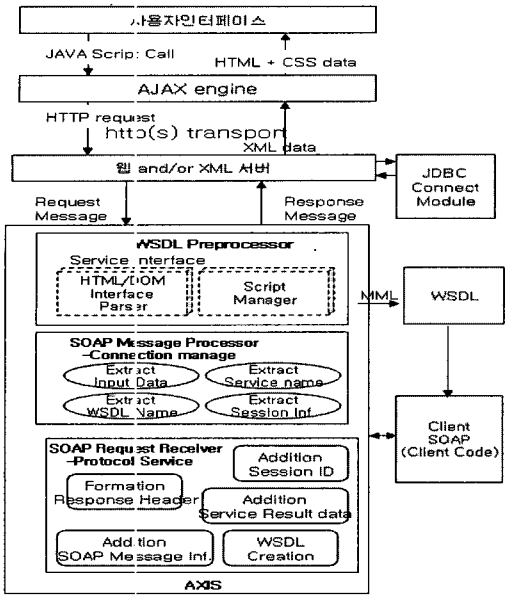
(그림 2)의 (a)는 전통적인 웹의 동기화 통신을 하며 웹 2.0은 비동기 통신을 한다.

(a)는 하나의 선으로 단절되지 않고 연결되어 있는 형태이다. 이는 사용자의 행위에 따라 데이터 전송이 발생하면 서버가 이를 받아 처리하여 클라이언트로 보내는 흐름이며, 서버의 처리가 끝나고 데이터가 도착할 때까지 사용자 브라우저는 다른 일을 하지 못하는 대기 상태에 있다. 그러나 (b)의 웹 2.0은 클라이언트의 행위가 일어나면 이벤트를 통합 제어를 담당하는 Ajax 엔진이 이벤트를 인식하여 브라우저에 출력할 사항과 서버에서 처리할 사항을 구분하여, 브라우저에 출력할 사항이면 바로 처리하고, 서버에서 처리할 사항이면 서버로 전송하여 처리한다. 사용자 이벤트가 끝나면 이벤트에 관한 것은 Ajax로 넘어감으로 서버의 처리가 완료되지 않아도 브라우저는 다른 일을 할 수 있다[16,17].

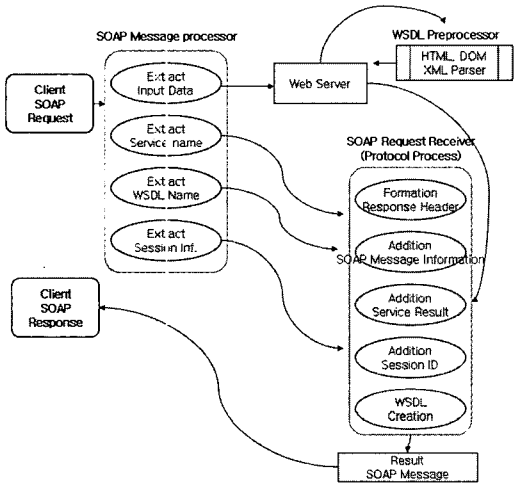
3. 웹 2.0에서 SOAP 메시지 서비스 프로세서(SMSP) 구현

SOAP 기반의 XML 메시지는 클라이언트가 서버로부터 Web Services를 요구할 때 혹은 서버가 클라이언트에게 Web Service 응답 메시지를 보낼 때 사용된다. Apache Web Services 구현은 Servlet 엔진인 톰캣과 AXIS에 의해서 지원된다. 그러나 Servlet 엔진은 추가의 통신포트를 사용하고 처리시간을 요구한다. 따라서 우리는 Servlet 엔진을 사용하지 않고 직접적으로 SOAP 요구와 응답 메시지를 처리하는 SMSP(SOAP Message Service Processor)라 명명된 SOAP 메시지를 처리하는 시스템을 설계 및 구현했다.

설계된 Web Services 시스템은 (그림 3)에서 보이고 있다. 표준 시스템(그림 1)과 구현된 시스템(그림 3) 사이의 가장 중요한 차이는 구현 시스템이 톰캣을 포함하지 않는다는 것이다. Servlet 엔진을 대체하기 위해 SMSP와 WSDL Builder가 사용되었다. 이 기능을 지원하는 Tomcat을 사용하는 대신에, WSDL 파일은 WSDL 생성기에 의해 직접적으로 생성하고 SOAP 메시지는 SMSP에서 처리된다.



(그림 3) 실험된 SOAP Message Processor(SMSP) 구현



(그림 4) SMSP의 실행 흐름도

추가되는 Web Services 모듈은 Server에 구현된다. 메시지 조작, 체인, WSDL 처리에 대한 기능, 그리고 수신된 SOAP 요구에 대한 처리는 서버 단에서 처리되어야 한다. 이를 위해 SOAP Message Processor, WSDL Preprocessor 그리고 SOAP Request Receiver(Protocol Process) 모듈이 구현되었다. 그리고 (그림 4)는 SMSP의 실행흐름도이다.

- SOAP Message Processor는 클라이언트로부터의 요청된 메시지가 표준화 된 SOAP 메시지와 웹 서버로부터 수신하는 메시지로 이루어져서 SOAP 메시지로 변해져야 하기 때문에 우리는 SOAP 메시지 분석하여 메시지를 생성한다.
- WSDL Preprocessor는 WSDL 파일을 생성시키는데 그것은 요청에 의해 활성화 된다. Java2 WSDL이 Java Class를 사용하여 WSDL 생성을 제공하지만, WSDL 생성기는 HTML/XML parser를 구현함으로서 WSDL 생성을 지원한다. 클라이언트가 HTTP를 통해 SOAP 메시지를 보냄으로써 웹 서비스를 요구한다면, 그 서버는 클라이언트 연결을 허락한다.
- SOAP Request Receiver는 SOAP 분석기, 서비스 프로세서 그리고 SOAP 생성기로 구성된다. 클라이언트들의 요구로부터 서비스 이름과 입력 데이터를 SOAP 분석기가 추출하여 서비스 프로세서 모듈에 서비스 처리를 요구한다. 그 서비스 처리 모듈에서 돌아오는 결과는 SOAP 생성기 모듈로 보내어진다. 응답 헤더와 응답 메시지 몸체(body)를 새로 만든 뒤에, SOAP 생성기 모듈은 서비스 처리 결과의 성공 여부를 검사한다. 그 결과가 성공적이면, 그 결과와 세션 ID가 더해진다. 아니면 정보는 붙여진다. 서비스 연결자는 클라이언트에게 응답 SOAP 메시지를 보냈다. 클라이언트가 SOAP 응답 메시지를 받으면, 웹 서비스 요구는 완료된다.

4. 실험 및 성능 평가

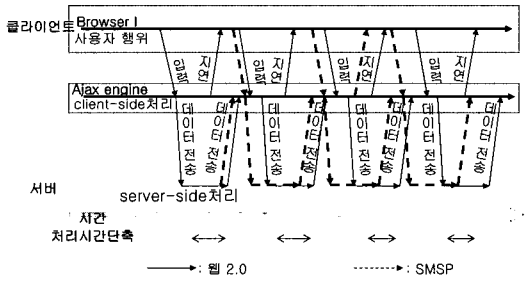
4.1 실험 환경 및 방법

본 연구의 실험은 윈도우 서버와 톰캣 5.5를 기반으로 웹서비스 시스템을 구축하여 SMSP를 기반으로 한 웹서비스와 비교하였다.

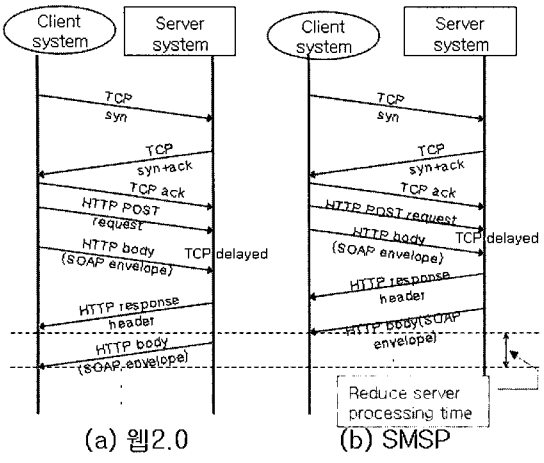
운영체제 : Windows 2000 Server
사용언어 : Java, JDK 1.4
웹서비스 : Tomcat 5.5와 AXIS or SMSP 와 AXIS
브라우저 : IE + Firefox,

SMAP의 성능을 평가하기 위해서 두개의 웹서비스 시스템이 준비 되었다. 웹 2.0 웹서비스 시스템 그리고 SMSP 웹서비스 시스템이다. 기존의 웹서비스 시스템과 웹 2.0의 서비스 통신 방식은 (그림 2)에서 설명 하였다. SMSP는 3장에서 설명했듯이 서버의 성능을 향상시킴으로서 처리 시간을 단축하여 성능 향상을 이루었다. (그림 5)는 웹 2,0과 SMSP의 처리성능을 도식화 한 것으로서 서버에서 Ajax엔진으로 전송되는 데이터 처리시간의 단축을 보인다.

(그림 6)은 아파치 웹 2.0 서버와 구현 시스템인 SMSP의 트랜잭션 그림이다. SMSP도 기본적인 통신 방법은 아파치 웹서버를 사용하였다. 아파치의 요구는 요청메세지가 2개의 패킷으로 나누어진다. 하나는 HTTP 헤더이고, 다른 하나는 HTTP body로 SOAP envelop에 포함되어 있다.



(그림 5) 웹 2.0 SMSP의 처리 시간 비교

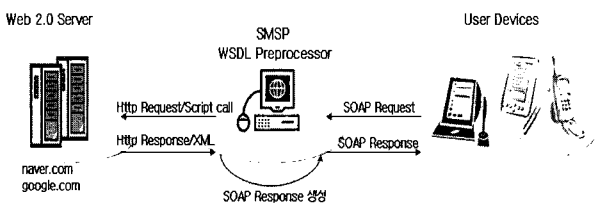


(그림 6) 웹 2.0과 SMSP의 이벤트 추적

기본적인 트랜잭션은 Apache 웹 2.0과 같은 패턴을 보인다. 차이점은 서버 사이드의 프로세싱 타임이 많이 줄었다는 점이다. 이를 위해 SOAP Message Processor, WSDL Preprocessor 그리고 SOAP Request Receiver(Protocol Process) 모듈이 구현되었다. 이 모듈은 서버에 구현되었다. 즉, 메시지 조작, 체인, WSDL 처리에 대한 기능, 그리고 수신된 SOAP 요구에 대한 처리는 Servlet 컨테이너 대신 서버에서 이루어진다.

(그림 7)은 실험 과정을 설명한다. 클라이언트가 SOAP request로 웹 서비스를 요구하는 경우에, 실험 시스템은 SOAP 메시지와 내용을 웹 서버에 HTTP call로 전달한다. 웹 2.0서버에서 받은 HTTP response를 실험 시스템 받으면, 실험 시스템은 WSDL을 생성시키고 클라이언트인 모바일 장치에게 SOAP 응답 메시지를 보낸다.

SOAP 요구는 모바일 클라이언트 시뮬레이션 프로그램에 의해 시뮬레이션 되고, 실험 시스템에 연결되고 여러 개의 SOAP 요구 메시지를 보낸다. 시간 간격은 매 1초에서 10초까지 사이에 SOAP 요구를 반복적으로 한다. 연결이 종료되면 시뮬레이션 프로그램은 실험 시스템에 연결하기 위해 반복적인 시도를 한다. 시스템에 접근하는 사용자는 동시에 200명의 사용자가 있다고 가정했다. SOAP 요구는 4개의 클라이언트 프로그램에 의해 새로 만들어졌고 각각의 프로그램은 즉시 50개 스레드들을 생성시켰다. 우리들의 실험에 콘텐트 웹 서버로서, 두 웹 사이트(www.naver.com (사전)과 www.yahoo.co.kr (주식)을 선택했다. 각각의 서버의 타임아웃은 30초이다.



(그림 7) 웹 2.0 웹서비스의 사용자 요구 처리과정 실험시스템

4.2 결과

<표 2>는 세 개의 실험 시스템에 대한 트랜잭션 지연에 대한 요약이다. 실험은 하나의 연결을 가지고 다수의 문자를 전송한다. 즉, 100개의 문자를 전송함에 있어 초기 TCP, HTTP 연결을 설정하고 SOAP 메시지로 문자를 전송한 것이다. 수신측은 네트워크 버퍼를 사용하였다. SMSP의 지연이 적은 이유는 (그림 5)에서 보였듯이 서버의 처리 시간 단축으로 인한 것이다.

〈표 2〉 무선 네트워크에 대한 처리 문자(분리된 서버와 클라이언트)

Test System	Latency(ms), 100 characters	Latency(ms), 300 characters	Latency(ms), 500 characters
Apache Web 2.0	162.9	163.0	163.1
Apache SOAP(Axis)	168.2	168.3	168.5
SMSP (웹2.0)	152.1	152.1	152.2

<표 3>에 실험에 결과를 요약하였는데, 모든 항목에서 성능 향상이 이루어졌다. 이 실험은 우리들의 모바일 웹 서비스 시스템의 실행 평가에 초점을 맞췄다.

〈표 3〉 50개 동시 요청에서 실험 결과

비교시스템	Req. Timeout	Connection refuse	Handshake error	Connection Trial
Web2.0	11.3	1.8	0.4	2.1
SMSP	0.7	0.1	0	0.2

<표 3>은 실험 시스템들의 트랜잭션 효과를 분석한 것이다. 여기서 request timeout은 요청에 대하여 타임아웃(1초)이 될 때까지 응답이 없는 것이다. connection refuses는 서버가 busy 상태이기 때문에 연결설정이 안된 개수, connection handshake error는 세션 연결 오류 횟수 그리고 connection trial은 클라이언트의 연결 시도 자체를 못한 횟수 이다.

첫 번째는 아파치의 표준 웹 서비스 시스템이며 두 번째는 본 연구의 SMSP이다. 실험은 1초에 50개의 접속시도를 하였다. 하나를 예를 들어 설명하면 실험과정에서 request timeout의 경우 10개 이하의 request의 경우에 두 시스템 모두 제한시간 내에 요구가 성공 하였으며, <표 3>에서 제시한 것처럼 50개의 경우에 SMSP는 1개미만의 발생확률을 가지고, Web 2.0은 11.3개의 발생확률을 가진다.

<표 3>의 SMSP의 결과는 Servlet 컨테이너(톰캣)을 사용하는 것보다 자체 SMSP의 효율이 높다는 것을 의미한다. 그리고 자체 성능의 향상은 시스템의 성능 향상을 가져오기 때문에 누적되는 연결 신호 처리를 향상 시킬 수 있다는 것을 의미한다. 즉 하나의 처리 시간이 적기 때문에 연속적으로 요청되는 요청을 빨리 처리할 수 있고, 이의 차이는 요청이 증가할수록 더 큰 격차를 가져오게 된다. 따라서 인터넷을 이용하여 서비스 하는 웹서비스는 시스템의 응답 속도 개선이 전체적인 성능에 많은 영향을 미친다는 것을 의미한다.

결국 연결 요청횟수가 많아질수록 오류 횟수는 기하급수적(exponential curve)으로 증가한다.

5. 결 론

웹 서비스는 인터넷 상호작용 서비스 통합 아키텍처에서 전자 상거래 등 많은 분야에서 중요한 역할을 담당하고. 점차 그 용도가 증가하는 추세이다. 특히 인터넷 웹 사이트들이 웹 2.0을 채택하는 추세이기 때문에 본 연구의 가치가 있는 것이다. 본 연구에서, 우리는 새로운 Web Service 아키텍처를 제안했다. 웹 서비스 구현에서 Tomcat Servlet 컨테이너를 제거함으로써 SOAP에 지연되어 있는 문제를 줄이는 새로운 SOAP 메시지를 처리하는 SMSP 시스템을 제안했다. SOAP 요구와 응답 메시지는 SMSP 시스템에 의해 직접적으로 처리된다. 우리는 Web Services 표준 프로토콜을 어기지 않고 개선된 모바일 웹 서비스 시스템을 구현 하였다. 실험 결과 무선네트워크에서 아파치 웹 2.0 보다 6.7%, 전통적인 아파치 웹 서비스 보다 9.7%의 성능 향상을 얻을 수 있었다. 또한 동시 요청이 50개 이상의 경우 접속성능은 약 38% 향상 되었다. 따라서 접근의 수행을 처리하는 SOAP 요구에서 표준 웹 서비스 구현보다 성능이 향상되었음을 증명한다.

향후 본 연구의 중점 사항은 I/O 시간을 현저히 줄일 수 있는 알고리즘에 관한 연구가 필요하다.

참 고 문 헌

[1] Gil cheol park and SeokSu Kim, "Design of a Intelligent SOAP Message Service Processor for Enhancing Mobile Web Service", Springer, LNAI 4252, pp.168-175, 2006.

[2] Gil cheol park, SeokSu Kim, and Wansung Jang, "An Automatic Conversion HTML/XML to WSDL for Ubiquitous Mobile Service", International Journal of Ubiquitous, Vol.1, No.1, pp.29-35, 2006.

[3] 차세대 인터넷서비스 기반으로서의 웹서비스에 대한 고찰, 안현 수, 정보관리연구, vol.33, no.1, pp. 48-60, 2002.

[4] http://blog.naver.com/ceolee79?Redirect=Log&logNo = 120034318058

[5] David Orchard, "Web Services Pitfalls,"

[6] http://www.xml.com/pub/a/2002/02/06/webservices.html, XML.com, Feb 2002.

[7] Park's Blog, http://fortytwo.co.kr/tt/

[8] 박호경, 웹 2.0 시대를 선도할 자바 웹 서비스 2.0, http://blog.naver.com/oyukihana?Redirect=Log&logNo=60034574330

[9] W3C, SOAP Version 1.2. SOAP Version 1.2, http://www.w3.org/TR/soap/

[10] Fabio Casati, Ming-Chien Shan, "Models and Languages for Describing and Discovering E-Services", Tutorial, Semantic Web Working Symposium, Stanford, USA, July 2001.

[11] 웹 서비스의 표준화 동향과 발전 방향, 이 경 하, 이 규 철, 데이터베이스연구, 제19권 제1호, 2003.3.

[12] Davis, A. and D. Zhang. A Comparative Study of DCOM and SOAP. in 4th international Symposium on Multimedia Software Engineering(MSE'02). 2002.

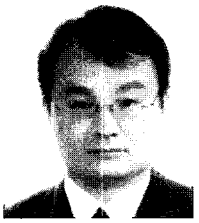
[13] Using Microsoft's Soap Toolkit for remote objectaccess, http://www.west-wind.com/presentations/soap/

[14] Davis, D. and M. Parashar. Latency Performance of SOAP Implementations. in 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. 2002.

[15] Booth, D. and C.K. Liu, Web Services Description Language (WSDL) Version 2.0, in Web Services Description Language (WSDL). 2005, W3C.

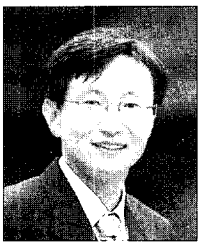
[16] The Apach Software Foundation, http://tomcat.apache.org/

[17] 김영보, Asynchronous Javascript+XML Ajax 활용, 가메출판사, 2006.



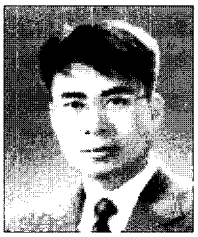
김 용 태

e-mail : ky7762@hannam.ac.kr
1984년 한남대학교 계산통계학과 학사
1988년 숭실대학교 전자계산학과 석사
1995년 충북대학교 전산학과 박사수료.
2002년~2006년 (주)가림정보기술 이사
2006년~현 재 한남대학교 멀티미디어학부 강의전담교수
관심분야 : 멀티미디어, 모바일 웹서비스, Real-time Multimedia Communication



박 길 철

e-mail : gcpark@hannam.ac.kr
1983년 한남대학교 전자계산학과 학사
1986년 숭실대학교 전자계산학과 석사
1998년 성균관대학교 전자계산학과 박사
2006년 UTAS, Australia 교환교수
1998년~현 재 한남대학교 멀티미디어학부 교수
관심분야 : multimedia and mobile communication, network security



김 석 수

e-mail : sskim@hannam.ac.kr
1991년 성균관대학교 대학원 석사
2000년 성균관대학교 대학원 박사
1998년~2000년 거창전문대학 교수
2000년~2003년 동양대학교 컴퓨터공학부 교수
2003년~현 재 한남대학교 멀티미디어학부 교수
관심분야 : 멀티미디어, 정보통신, 웹솔루션, 정보보호, 원격교육 플랫폼 및 컨텐츠

이 상 호

e-mail : shlee@chungbuk.ac.kr
1976년 숭실대학교 전자계산학과 학사
1981년 숭실대학교 전자계산학과 석사
1989년 숭실대학교 전자계산학과 박사
1981년~현 재 충북대학교 전기전자컴퓨터공학부 교수
관심분야 : 네트워크보안, Network Management, Protocol Engineering