韓國 컴퓨터情報學會 論文誌
第13卷 第3號, 2008. 5.

2008-13-3-4-2

모바일 웹 서비스를 위한 고속 메시지 처리 시스템

김 진 일*, 김 용 태**, 박 길 철***

A High-Speed Message Processing System for Mobile Web Services

Jin-Il Kim*, Yong-Tae Kim**, Gil-Cheol Park***

요 약

표준 웹 서비스나 웹 어플리케이션 서버들이 SOAP 메시지를 처리하기 위해서는 웹 서블릿 컨테이너를 필요로 한다. 그러나 이것은 추가적인 통신 포트가 필요하고 웹 서비스 모듈을 직접 구현하여 추가하는 부가적인 작업이 필요로 하는 문제점을 가지고 있다. 또한 최근의 많은 웹 애플리케이션들은 정적인 문서보다 동적인 문서 처리에 대한 요구가 점점 더 커지고 있지만 한 연구 결과에 따르면 아파치 웹 서버의 경우에 동적 문서가 많아질수록 성능이 저하될 수 있다는 것이다. 이러한 문제점을 해결하기위해서 본 논문에서는 모바일 웹 서비스를 위한 고속 메시지 처리 시스템을 설계하고 구현한다. 제안된 모바일 웹 서비스 시스템은 WSDL 변환기와 SOAP_MP로 구성되는 데 WSDL 변환기는 HTML/XML 메시지를 분석하여 WSDL 파일로 자동으로 변환함으로써 시스템 부하를 감소시키고 SOAP_MP는 웹 서블릿 컨테이너를 제거하여 SOAP 메시지를 처리 시간을 최소화한다. 제안된 시스템의 성능 평가를 위해서 표준 웹 서비스 시스템과 비교하여 실험 결과를 분석한다.

Abstract

A standard Web server or web application servers require the Web Servlet container to execute SOAP messages. But it requires additional process to make web service modules and need communication port. Also, Recently many webapplications is becoming increasingly demand against the dynamic document than the static documnet. But a recent study has found that Apache Web Server always does not show the better performance. The more it have the dynamic documents, rather it can show worse performance. To solve this problem, we propose a new High-Speed Message Processing System, in which the SOAP_MP and the WSDL builder are used. The WSDL builder convert HTML/XML to WSDL files automatically and the SOAP_MP minimize SOAP massage processing time by eliminating the Tomcat Servlet container in the mobile Web Services implementation. We compare and analyze the System, which was proposed by us, with the standard Web Service system.

▸ Keyword : 웹 서비스(Web Services), 모바일(mobile), 웹 서비스 기술 언어(WSDL), SOAP

• 제1저자 : 김진일 • 교신저자 : 김용태
• 접수일 : 2008. 3. 12, 심사일 : 2008. 4. 2, 심사완료일 : 2008. 5. 24.
* 배재대학교 교양교육 교수 **한남대학교 멀티미디어학부 강의전담교수 ***한남대학교 멀티미디어학부 교수

Ⅰ. 서 론

최근 모바일 인터넷의 급속한 성장은 인터넷 기반의 비즈니스 어플리케이션 패러다임을 빠르게 변화시키고 있다. 따라서 비즈니스 프로세스는 이러한 변화에 능동적으로 대처하는 유·무선 통합 환경을 제공해야 한다. 이러한 환경 구축을 위하여 서로 다른 시스템과 연동하는 상호 운용성이 중요하다[1]. 이를 위해 저비용으로 서로 다른 시스템간의 통합을 가능케 하는 분산 컴포넌트 기반의 웹 서비스 기술을 모바일 환경에 접목시키는 연구가 활발하게 진행되고 있다. 표준 웹 서비스 구축은 아파치 웹 서버와 함께 톰캣 AXIS를 이용하면 쉽고 빠르게 웹 서비스를 구현할 수 있는 장점이 있다. 하지만 웹 서버 외에 톰캣 서블릿 엔진을 설치하여 사용하려면 웹 서버 설치와 추가적인 인터넷 포트가 필요하고 처리 시간을 요구한다. 따라서 서버의 보안적인 측면과 관리적인 측면에서 보면 웹 서버에 부가적으로 웹 서비스 모듈을 직접 구현하여 추가하는 작업이 필요하다[2]. 또한 최근 연구에 따르면 다중 쓰레드 모델을 사용하는 아파치 웹 서버 2.0 버전의 경우에 동적 문서가 많아질수록 오히려 성능이 저하될 수 있다는 것이다[3,4]. 그러나 최근의 많은 웹 애플리케이션들은 정적인 문서보다 CGI를 통한 동적인 문서 처리에 대한 요구가 점점 더 커지고 있다. 이러한 급속도로 증가하는 수요에 따라 모바일 인터넷 서비스는 대용량 트래픽을 처리할 수 있는 확장성과 함께 더욱 빠르고 안정된 서비스를 지원하기 위한 시스템이 필요하다.

따라서 본 논문에서는 이러한 문제와 요구사항을 해결하기 위하여 모바일 웹 서비스를 위한 고속 메시지 처리 시스템을 제안한다. 제안된 모바일 웹 서비스 시스템은 WSDL 변환기와 SOAP_MP로 구성되는 데 WSDL 변환기는 HTML/XML 메시지를 문법적으로 분석하여 WSDL 파일로 자동으로 변환함으로써 시스템의 부하를 감소시키고 SOAP_MP는 웹 서블릿 컨테이너를 제거하고 SOAP 메시지를 처리 시간을 최소화한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 웹 서비스 및 모바일 서비스에 관련된 기술을 요약 ·정리하고 3장에서는 HTML/XML 파일을 WSDL 파일로 자동으로 변환하는 변환기와 웹 서블릿 엔진을 사용하지 않고 SOAP_MP로 구성된 고속 메시지 처리 시스템의 구현 방법을 기술하고 4장에서는 실험 및 성능평가에 대하여 기술하고, 마지막으로 5장에서는 결론과 향후 연구 방향에 대하여 기술한다.

Ⅱ. 관련연구

2.1 모바일 웹 서비스

현재 웹 서비스 기술은 그림 1에서 처럼 웹 서비스에 대한 요청과 응답에 사용되는 메시지 형식을 정의하고 웹 서비스 사용을 위한 메시지 전송 프로토콜인 SOAP(Simple Object Access Protocol), 웹 서비스의 위치, 전송할 SOAP 메시지의 형태 등 웹 서비스의 인터페이스를 기술하는 WSDL(Web Services Description Language), 웹 서비스를 게시하고 찾는 절차를 규정한 UDDI(Universal Description, Discovery, and Integration)라는 XML 기반의 세 가지 대표적인 표준을 이용한다[5]. XML 메시지를 웹을 통해 전송함으로써 서로 다른 컴퓨팅 환경에서 사용되는 모든 애플리케이션들이 직접 소통하고 실행하는 동적 소프트웨어 시스템으로 모바일 웹 서비스로 영역이 확장되어 WAP/WML 등의 기반 기술과 연동된다.

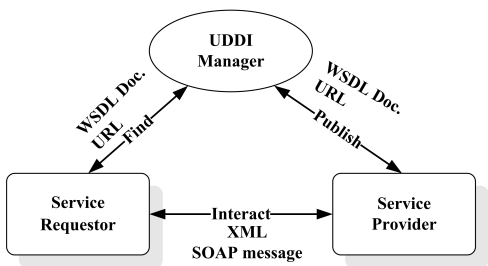

그림 1. 웹 서비스의 구조
Fig. 1. Web Service Architecture

그림 2에서 나타나는 것처럼, 무선 인터넷 프로토콜 WAP은 무선망과 웹의 연동을 위하여 클라이언트와 웹 서버 간에 WAP 프럭시(Proxy)라는 WAP 게이트웨이(Gateway)가 존재한다는 점에서 큰 차이가 있다. WAP 단말기는 WAP 프럭시와 연결되고, WAP 프럭시는 일반 웹 서버로 연결한다[6].

WAP 단말기의 웹 연결 요청은 WAP 프로토콜을 사용하여 메시지를 압축한 다음, WAP 프럭시로 전송한다. 그러면 WAP 프럭시는 메시지를 디코딩하고 URL을 사용하여 해당 웹 서버로 요청한다. WAP 프럭시는 WAP 기반 무선

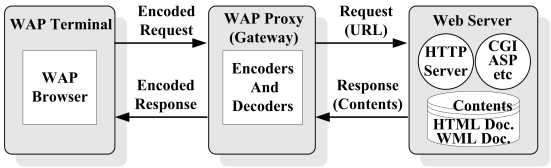

그림 2. 무선 인터넷 연결 구조
Fig. 2. Structure of wireless Internet

인터넷의 핵심 요소로써 WAP 프로토콜과 인터넷의 HTTP 프로토콜을 변환하는 역할을 수행한다. 즉, 프럭시의 기능은 프로토콜 변환이 대표적인 것이다[7]. 웹 서버는 WML과 WML 스크립트의 형태로 콘텐츠를 전송하고 다시 WAP 프럭시는 WAP 프로토콜에 맞게 메시지를 변환하여 WAP 단말기로 콘텐츠를 인코딩하여 전송한다. 이처럼 자원이 한정적인 모바일 장치들을 사용하는 경우 전체 웹 서비스 성능을 저하시키는 문제점이 발생한다.

모바일 인터넷은 유선 인터넷과 달리 기술 환경과 특성상 여러 가지 제약점들을 가지고 있기 때문에 WAP 프럭시 는 캐싱, 압축, 대용량 트래픽에 대한 확장성을 고려해야 한다. 하지만 기존 WAP 프럭시들은 기본적으로 캐싱과 압축의 기능을 제공하지만 확장성 관점에서는 대부분의 경우에 고려하지 않고 몇몇 연구[8,9]에서만 고려하고 있다.

따라서 본 연구에서는 급속도록 증가하는 무선 인터넷 서비스의 수요에 따라 대용량 트랙픽을 처리할 수 있는 시스템이 요구되고 있기 때문에 웹 서비스의 성능 향상을 위해 고속 메시지 처리 시스템을 제안하고자 한다.

2.2 아파치와 톰캣

웹 서비스를 구축하기 위해서는 먼저 아파치 웹 서버, IIS 웹 서버 등과 같은 웹 서버와 함께 JSP(Java Server Pages)를 지원하는 톰캣과 같은 JSP 컨테이너가 필요로 하는 데, 아파치/톰캣 기반의 표준 웹 서비스 시스템의 구조는 그림 3과 같다[2]. 아파치 웹 서버는 NCSA httpd 1.3을 기반으로 개발된 웹 서버이며 Apache HTTP Server Project에서 지속적으로 업그레이드 버전을 발표하여, 현재 발표된 아파치의 최신 버전은 2.0.61이다[1]. 아파치 웹 서버는 1.3 버전은 다중 프로세스를 사용하였지만 2.0 버전은 다중 쓰레드를 사용한다. 다중-프로세스는 사용자 수가 증가하면 프로세스의 생성과 프로세스간의 문맥 전환을 위한 오버헤드가 크다. 이를 해결하기 위한 방법으로 다중 쓰레드를 활용한다. 쓰레드는 한 프로세스의 주소 공간을 쓰레드들 간에 공유하여 쓰레드의 생성 및 문맥 전환의 부담을 최소화하는 장점을 가진다.

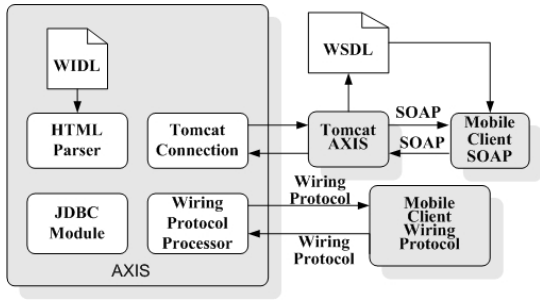

그림 3. 표준 웹 서비스 구조
Fig. 3 Structure of Standard Web Service System

일반적으로 톰캣 서블릿 엔진은 웹 서버 기능이 포함되어 있다. 그럼에도 불구하고 아파치 서버나 다른 웹 서버와 연동하여 사용하는 데 그 이유는 다음과 같다. 첫째, 톰캣은 웹 서버 만큼 정적인 문서를 빠르게 처리하지 못하기때문에 일반적인 웹 요청은 웹 서버가 처리하고 JSP 문서나 서블릿 등 JAVA와 관련된 요청을 톰캣이 수행함으로서 톰캣의 작업부하를 줄일 수 있다. 둘째, 톰캣은 아파치 웹 서버만큼 다양한 옵션 및 환경 설정 기능을 제공하지 못할 뿐 아니라 아파치 웹 서버만큼 안정적이지 못하다. 마지막으로 이미 많은 사이트들이 웹 서버 기반으로 시스템을 개발하였다. 이러한 이유들 때문에 웹 서버와 톰캣을 연동해서 사용한다. 아파치/톰캣 구성에서 정적 콘텐츠는 아파치상에 배치되어 일반적으로 보안에 제외되어 클라이언트의 접속을 처리하게 된다. 이 구조는 네트워크의 내부에 있는 서버로 부터, 네트워크 외부의 클라이언트에 대해 콘텐츠 전달을 제어하는 상황에서 톰캣은 부하도 많이 걸리고 대용량의 파일을 전송하는데도 문제점을 가지고 있다.

아파치 SOAP은 자바 서블릿으로 구현되며 아파치/톰캣 응용 서버를 지원한다. SOAP 메시지는 특정한 URL과 함께 톰캣 서버에 전송된다. 서버에서 SOAP 서비스는 메시지를 해석하고 이에 대한 적절한 방법을 호출하여 다른 SOAP 메시지를 사용하여 결과를 되돌려 준다. SOAP 서비스의 서버 구현은 적당한 매개 변수와 HTML 또는 XML 형식에서 가져온 리턴을 가지고 위치를 알아내는 서비스이다. 그 차이는 표준이 되는 SOAP 메시지를 이용하는 SOAP 서비스와 원격 호출하는 공식화된 방법을 제공한다[10,11].

SOAP 메시지를 생성하는 과정에서 톰캣의 역할을 간략히 함으로써 가벼운 시스템 구조를 가질 수 있고 또 SOAP 및

WSDL을 생성하는 방법에 따라 서버의 부하를 감소시킬 수 있다. 이것은 웹 서블릿 엔진을 위한 부가적인 처리와 통신 포트를 제거하기 때문에 효율적인 웹 서비스 시스템 구축이 가능하다.

따라서 본 논문에서는 웹 서비스의 성능 향상을 위해 HTML/XML 파일을 WSDL 파일로 자동으로 생성하기 위한 변환기와 웹 서블릿 엔진을 사용하지 않고 SOAP 메시지를 처리하는 SOAP_MP로 구성된 고속 메시지 처리 시스템을 구현한다.

III. 제안된 고속메시지 처리 시스템

3.1 HTML/WSDL 변환기

무선 웹 서비스 통신을 위해서는 WSDL 언어가 필요하고, 기존의 웹에 존재하는 HTML 데이터를 무선 통신 데이터 형식인 WSDL 데이터 형식으로 변환하기 위하여 HTML/WSDL 변환기를 설계한다. 이 변환기는 세 개의 하위 모듈, 규칙 스크립트(rule script), 스크립트 엔진(script engine), 마크업 언어 변환기(markup language converter)로 구성한다.

규칙 스크립트는 내용을 재구성하기 위한 규칙을 저장하고, 관리 프로그램을 이용하여 사용자가 생성한다. 저장된 규칙들은 이동 단말기의 구조적인 정보를 표시하고 개별화 정보를 포함한다.

스크립트 엔진은 스크립트 규칙과 클라이언트 정보를 사용하여 콘텐츠를 재구성한다. 마크업 언어 변환기는 서버가 제공한 마크업 언어가 클라이언트가 처리 가능한 언어와 다른 경우에 마크업 언어를 변환한다.

규칙 스크립트는 다음과 같이 생성한다. 만약 웹 사이트 주소가 제공된다면 제안한 시스템은 웹 사이트의 정보를 읽고 분석한다. 다음으로 분석된 정보는 Java 1.4의 org.w3c.dom 라이브러리를 이용하여 AXIS에 독립적으로 실행하며, DOM 트리를 사용해서 나타낸다. 사용자는 무선 인터넷 콘텐츠로 제공하는 노드 정보를 선택하고 저장한다. JML(Java Mark-up Language) 에디터는 XML 태그들과 저장된 아이템들의 특성으로 정의한다. TITLE, BASEURL, LINK, HREF, CONTENT 그리고 ELEMENT는 XML 태그의 예이고 이런 많은 특성들에 의하여 모바일 콘텐츠를 요구에 알맞게 생성한다.

그림 4는 클라이언트가 서버에 WSDL을 요청하는 과정은 다음과 같다. 클라이언트가 HTTP를 통해서 웹 서버에 접속한다. 그리고 클라이언트의 요청이 WSDL 요청으로 확인되면 아 요청을 분석한 다음, HTML/XML 파서를 이용하여 WSDL을 생성한다.

```
Procedure Client()
{
  call Server(SOAP_Message);
  Wait for return of transaction result from server;
  the receipt of a transmission result;
  transmission WSDL request;
}
Procedure Server()
{  Procedure Connection()
   {
      connection acceptance;
      analysis of request a kind;
      If (kind of request = request WSDL) then
      {
        call WSDL_Creation()
        Wait for return result from WSDL_Creation;
        the receipt of a result;
        transmission to Client transaction result;
      }
   }
   Procedure WSDL_Creation()
   {
     analysis request;
     an extract of WSDL name;
     creation of response header information;
     an extract of WSDL information;
     creation of a XML document information;
     addition of WSDL information
     addition of ComplexType;
     creation of a WSDL;
     creation of a result;
     call Connection()
     return of result to Connection;
   }
}
```

그림 4. 클라이언트의 WSDL 요청 과정
Fig 4. WSDL request process of client

HTTP를 사용하는 클라이언트의 XML 웹서비스 메시지에 대한 WSDL 정보를 생성하는 과정은 다음과 같다.

- 응답 헤더 정보를 생성(new HttpHeader())
- WSDL 정보 획득(ScriptManager.getScript())
- 새로운 XML 문서 정보 생성 (DocumentBuilder.newDocument())
- WSDL에 필요한 엘리먼트 생성 (Document.createElement())
- WSDL 모든 서비스, 바인딩에 필요한 엘리먼트 추가 (WSDLInfo.addOperation())
- 바인딩의 VARIABLE에 필요한 complexType 추가 (WSDLInfo.addComplexType())

■ 생성한 XML 문서에 element를 추가 (Document.appendChild())

WSDL을 생성하고 명시된 정보에 의해 WSDL이 요구하는 정보를 서버로부터 획득하여 클라이언트에게 생성된 결과를 전송한 후 종료한다.

3.2 SOAP 메시지 프로세서

웹 서비스에서 SOAP 기반의 XML 메시지는 클라이언트가 서버로부터 웹 서비스를 요구할 때나 서버가 클라이언트에게 웹 서비스 응답 메시지를 보낼 때 사용한다. 표준으로 설정하는 웹 서비스 구현은 아파치 톰캣과 AXIS로 구성된다. 그러나 서블릿 엔진은 추가의 통신 포트를 사용하고 처리하는 시간을 요구한다. 따라서 본 논문에서는 서블릿 엔진을 사용하지 않고 직접 SOAP 요구와 응답 메시지를 처리하는 SOAP 메시지 프로세서(SOAP_MP)을 제안한다. 그림 5는 SOAP_MP와 WSDL 변환기를 사용하여 웹 서비스 시스템을 구현한 것이다. 기존의 시스템과 제안된 시스템 사이의 가장 중요한 차이점은 톰캣의 사용 여부이다. 본 논문에서 제안한 시스템에서는 톰캣을 제거하고 대신에 WSDL 파일은 WSDL 변환기에 의해 직접 생성하고 SOAP 메시지는 SOAP_MP에서 처리한다는 점이다.

3.2.1 SOAP 요구 메시지 분석

클라이언트가 접근하는 URL은 WSDL을 통해 획득하며, "http://host주소:port/webservice/WSDL명" 형태이다. 클라이언트가 SOAP 형태의 XML 문서로 HTTP를 통해서 웹 서비스를 요청하기 위해 요청 서비스명, 입력값, 세션아이디가 포함된 SOAP 메시지를 서버로 전달하면 서버는 클라이언트의 접속을 수락하고 클라이언트의 요청 형태를 분석한다.

클라이언트의 요청 형태가 웹 서비스인 경우는 SOAP 메시지 분석부로 SOAP 메시지를 전달하고, SOAP 메시지 분석부는 클라이언트의 요청을 분석하여 클라이언트가 요청한 서비스의 WSDL명을 추출한 후 전달된 SOAP 메시지를 분석하여 정보를 생성한다. 분석한 정보에서 서비스명과 입력값들을 추출한 후 서비스 처리부에 서비스 처리를 요청한다.

3.2.2 SOAP 응답 메시지 생성

서비스 처리부에서 처리한 결과는 SOAP 메시지 생성부로 전달되어 응답 SOAP 메시지를 생성하기 위하여 응답 헤더 정보와 응답 SOAP 메시지에 대한 정보를 생성하고, 서비스 처리 결과가 정상인지 확인하여, 정상이면 서비스 결과 값과 세션 아이디를 추가하고, 정상이 아니면 실패 정보를 추가하고, SOAP 메시지 정보에서 SOAP 메시지를 생성하여 접속부로 전송하면, 접속부는 다시 클라이언트로 전달한다. 그림 6은 SOAP_MP 시스템의 실행 흐름도를 나타낸다.

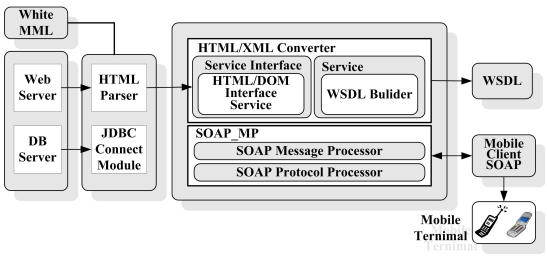

그림 5. SOAP_MP와 WSDL 변환기를 이용한 모바일 웹 서비스
Fig 5. Mobile Web Service using the SOAP_MP and the WSDL Builder

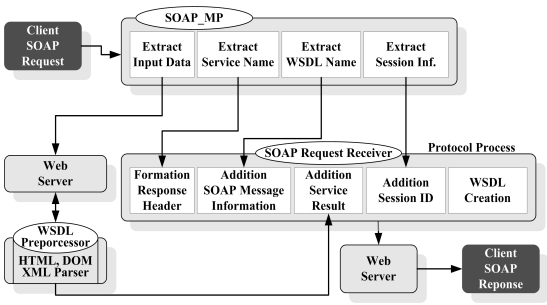

그림 6. SOAP_MP 시스템의 실행 흐름도
Fig 6. Execution flow of SOAP_MP system

IV. 실험 및 성능 평가

4.1 실험방법

실험은 제안된 모바일 웹 서비스 시스템의 성능 평가를 위해서 표준 웹 서비스 구조에서 구현된 시스템과 비교한다. 표준 시스템이 AXIS를 가진 톰캣 서블릿 컨테이너를 요구하는 반면에 본 논문에서 제안된 시스템은 서블릿 컨테이너를 제거하고 SOAP 메시지를 처리하며 WSDL은 WSDL 변환기로 만들어진다. 두 시스템의 성능을 비교하기 위한 과정은 그림 7과 같다.

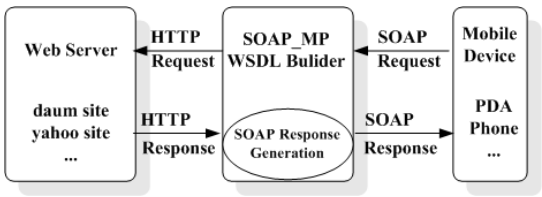

그림 7. 실험 과정
Fig 7. Experiment Process

만약 클라이언트가 SOAP 요구로 웹 서비스를 요구하는 경우 실험 시스템은 SOAP 메시지를 분석하고 콘텐츠 웹 서버에 HTTP 요구를 전송한다. 만약 실험 시스템이 웹 서버로부터 HTTP 응답 메시지를 수신하는 경우 실험 시스템은 WSDL를 생성하고 클라이언트 모바일 장치에 SOAP 응답 메시지를 전송한다.

SOAP_MP는 3장에서 설명한 것과 같이 서버 성능을 향상시킴으로써 처리 시간을 단축하여 성능 향상을 이루었다. 각각의 시스템을 구현하기 위해 사용된 구성 요소는 다음과 같다.

- 운영체제 : Windows 2000 서버
- 사용언어 : Java, JDK 1.4
- 웹서비스 : Tomcat 5.5와 AXIS
  or SOAP_MP와 AXIS
- 브라우저 : IE + Firefox

실험을 위해 SOAP 요구는 실험 시스템에 연결된 모바일 클라이언트 시뮬레이션 프로그램을 사용하여 SOAP 요구를 반복적으로 요청한다. 연결이 종료되면 시뮬레이션 프로그램은 실험 시스템에 반복적으로 연결을 시도한다. 시스템에 접근하는 동시 사용자 수는 500명으로 가정한다. SOAP 요구는 5개의 클라이언트 프로그램에 의해 생성되고 각 프로그램은 동시에 100개의 쓰레드를 생성시킨다. 실험에 사용한 콘텐츠 서버로서는 다음 사이트의 "사전" 과 야후사이트의 "주식" 으로 한다. 각각의 서버의 타임아웃은 10초이다. 두 시스템을 비교하기 위하여 다음과 같은 정보를 수집하였다.

- 실험 시간(Test Time) : 실험 진행 시간
- 총 요구 횟수(Total Request) : 실험 시간 동안에 발생한 총 요구 횟수
- 접속 실패 횟수 - 서버 busy(Connection Refuse) : 서버의 busy 상태로 인해 접속에 실패한 횟수
- 세션 오류 횟수(Connection Handshake Error) : 세션 연결 오류 횟수
- 접속 실패 횟수 - 연결실패(Connection Trials) : 서버 접속에 실패한 클라이언트 수
- 접속 제한시간 초과 횟수(Request Timeout) : 요청에 대하여 주어진 시간을 초과한 횟수

4.2 실험 결과

실험 결과를 요약하면 표 1과 같다. 제안된 시스템은 비교한 모든 항목에 대해서 우수한 결과가 나타냈다. 비록 제안된 시스템의 실험 시간이 표준 시스템의 실험 시간보다 짧았지만 전체 요청 수는 표준 시스템 보다 많았고 접속 제한 시간 초과 횟수는 표준 시스템보다 적었다. 예를 들면 제안된 시스템에서의 초당 평균 요구 횟수는 17.94인 반면에 표준 시스템에서는 9.64이다. 또한, 표준 시스템에서는 많은 연결 에러가 발생한다.

표 1. 실험 결과
Table 1. Experiment Results

실험 정보	표준 시스템	제안된 시스템
실험 시간	46,200	29,400
총 요구 횟수	445,422	524,573
접속 실패 횟수(서버 busy)	18,960	0
세션 오류 횟수	513	0
접속 실패 횟수(연결 실패)	22,534	243
접속 제한시간 초과 횟수	119,891	756

이러한 결과는 서블릿 컨테이너를 사용하는 것보다 본 논문에서 제안한 시스템의 효율이 높다는 것을 의미한다. 그리고 웹 서비스의 성능 향상은 전체 시스템의 성능 향상을 가져와 누적되는 연결 신호 처리의 향상을 의미한다. 결국 연결 요청 횟수가 많아질수록 기존의 웹 서비스 시스템의 오류 횟수는 기하급수적으로 증가한다. 좀 더 자세히 실험 요인(factor)별로 비교해 보면, 그림 8에서 그림 11과 같다.

그림 8의 접속시간 초과 횟수(request timeout)을 비교해 보면 접속 시도가 적은 경우에는 두 시스템 모두 제한 시간 내에 요구가 성공하였다. 그러나 접속 시도가 50인 경우에는 표준 시스템이 12.1개의 발생확률을 가지는 반면에 제안

된 시스템은 1개미만의 발생 확률을 가지는 것으로 나타났다. 접속 시도 횟수가 증가할 수록 표준 시스템이 접속시간 초과 횟수가 기하급수적으로 증가함을 볼 수 있다. 그림 9, 그림 11의 서버의 Busy 상태로 인한 접속 실패 횟수(Connection refuse)와 연결실패로 인한 접속실패횟수(Connection Trials)의 경우도 비슷한 결과를 얻을 수 있었다. 그림 10의 세션 오류(Connection Handshake Error)의 경우 제안된 시스템은 접속 시도가 100미만인 경우에는 에러가 발생하지 않았지만 표준 시스템은 10미만인 경우에는 속 시도가 50, 100인 경우에 각각 0.4개, 11개의 발생 확률을 가진다.

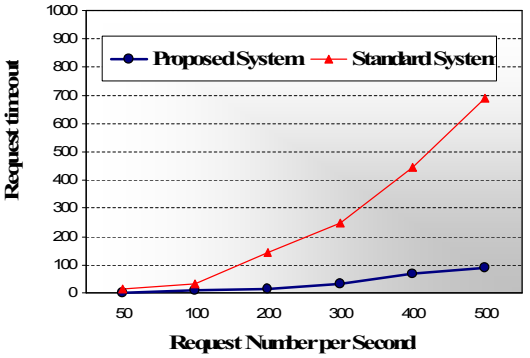

그림 8. 접속시간 초과 횟수
Fig. 8. Request Timeout

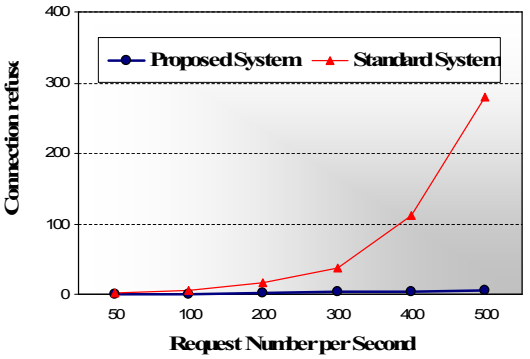

그림 9. 접속 실패 횟수(Server Busy)
Fig 9. Connection Refuse

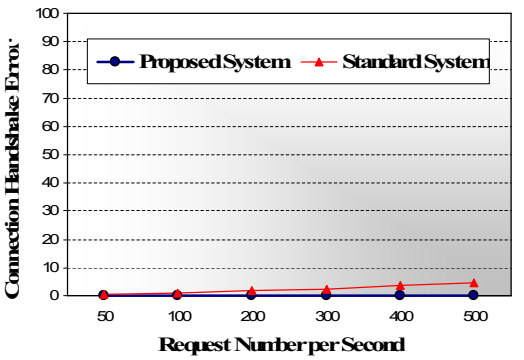

그림 10. 세션 오류 횟수
Fig 10. Connection Handshake Error

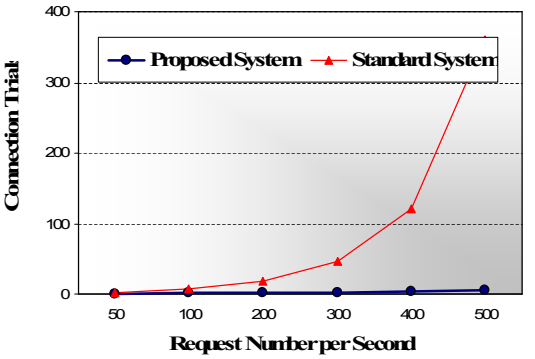

그림 11. 접속실패 횟수(연결실패)
Fig 11. Connection Trials

V. 결론

표준 웹 서비스를 구축하기 위해서는 아파치 웹 서버와 톰캣을 연동하여 사용한다. 아파치 웹 서버 2.0 버전에서 뿐만 아니라 많은 웹 서버 혹은 웹 애플리케이션 서버들이 다중 쓰레드 모델을 적용하고 있는 추세이다. 그러나 동적 문서의 비중이 높을수록 다중 쓰레드 모델을 사용하는 것이 항상 좋은 성능을 발휘하는 않는다는 연구 결과가 있다. 그러나 최근의 많은 웹 애플리케이션들은 정적인 문서보다 CGI를 통한 동적인 문서를 점점 더 많이 처리해야하는 요구가 커지고 있다. 이러한 요구에 효율적으로 대처하기 위한 웹 서버는 더 높은 확장성과 함께 사용자 환경 변화에 능동적으로 대처하여 더욱 빠르고 안정된 모바일 인터넷 서비스를 제공할 수 있도록 고

속 메시지 처리 시스템이 필요하다.

따라서 본 논문에서는 이러한 요구에 부응하는 고속 메시지 처리 시스템을 설계하고 구현하였다. 제안된 모바일 웹 서비스 시스템은 WSDL 변환기와 SOAP_MP로 구성되는 데 HTML/XML 메시지를 문법적으로 분석하여 WSDL 파일로 자동으로 변환하는 WSDL 변환기는 현재 대부분의 웹 서버들이 HTML 기반 콘텐츠를 재사용하도록 제공할 수 있다. 그리고, SOAP_MP는 웹 서블릿 컨테이너를 제거하고 SOAP 메시지를 처리 시간을 최소화한다. 그러므로 제안된 시스템은 유·무선 인터넷 서비스의 일관성을 제공하고 유지· 개발 비용을 최소화하기 위한 시스템 구축에 유용하게 사용될 수 있다.

향후에는 초고속 네트워크로 연결된 컴퓨터의 유휴자원을 공유하여 특정 작업에 집중시킴으로써 작업 속도를 무한정 향상시킬 수 있는 그리드 컴퓨팅 네트워크를 기반으로 하는 고속 모바일 웹 서비스 처리 시스템의 개발이 필요하다.

참고문헌

〔1〕 장덕성, 모바일 솔루션 기업의 성공적 비즈니스 모형, 한국컴퓨터정보학회 논문지, 제10권 3호, pp.275- 286, 2005.

〔2〕 The Apache Software Foundation. http://www.apache.org, 1999-2007.

〔3〕 Mi-ryeong Yeom, Kihun Chong, Sam H. Noh, An Assessment of the Apache Web Server 2.0 Performance on Linux, In Proceedings of 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2002) July 14-18, 2002.

〔4〕 전홍석, 이승원, 강현규, 아파치 웹 서버에서의 다중 쓰레드 풀 활용 기법 분석, 정보과학회논문지 : 시스템 및 이론, 제 32 권 제 1 호, pp.21-28, 2005.

〔5〕 최유순, 박종구, 웹서비스를 위한 WSDL 리포지토리 설계, 한국해양정보통신학회논문지, 제11권 제4호, pp.745-753, 2007.

〔6〕 OMA Technical Section, http://www. openmobile alliance.org /tech /affiliates /wap /wapindex.html

〔7〕 김인희, 모바일 분산 인터미디어리 기반 웹서비스를 위한 클라이언트/서버 메시지 처리 모델 설계, 서울여자대학교, 석사학위, 2005.

〔8〕 K. Kim, H. Lee and K. Chung, A Distributed Proxy Server System for Mobile Web Service, Proceedings of the 15th International Conference on Information Networking, IEEE, pp. 8A 749-754, 2001.

〔9〕 Anindya Datta and et al., Proxy-based acceleration of dynamically generated content on the World Wide Web: an approach and implementation, Proceedings of the 2002 ACM SIGMOD international conference on management data, 2002.

〔10〕 Park's Blog, http://fortytwo.co.kr/tt/

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

저 자 소 개

김 진 일

2000년 한남대 컴퓨터공학과 박사
2006년 - 현재, 배재대 교양교육지원센터 IT 분야 교수
〈관심분야〉 교육 공학, 모바일 웹 서비스, 네트워크, 퍼지이론,

김 용 태

2002년-2006년 가림정보기술 이사
2008.2. 충북대학교 전산학과 박사
2006.3-현재 한남대학교 멀티미디어학부 강의전담교수
〈관심분야〉 모바일 웹서비스, 정보보안, AAA, 모바일 통신보안, 멀티미디어

박 길 철

1998년 성균관대학교 전산학과(박사)
2006년 UTAS, Australia 교환교수
1998년 - 현재 한남대학교 멀티미디어학부 교수
〈관심분야〉 multimedia and mobile communication, network security