모바일 웹서비스의 성능 향상을 위한 인터미디어리 기반의 프레임워크 설계

김용태*, 정윤수**, 박길철***

A Design of Intermediary based Framework for Improving Mobile Web-Service Performance

Yong-Tae Kim*, Yoon-Su Jeong**, Gil-Cheol Park***

※ 본 연구는 지식경제부 지역혁신센터 사업인 민군겸용 보안공학 연구센터 지원으로 수행되었음

요 약

최근 많은 모바일 디바이스 환경에서 웹서비스 이용에 대한 요구가 증가하고 있지만 현실의 여러 가지 한계와 문제점들로 인해 무선 네트워크에서의 웹 브라우징은 많은 제약이 발생한다. 최근의 급속하게 변화하는 웹 환경에서 이전의 웹서버 성능으로는 다양한 요구 사항을 만족할 수 없기 때문에, 모바일 웹서비스 엔진의 성능 향상이 필요하다. 따라서 본 논문에서는 인터미디어리 기술을 모바일 웹 서비스에 도입하여 웹서비스 이용자를 위한 서비스 환경을 개선한다. 본 논문에서는 모바일 웹서비스의 성능 향상과 모바일 디바이스와 웹서버의 상호운용을 위한 인터미디어리 기반 웹서비스 프레임워크를 설계 구현한다.

ABSTRACT

Nowadays, in many mobile device environment, while the demand for using web service is increasing, there are many limitations on web-browsing in wireless network because of various restrictions and problems in reality. We need improvement in function of mobile web service engine because we can't be satisfied with the multiple demands by using previous capacity of web server in rapidly changing web environment. Therefore, this paper improves the service environment for web service users by introducing intermediary technology to web service. This paper embodies intermediary based web service framework for improvement of the function of mobile web service and for the interoperability of mobile device and web server.

키워드

Web Services, Mobile Device, WAP, XML, WML, Intermediary

* 한남대학교 멀티미디어학부 강의전담 교수
** 충북대학교 전자계산학과(교신저자)
*** 한남대학교 멀티미디어학부 교수

접수일자 2008. 06. 11

Ⅰ. 서 론

인터넷 환경과 IT 분야의 정보 통신 기술에 대한 패러다임 변화의 중심에는 XML 기반의 웹서비스가 존재한다. 웹 기반의 웹서비스 역시 비즈니스 응용의 영향으로 변화한다. 웹서비스의 가능성은 웹사이트보다는 다양한 네트워크 제품의 결합과 상호운용성에 의해 부가 가치를 가진 새로운 형태의 웹서비스를 창출한다[1]. 즉 다양한 인터넷 단말기, 가전, 무선 단말기 등의 결합으로 강력한 비즈니스를 창출한다[2].

현재 대부분의 웹서비스는 PC 기반으로 인한 문제점으로 휴대폰, PDA 등과 같은 다양한 모바일 디바이스까지 웹서비스 제공의 필요성이 대두되었다. 그러나 기존의 WAP 모바일 인터넷 서비스는 불필요한 웹 정보까지 수신되어 전송의 효율성, 경제성에서 불이익이 발생한다. 최근에는 유선 통신 시스템과 무선의 일부 구성 요소의 공유에 의해 서로 공유하면서 새로운 이익의 창출이 가능하게 되었다.

웹서비스의 근본적인 이슈의 대부분은 아파치와 톰캣 기반이며, 플랫폼 독립적인 통합과 자원 공유에 기인한다. 그러나 톰캣 서블릿 엔진의 사용은 웹 서버 설치, 인터넷 포트의 추가 필요, 처리 시간 요구 그리고 서버의 보안과 관리에 추가 요소가 발생한다. 또한 클라이언트 요청 처리를 위한 쓰레드(thread)의 실시간 생성에 의한 단점이 발생한다.

따라서 본 논문에서는 향상된 모바일 웹서비스 제공을 위해 서블릿 엔진과 무관하게 직접 SOAP 요구와 응답 메시지를 처리하여 성능 향상을 모색하고, 모바일 장치의 성능 제약 사항을 극복하고 기존의 웹서비스와 상호운용을 위하여 WAP 게이트웨이를 개선하는 인터미디어리(Intermediary) 기반의 모바일 웹 서버를 설계 구현한다.

본 논문의 2장은 관련 연구에 대하여 기술하고, 3장에서는 본 논문에서 제안하는 모바일 웹서비스 시스템의 프레임워크 설계에 대하여 기술한다. 4장은 제안 시스템의 실험 결과와 평가를 나타낸다. 마지막으로 5장에서 결론과 추후 작업의 내용을 기술한다.

Ⅱ. 관련 연구

2.1 웹 서비스의 개요

웹 서비스는 기존의 인터넷 기반(HTTP 프로토콜)에서 자료 교환을 위한 모듈화된 컴포넌트이다. 플랫폼 독립적으로 다양한 하드웨어와 소프트웨어의 연계, JIT (Just-In-Time) 통합과 자원 공유를 위한 상호운용성을 향상시키며, 방화벽과 무관한 개방형 표준 데이터 표현 기법인 XML (eXtensible Markup Language)과 인터넷 프로토콜을 결합한 새로운 패러다임의 분산 컴퓨팅 기술로 인터넷 접근이 용이한 기술이다.

웹서비스는 XML 사용으로 메시지 교환의 상호운용성(interoperability)이 높다. 상호운용성은 다양한 웹서비스의 동적인 발견과 합성으로, 부가 가치를 가진 새로운 형태의 웹서비스를 창출한다. XML 기반의 상호운용성을 위한 개방형 표준 기술은 SOAP, WSDL, UDDI 등이 있다[3]. 그러므로 웹서비스 이용자는 H/W에 무관하게 서비스 컨텐츠의 이용이 가능하다. 즉 PC와 PDA, 휴대폰, 모바일 같은 다양한 디바이스에 의한 웹서비스 접근이 용이하다. 따라서 인터넷 기반의 웹서비스 기술을 모바일 환경과 연계하는 유비쿼터스 환경에서 무선 네트워크의 컨텐츠 서비스에 대한 필요성이 대두되고 있다.

웹서비스는 서비스 교환과 기술, 등록과 발견의 단계로 구성되며, 웹서비스의 구조는 다양한 하드웨어와 소프트웨어의 독립성과 상호운용성을 보장하는 표준 기술 기반으로 서비스 제공자와 서비스 저장소(Service Registry), 그리고 서비스 요청자 등이 유기적인 결합으로 구현된다[4].

2.2 아파치 웹 서비스

아파치(Apache) 웹 서버는 NCSA(National Center for Supercomputing Applications)의 httpd 1.3을 기반으로 개발되었다. Httpd 서버는 트래픽 증가에 비효율적이고, 여러 웹 사이트 관리 문제도 발생한다. 1.3 버전까지는 프리포크 기법의 다중프로세스 모델을 사용하여 매우 안정적이지만, 프로세스 생성에 많은 오버헤드로 반응이 느리다. 반면에 쓰레드는 생성시 부하가 적고 자원을 공유가 가능하다. 아파치 2.0 버전에서 기존의 1.3 버전과 차이점은 멀티쓰레딩의 지원으로 웹 서버의 확장성을 증가시키고, 여러 프로세스와 여러 쓰레드의 혼합 실

행이 가능하다[5].

아파치 톰캣 AXIS의 사용은 웹서비스를 쉽고 빠르게 구현하지만, 가장 큰 문제는 웹 서버 외에 톰캣을 설치해야 한다. 톰캣 서블릿 엔진의 사용은 웹 서버 설치와 인터넷 포트가 추가와 처리 시간을 요구한다. 따라서 서버의 보안과 관리에서 부가적인 추가 요소가 발생한다. 이러한 단점 보안을 위해 웹 서버에 부가적인 웹서비스 모듈의 추가 작업이 필요하다.

그리고 이전에는 클라이언트가 서버에 접속할 때마다, 클라이언트의 요청 처리를 위해 쓰레드를 실시간으로 생성하여 접속을 처리하였으나, 이러한 경우는 시간과 자원을 많이 사용하여, 다수의 클라이언트가 동시에 접속하는 경우는 정상적인 동작이 불가능하다.

2.3 모바일 웹 서비스

모바일 웹서비스 환경은 디바이스의 크기, CPU 성능, 저장 공간, 화면의 크기, 네트워크 속도, 입력 장치 등에서 많은 차이를 나타내며, 휴대성과 신속성의 장점을 제외하고는 대부분의 면에서 데스크톱 환경과 비교하여 다양한 면에서 떨어지는 조건을 갖고 있다[6].

플랫폼 독립적으로 디바이스와 소프트웨어를 연계한 데이터 전송과 정보 통신 기술을 통합하는 디지털 컨버전스 기술의 중심에는 모바일 장치가 존재하고, 기존의 인터넷 기반의 서비스가 모바일 기반의 서비스로 변하고 있다. 모바일 웹서비스는 기존의 웹 정보를 휴대폰이나 PDA에 적합한 형태로 추출하고, 실시간으로 가공하여 데이터를 제공하므로, 전송 효율이나 비용 등의 경제적인 측면에서 많은 장점을 갖는다[7].

무선 인터넷 프로토콜(WAP)은 무선 통신망과 웹서비스의 상호운용을 위하여 모바일 디바이스와 웹서버 사이에 WAP 프록시 기능을 담당하는 WAP 게이트웨이가 존재한다. WAP 게이트웨이는 WAP 프로토콜과 HTTP 프로토콜 중간에서 메시지 변환과 WAP 프로토콜(WSP, WTP, WTLS, WDP)과 IP 기반의 패킷 네트워크 사이에서 메시지를 변환하는 인터페이스이다.

WAP 기반의 모바일 디바이스는 WAP 프록시를 통해서 웹서버에 연결하고, 웹서비스 연결 요청은 WAP 프로토콜에 의해 메시지를 압축하여 WAP 프락시로 전송하면, 프락시는 메시지를 디코딩하여 URL로 웹서버에 요청한다. 다음의 [그림 1]은 WAP 기반의 무선 인터넷 구성도를 나타낸다.

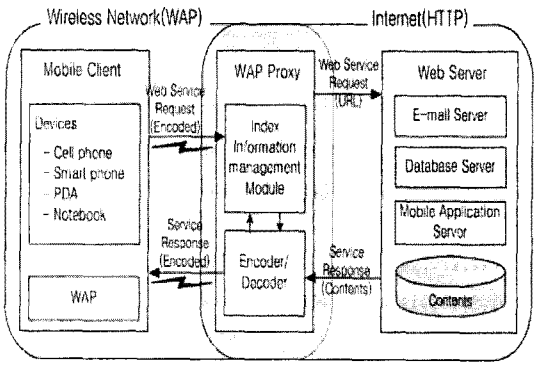

그림 1. WAP 기반의 무선 인터넷 구성도
Fig. 1 Configuration of the WAP-based wireless Internet

웹서버는 WML과 WML 스크립트 형태로 모바일 디바이스의 요청 결과를 전송하고, WAP 프록시는 WAP 프로토콜 형식으로 응답 서비스의 변환과 인코딩하여 모바일 디바이스로 전송한다. WAP 프록시는 HTML과 WML 사이의 변환 수행으로 WAP 게이트웨이, 인터미디어리라고 한다. WML은 대역폭, 메모리, CPU 측면에서 제약을 가진 모바일 디바이스에 웹서비스 제공을 위한 디스플레이 표현과 사용자와 상호작용을 표현하는 XML 기반의 WAP 표준이다[8].

인터미디어리는 프레임워크의 유연성을 지원하며, 무선 네트워크 환경에서 웹서비스의 사용을 용이하게 한다. 인터미디어리는 서비스 요청자와 공급자 중간에 위치하는 구성 요소이며, 요구와 응답 메시지의 전송을 담당하며, 타임스탬프, 로그 등 제3의 공증을 담당한다. 그리고 클라이언트와 웹서버사이의 메시지 변경 처리하여 캐싱, 필터링, 코드 변환, 개인 특성화 기능들을 지원한다[9,10]. 인터미디어리 서버는 SOAP과 아파치의 AXIS 수행에 의해 지원된다.

Ⅲ. 인터미디어리 기반의 웹서비스 프레임워크 설계

3.1 인터미디어리 프레임워크의 구성 요소

본 논문에서는 프레임워크의 유연성, 모바일 웹서비스의 성능 향상, 무선 네트워크 환경에서 웹서비스 제공 그리고 모바일 디바이스와 웹서버의 상호운용을 위해

인터미디어리 기반 웹서비스 프레임워크를 설계/구현한다. 톰캣 AXIS를 사용하는 기존의 웹서비스 형태의 단점 보안을 위해 서블릿 엔진을 제거하고 부가적인 웹서비스 모듈을 직접 구현/추가하여 모바일 디바이스의 SOAP 요청과 응답 메시지를 직접 처리하는 인터미디어리 기반의 시스템을 설계한다. 인터미디어리 기반의 웹서비스 구현을 위해서 표준의 WSDL 문서와 SOAP 메시지를 사용한다.

본 논문에서 제안한 프레임워크는 모바일 디바이스가 웹서비스의 호출을 위하여 기존 시스템의 확장된 구조를 나타낸다. 인터미디어리 기반 웹서비스 모델은 그림 2와 같으며 기본적으로 인터미디어리 서버와 무선 통신을 담당하는 모바일 디바이스로 구성하며, 세부적인 구성 요소는 모바일 메시지 에이전트, WSDL 전처리기, SOAP 메시지 처리 에이전트 그리고 세션 관리 에이전트를 포함한다.

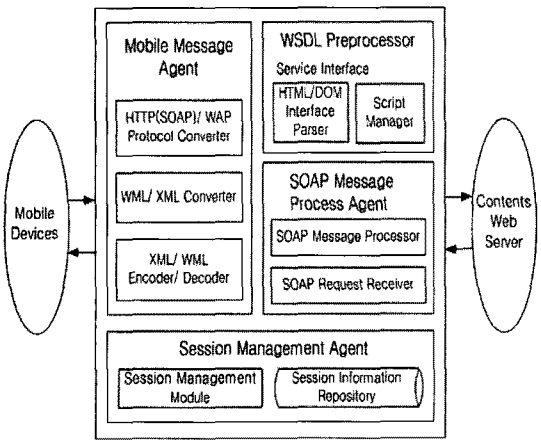

그림 2. 인터미디어리 기반의 웹서비스 구성 요소
Fig. 2 Web services components of the Intermediary-based

본 논문에서 제안한 모바일 웹서비스를 위한 인터미디어리는 휴대폰, PDA, 네비게이터 등과 같은 모바일 디바이스와 웹서버 중간에서 인터넷 기반에서 제공되는 각종 컨텐츠들과 동적인 웹서비스를 모바일 사용자에게 제공한다. 또한 모바일 웹서비스 환경과 인터넷 기반 환경이 다양한 면에서 서로 상이하므로 필요한 구성 요소로 HTTP/SOAP 프로토콜과 WAP 프로토콜 사이의 변환 기능 그리고 인터넷 기반의 XML 문서와 WML 문서의 변환/인코딩 기능 등을 제공한다.

3.2 모바일 메시지 에이전트

모바일 메시지 에이전트는 HTTP(SOAP)/WAP 프로토콜 변환기, WML/XML 변환기, WML/XML 인코더/디코더로 구성하며, 모바일 디바이스로부터 웹서비스 요청 메시지, WSDL에 대한 요청 메시지 그리고 핸드오프 처리 요청 메시지를 수신하여 분류 처리한다.

HTTP(SOAP)/WAP 프로토콜 변환기는 모바일 환경과 인터넷 기반에서 메시지 전송이 서로 상이하므로 클라이언트의 요청 메시지 또는 웹서버의 서비스 결과 전송을 위하여 각각의 환경에 적합한 프로토콜로 변환하는 기능을 담당한다.

모바일 디바이스의 웹서비스 요청 메시지인 경우는 SOAP 메시지 처리 에이전트에서 WAP을 사용하는 클라이언트의 WML 웹서비스 요구 처리를 위하여 WML/XML 변환기에서 XML로 변환하여 웹서비스가 존재하는 웹서버와 HTTP 통신 기반 XML/SOAP 메시지 송수신으로 연결한다. 그리고 웹서버에서 수신한 서비스 결과는 WML/XML 변환기에서 모바일 형식(WML)으로 변환하여 모바일 디바이스로 전송한다.

그리고 핸드오프 처리 요청 메시지인 경우는 세션 관리 에이전트로 클라이언트 요청을 전달하여 세션 복구 요청을 처리하고, 사용하지 않는 세션 정보의 관리와 처리를 담당한다. 핸드오프 처리 결과에 의한 메시지는 모바일 디바이스에게 WAP/WML 기반으로 전송한다. 모바일 메시지 에이전트로부터 핸드오프 메시지 처리에 대한 정보를 가지고 세션 관리 에이전트에 의해 핸드오프 처리 메시지를 생성하여 통신하므로 웹서버로부터 완전한 서비스 결과의 수신이 가능하다.

3.3 WSDL 전처리기의 설계

모바일 웹서비스를 위해서는 기본적으로 WSDL 기술이 필요하다. 기존의 웹에 존재하는 HTML 데이터를 모바일 통신을 위한 WSDL 데이터 형식으로 변환을 위하여 WSDL 전처리기를 설계한다. WSDL 전처리기는 웹서비스에 대한 요청을 WSDL 파일로 자동으로 생성하고 클라이언트 요청으로 활성화한다. 본 논문에서 제안한 WSDL 생성기는 HTML/XML 파서를 구현하여 WSDL 생성을 지원하고, 클라이언트의 요청 메시지를 WSDL로 변환한다. WSDL 전처리기는 Java 1.4의

org.w3c.dom 라이브러리를 이용하여 AXIS에 독립적으로 실행하며, Dom의 Document 객체를 사용하고, 모든 태그는 Element 객체를 사용한다. WSDL 전처리기는 특정 스크립트에 대한 WSDL을 생성한다. 모바일 디바이스의 WSDL 요청 처리 과정은 그림 3과 같다.

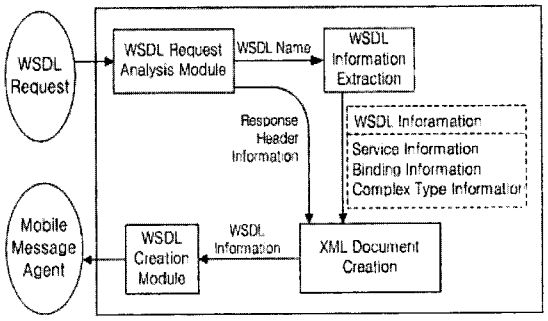

그림 3. WSDL을 요청하는 경우의 처리 과정
Fig. 3 A process of a case to request WSDL

모바일 클라이언트가 모바일 메시지 에이전트를 통해 접속하고, 모바일 클라이언트가 WSDL을 요청하는 경우 클라이언트의 요청을 분석하고 WSDL 생성을 위하여 HTML/XML 파서를 이용한다. 모바일 클라이언트의 XML 웹서비스 메시지에 대한 WSDL 정보를 생성하는 과정은 다음과 같다.

- 응답 헤더 정보를 생성(new HttpHeader())
- WSDL의 정보 획득(ScriptManager.getScript())
- 새로운 XML 문서 정보 생성(DocumentBuilder.new Document())
- WSDL에 필요한 엘리먼트 생성(Document.create Element())
- WSDL의 모든 서비스, 바인딩에 필요한 엘리먼트 추가(WSDLInfo.addOperation())
- 바인딩 VARIABLE 표현에 필요한 complexType 추가(WSDLInfo.addComplexType())
- 생성한 XML 문서에 element 추가(Document.append Child())

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

3.4 SOAP 메시지 처리 에이전트 설계

SOAP 메시지 처리 에이전트는 모바일 클라이언트의 웹서비스 요청 접수와 처리를 위하여, SOAP 요청 접속 모듈과 SOAP 메시지 분석 모듈, 서비스 프로세서 모듈 그리고 SOAP 메시지 생성 모듈로 구성한다.

모바일 클라이언트의 접근 URL은 WSDL을 통해 획득하며, http://host주소:port/webservice/ WSDL명 형태를 가진다. 클라이언트가 SOAP 형태의 XML 문서로 웹서비스 요청을 위해 요청 서비스명, 입력값, 세션 아이디를 포함한 SOAP 메시지를 서버로 전달한다.

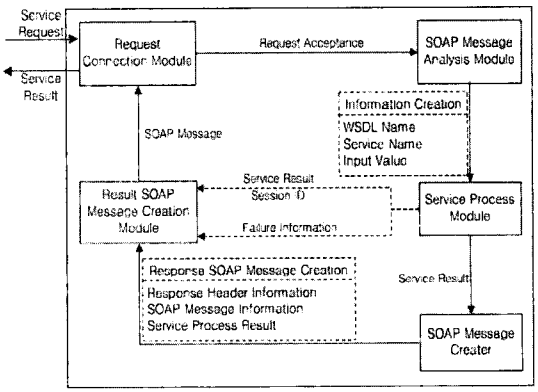

그림 4. 클라이언트의 웹서비스 요청 처리 과정
Fig. 4 A Web service request process of a client

[그림 4]는 클라이언트의 웹서비스 요청 처리 과정으로, SOAP 요청 접속 모듈은 모바일 메시지 에이전트가 전달한 클라이언트 요청을 접수/분석하여, WSDL 요청인 경우는 WSDL 전처리기에 요청을 전달하고, 웹서비스 요청인 경우는 SOAP 메시지 분석 모듈에 전달한다.

SOAP 메시지 분석 모듈에서는 클라이언트의 요청을 분석하고, 요청 서비스의 WSDL명을 추출한 후 전달된 SOAP 메시지를 분석하여 정보를 생성한다. 분석 정보에서 서비스명과 입력값들을 추출한 후 서비스 처리부에 처리를 요청한다.

SOAP 메시지 생성 모듈에서는 서비스 처리부에서 처리한 결과를 SOAP 메시지로 생성한다. 응답 SOAP 메시지 생성을 위해 응답 헤더 정보와 응답 SOAP 메시지에 대한 정보를 생성하고, 서비스 처리 결과가 정상인지 확인하여, 정상이면 서비스 결과값과 세션 아이디를 추가

하고, 정상이 아니면 실패 정보를 추가한다. SOAP 메시지 정보에서 SOAP 메시지를 생성하여 모바일 메시지 에이전트로 전달하면, 모바일 메시지 에이전트는 다시 모바일 클라이언트로 전달한다.

3.5 세션 관리 에이전트(Agent)설계

모바일 디바이스의 특성상 이동성에 의해서 핸드오프가 발생한다. 핸드오프는 모바일 디바이스가 무선 구역의 변경으로 통신 채널의 자동 전환 과정에서 통신의 단절, QoS(Quality of Service)가 감소하는 상황을 나타낸다. 따라서 세션 관리 에이전트에서는 모바일 이동성에 무관하게 양질의 웹서비스를 제공하기 위한 기능을 담당한다.

본 논문에서는 개선된 인터미디어리 웹서버 성능 향상을 위하여 모바일 클라이언트의 세션 정보 관리를 주기적으로 실시한다. 또한 NBTM 관리자 구현으로 클라이언트의 접속 요청에 대하여 논-블록킹 I/O를 사용하고 클라이언트 요청 처리를 위하여 쓰레드풀을 사용하여, 통신 유휴 시간에 쓰레드 사용을 방지하여, 자원을 낭비하지 않고, 많은 네트워크 I/O를 처리한다.

모바일 클라이언트는 여러 방법으로 HTTP 기반의 인터미디어리 웹서버에 접속하고, 세션 관리 에이전트는 클라이언트의 세션 정보를 생성하여 세션 정보 저장소에 저장하고, 클라이언트의 세션 복구 요청에 의해 클라이언트의 세션 복구 요청을 처리하고, 사용하지 않는 세션 정보의 관리 및 처리를 담당한다. 다음의 [그림 5]는 세션 관리 에이전트의 처리 과정을 나타낸다.

모바일 클라이언트의 접속이 수락되면 접속 클라이언트의 서버 이용 가능 여부를 현재 접속 클라이언트의 수(Current User)와 최대 수용 가능한 접속 수(Max User)를 비교하여 판단한다. 클라이언트의 서버 이용이 가능하면, 접속 클라이언트의 정보 확인과 확인된 정보를 사용하여 세션 정보를 생성하고 클라이언트에 부여할 세션 아이디를 생성하고, 세션 정보를 세션 아이디에 매핑시켜 세션 정보 저장소에 저장하고, 세션 아이디를 포함한 접속 성공 메시지를 생성하지만 서버 이용이 불가능하면 접속 실패 메시지를 생성한다.

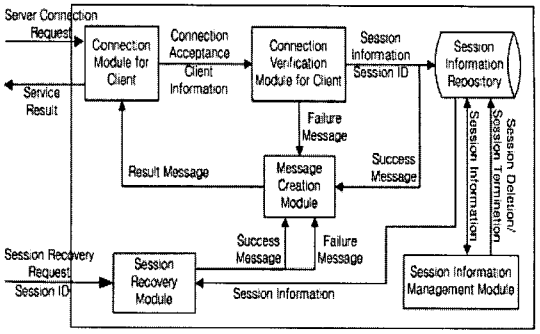

그림 5. 세션 관리 에이전트의 처리 과정
Fig. 5 A process of a session management agent

생성된 메시지는 클라이언트로 전송하고, 클라이언트는 결과 메시지를 수신하여 성공 메시지인지 판단하여 성공 메시지인 경우는 차후 세션 복구가 필요할 때 사용할 세션 아이디를 기억하고 실패 메시지인 경우는 서버와의 연결을 종료한다. 만약에 의도하지 않은 상황에 의해서 서버와 클라이언트의 연결이 끊어진 경우에는 클라이언트가 이전의 세션을 복구한다.

Ⅳ. 구현 및 평가

4.1 실험 환경 및 방법

본 논문의 모바일 인터미디어리 웹서비스 시스템의 성능 평가를 위한 실험은 각각 구성이 다른 2개의 웹서비스 시스템을 준비하였다. 첫 번째 시스템은 기존 방식을 적용하는 전형적인 Tomcat 5.5와 AXIS 기반의 웹서비스 시스템이고, 또 다른 시스템은 본 논문에서 제안한 AXIS 기반의 서블릿 컨테이너를 제거한 웹서비스 시스템이다. 각각의 시스템 구현을 위해 사용된 구성 요소는 표 1과 같다.

표 1. 실험 시스템의 구성 요소
Table. 1 Components of an experiment system

	클라이언트	서버
OS	Windows 2000	Windows 2000
CPU	Intel XEON 2.4GHz	Intel Xeon 3.0GHz
Memory	2GB RAM	2GB RAM
사용언어	Java, JDK 1.4	
브라우저	IE, Firefox	

4.2 구현 및 평가

본 논문에서 제안한 프레임워크의 성능 향상을 위해 모바일 메시지 에이전트, WSDL 전처리기, SOAP 메시지 처리 에이전트 그리고 세션 관리 에이전트 모듈이 인터미디어리 웹서버에 각각 구현되었다. 따라서 인터미디어리 웹서버에서는 메시지 조작, 체인, WSDL 처리 기능, 그리고 SOAP 요구 메시지의 처리가 서블릿 컨테이너와 무관하게 이루어진다.

본 논문의 실험 시스템 구성은 그림 6과 같이 모바일 클라이언트와 제안한 인터미디어리 웹서버 그리고 콘텐츠 서버가 존재한다. 모바일 디바이스가 SOAP 형태로 웹서비스를 요청하면, 실험 시스템은 SOAP 메시지와 내용을 콘텐츠 웹서버로 HTTP 호출로 전송한다. 콘텐츠 웹서버의 HTTP 응답을 실험 시스템이 수신하면, 실험 시스템은 WSDL을 생성시키고 클라이언트인 모바일 디바이스로 SOAP 응답 메시지를 전송한다.

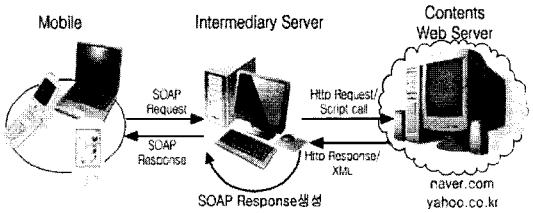

그림 6. 실험 시스템의 사용자 요구 처리 과정
Fig. 6 A user request process of an experiment system

실험을 위한 가정으로 SOAP 요구는 모바일 클라이언트 시뮬레이션 프로그램에 의해 생성되고, 실험 시스템에 연결되어 여러 개의 SOAP 요구 메시지를 보낸다. 시간 간격은 1초에서 10초 사이에 SOAP 요구를 반복 요청한다. 연결이 종료되면 시뮬레이션 프로그램은 실험 시스템에 연결을 위해 반복적인 연결 시도를 한다.

시스템에 접근하는 사용자는 동시에 200명으로 가정하고 모바일 디바이스의 요구는 4대의 클라이언트에서 중계 프로그램을 실행시켜 생성하였다. 4대의 클라이언트 프로그램에 의해 모바일 디바이스의 요청이 새로 생성되고 각각의 프로그램은 즉시 50개 쓰레드를 생성시켰다. 각각의 서버 타임아웃은 30초이다.

본 논문의 실험을 위한 시스템 구성에 대한 테스트 결과는 표 2와 같다. 테스트 결과에서 나타나는 것처럼 제안한 실험 시스템의 결과가 연결 오류 횟수는 비교 대상 모두 0%이며 리퀘스트 오류 개수는 0.024%로 더 좋게 나왔다.

표 2. 각각의 버전에 대한 테스트 결과
Table. 2 The test results regarding each version

	아파치 1.52	아파치 2.0	제안시스템
전체 시간	1시간 2분	1시간 9분	1시간 20분
연결 오류 횟수	0/25883/0/55968 (0%/46.25%/0%)	0/2924/0/38228 (0%/7.65%/0%)	0/0/0/41138 (0%/0%/0%)
리퀘스트 오류 횟수	5523/36149 (15.278%)	11032/37162 (29.686%)	10/40948 (0.024%)

* 백분율=오류 횟수/전체연결시도횟수
* 연결오류: 타임아웃 횟수/refused횟수/핸드쉐이크 에러 횟수/전체연결시도횟수
* 리퀘스트오류: 타임아웃 횟수/전체요청 횟수

그림 7은 4개의 실험 시스템에 대한 트랜잭션 지연에 대한 결과이다. 실험은 하나의 연결에 다수의 문자를 전송한다. 초기 TCP, HTTP 연결을 설정하고 SOAP 메시지로 문자를 전송한다. 제안 시스템의 지연이 적은 이유는 서버의 처리 시간 단축으로 인한 것이다. 그림 7에서 나타나는 것처럼 모든 항목에서 성능 향상이 이루어졌다. 이 논문의 실험은 모바일 웹서비스 시스템의 실행 평가에 초점을 맞췄다.

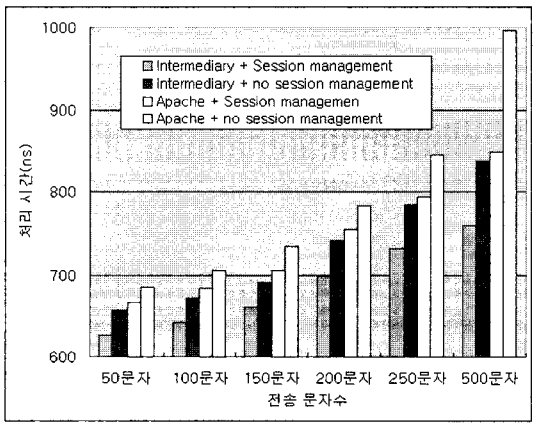

그림 7. 각각의 문자에 대한 처리 결과
Fig. 7 The test results regarding each character

표 3는 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 연결 설정이 안 된 개수, connection handshake error는 세션 연결 오류 횟수 그리고 connection trial은 클라이언트의 연결 시도 자체를 못한 횟수이다.

표 3. 100개 동시 요청의 실험 결과(단위 : 개수)
Table. 3 The experiment results of a request simultaneous 100

비교시스템	Req. Timeout	Connection refuse	Handshake error	Connection Trial
웹2.0	22.6	3.6	0.8	4.2
제안시스템	1.4	0.2	0	0.4

실험은 1초에 100개의 접속을 시도한다. 실험 과정에서 request timeout의 경우 10개 이하의 request 경우에 두 시스템 모두 제한 시간 내에 요구가 성공하였으며, 100개의 경우에 제안 시스템은 1.4개 미만의 발생 확률을 가지고, 표준 시스템은 22.6개의 발생 확률을 가진다.

<표 4>의 결과는 서블릿 컨테이너를 사용하는 것보다 본 논문의 제안 시스템의 효율이 높다는 것을 의미한다. 그리고 웹서비스의 성능 향상은 전체 시스템의 성능 향상을 가져와 누적되는 연결 신호 처리의 향상을 의미한다. 즉 하나의 처리 시간이 짧기 때문에 연속적인 요청을 빠르게 처리할 수 있고, 이의 차이는 요청이 증가할수록 더 큰 격차를 가져온다. 따라서 인터넷 기반의 웹서비스는 시스템의 응답 속도 개선이 전체적인 성능에 많은 영향을 끼치는 것을 의미한다. 결국 연결 요청 횟수가 증가할수록 기존의 웹서비스 시스템의 오류 횟수는 기하급수적으로 증가한다.

V. 결 론

본 논문에서는 모바일 웹서비스의 성능 향상과 지연처리의 해결을 위하여 새로운 모바일 웹서비스 아키텍처를 제안하였다. 새롭게 제안한 웹 서버는 부가적인 모바일 웹서비스 모듈을 직접 구현하여 추가하였다. 또한 톰캣 서블릿 컨테이너를 제거하여 SOAP에 내재된 지연 문제를 줄이는 새로운 SOAP 메시지 처리를 위한 인터미디어리 웹서버의 프레임워크를 제안하였다. SOAP 요구와 응답 메시지는 인터미디어리 웹서버 시스템에 의해 직접 처리된다.

또한 I/O 지연 현상의 감소와 자원 절약, 많은 네트워크 I/O 처리를 위해 다중 쓰레드를 사용하는 논-블록킹 I/O를 구현하였다. 그리고 연결 성공 비율을 증가시켜 개선된 웹서버 성능의 향상을 위하여 클라이언트의 세션 정보 관리를 주기적으로 실시하였다. 논-블록킹 쓰레드 매니저는 연결 요청의 증가에 따른 연결 오류의 비율을 감소시킨다. 따라서 웹서버 성능을 향상시킴으로써 처리 시간을 단축하여 성능 향상을 이루었다.

본 논문에서는 웹서비스 표준 프로토콜을 기반으로 하는 개선된 모바일 인터미디어리 웹서비스 시스템을 구현하였다. 실험 결과 무선 네트워크에서 아파치 2.0 버전을 사용하여 구성한 시스템보다 6.7%, 전통적인 아파치 1.52 버전의 웹서비스 보다 9.7%의 성능 향상을 얻을 수 있었다. 또한 동시 요청이 50개 이상의 경우 접속 성능은 약 38% 향상되었다. 따라서 실험 결과는 접근의 수행을 처리하는 SOAP 요구에서 표준 웹서비스 구현보다 성능이 향상되었음을 증명하였다. 본 논문의 연구는 매우 작은 연결 오류를 가지고 있으므로 전형적인 웹서비스 구현보다 효율적인 서비스 요구를 처리할 수 있다. 향후 연구의 중점 사항으로 I/O 시간을 감소시키는 알고리즘에 관한 연구가 필요하다.

참고문헌

[1] Paul Muschamp, "An introduction to Web Services", BT Technology Journal 22, No 1, pp. 9-18, Springer Netherlands, 2004.

[2] 김성원, "P2P를 적용한 인터넷비즈니스의 가능성에 관한 연구", 이비즈그룹, working paper no.8, 2000.

[3] 허정회, "웹 서비스 보안 기술 분석 및 응용 방안 연구", NCA IV-RER-03074, 한국전산원, 2003. 12.

[4] "Understanding Web services", Microsoft Corporation. Inc, 2005.

[5] http://www.superuser.co.kr/apache/apache2_manual/new_features_2_0.html

[6] 전 종홍, 이승윤, "웹 2.0 기술 현황 및 전망", 전자통신연구소, 전자통신동향분석 제21권 제5호, 2006.

[7] Gehlen, G., Bergs, R., "Performance of mobile Web Service Acc Protocol (WAP)". In Proceedings of World Wireless Congress. Sanfransico, USA ess using the Wireless Application, pp. 427-432, 2004.

[8] Sakkopoulos E., Lytras M., Tsakalidis A., "Adaptive mobile web services facilitate communication and learning Internet technologies", Education, IEEE Transactions on, Vol 49, pp. 208-215, 2006.

[9] A Pashtan, S Kollipara, M Pearce, "Adapting Content for Wireless Web Services", IEEE Internet Computing, Vol. 7,pp.79- 85 , 2003

[10] R Grieco, D Malandrino, F Mazzoni, V Scarano, "Mobile-Web Services VIA Programmable Proxies", Springer Boston,Vol. 191, pp. 139-146, 2005.

저자소개

김 용 태(Yong-Tae Kim)

1984년 한남대학교 계산통계학과
1988년 숭실대학교 전자계산학과 석사.
2008년 충북대학교 전산학과 이학박사
2002년~ 2006년 가림정보기술 이사
2006.3 ~ 현재 한남대학교 멀티미디어학부 강의전담 교수
※관심분야 : 모바일 웹서비스, 정보보안, 센서 웹, 모바일 통신보안, 멀티미디어

정 윤 수(Yoon-Su Jeong)

2000년 2월 : 충북대학교 대학원 전자계산학 이학석사
2008년 2월 : 충북대학교 대학원 전자계산학 이학박사
※관심분야 : 센서 보안, 암호이론, 정보보호, Network Security, 이동통신보안

박 길철(Gil-Cheol Park)

1983년 한남대학교 계산통계학과
1986년 숭실대학교 전자계산학과 석사.
1998년 성균관대학교 전자계산학과 박사.
2006. UTAS, Australia 교환교수
1998. 8. ~ 현재 한남대학교 멀티미디어학부 교수
※ 관심분야 : multimedia and mobile communication, network security