韓國 컴퓨터情報學會 論文誌
第13卷 第5號, 2008. 9.

2008-13-5-3-7

SOAP 메시지 처리 개선과 NBTM을 사용한 모바일 웹 서버의 성능 향상

김 용 태*, 정 윤 수**, 박 길 철***, 이 상 호****

Mobile Web Server Performance Improvement with Enhancing SOAP Message Transaction and NBTM

Young-Tae Kim*, Yoon-Su Jeong**, Gil-Cheol Park***, Sang-Ho Lee****

요 약

최근의 급속하게 변화하는 모바일 인터넷 환경에서 이전의 웹 서버 성능으로는 사용자의 빈번한 연결 요구에 대해 적절한 대응이 어렵기 때문에 웹 서비스 엔진의 성능 향상이 필요하다. 따라서 본 논문은 웹 서비스의 성능 향상을 위해서 부가적으로 설치하는 톰캣 서블릿 컨테이너와 무관하게 사용자의 접속 요청을 처리하고, SOAP(Simple Object Access Protocol) 메시지 처리 시간을 단축하는 모바일 웹 서비스의 서버 구조를 제안한다. 제안한 웹 서버 구조는 사용자 요구 수신기, 웹 문서 처리기, SOAP 프로토콜 처리기, NBTM(Non-Blocking and Thread Manager) 관리자, 세션 관리자로 구성한다. 제안된 시스템은 표준 웹 서비스 프로토콜을 완전하게 지원하고, 웹2.0의 모바일 웹 서비스 시스템의 통신 오버헤드와 메시지 처리 시간, 서버의 오버헤드를 감소하며, 웹2.0 환경에서 구현 실험을 통한 지연 수행 평가에 의해서 웹2.0 표준 모바일 웹 서비스 시스템과 성능을 비교, 평가하여 성능 향상을 확인하였다.

Abstract

Recently, the mobile Internet is rapidly changing environment in a user's web server performance requires frequent connection is difficult because the proper response is needed to improve the performance of web services engine. Therefore, the goal of this paper is to improve the performance of web services for the installation of additional access to users regardless of tomcat servlet container and process the request and to shorten processing time of SOAP (Simple Object Access Protocol) message of the server structure to offer web mobile services. The web-sever structure is consists of user demand receiver, web document processor, SOAP protocol processor, NBTM(Non-Blocking and Thread Manager) manager, session manager. The system fully supports a standard web-service protocol, reduces the message processing time and communication overhead of mobile web-service system in Web 2.0 and overhead of server, and confirms the improvement of capability by comparing and evaluating Web 2.0 standard mobile web-service system through delaying achievement evaluation by the experiment of realization in Web 2.0 environment.

• 제1저자 : 김용태
• 접수일 : 2008. 4. 8, 심사일 : 2008. 5. 29, 심사완료일 : 2008. 9. 25.
* 한남대학교 멀티미디어학부 강의전담 교수 ** 충북대학교 전자계산학과(교신저자: 정윤수)
*** 한남대학교 멀티미디어학부 교수 **** 충북대학교 전기전자 컴퓨터공학부 교수
※ 본 연구는 지식경제부 지역혁신센터 사업인 민군겸용 보안공학 연구센터 지원으로 수행되었음

www.kci.go.kr

▸ Keyword : 모바일 웹 서비스(Mobile Web Services), 웹 2.0(Web 2.0), 아약스(AJAX), 단순 객체 접근 프로토콜(SOAP), 웹 서비스 기술 언어(WSDL), 아파치(Apache)

Ⅰ. 서 론

인터넷 환경과 관련 기술의 급속한 발전으로 정보 기술의 패러다임의 변화와 웹의 가능성 때문에 비즈니스 어플리케이션과 인터넷 이용자가 급속도로 증가하고 있다. 그리고 동적인 프로그램, 멀티미디어 파일과 같은 대용량의 요구가 증가하는 웹에서 기존 웹 서비스 기술의 적용은 네트워크의 트래픽과 서버의 지연 처리 문제가 발생한다. 그리고 아파치와 톰캣의 멀티쓰레드 패러다임을 기반으로 하는 대부분 웹 서버에서 쓰레드는 클라이언트에 의한 연결 종료까지 모든 요구를 체크하는 부담을 가지며, 동시에 클라이언트의 최대 개수가 생성된 쓰레드의 개수에 제한을 받는다.

또한 서버가 과부하 상태인 경우에는 폐쇄 연결, 암호 핸드쉐이크 수행 횟수를 증가시키고[1], [2]에서 나타나는 것처럼 서버 처리량을 감소시키고 평균 응답 시간을 증가시키면서, CPU 작업을 낭비하기 때문에 웹 어플리케이션 서버의 성능에 부정적인 영향을 나타낸다. 따라서 효과적인 방법으로 서버의 과부하를 개선하고 성능 향상을 위한 확장된 모바일 웹 서비스 엔진의 필요성이 증대되고 있다.

본 논문에서는 사용자들이 가장 많이 사용하는 아파치(Apache) 웹 서버의 문제점을 분석하여 개선 방안을 제시하고, 모바일 웹 서버의 성능 향상과 지연 처리에 필요한 기술을 제안한다. 또한 무선 모바일 환경의 특성인, 순간 단절 현상에 대처하기 위하여, 세션 정보를 관리하여 동일한 접속 요청시 재접속의 문제를 해결하고, 신규 접속의 성능을 향상시키기 위하여 필요한 기술을 제안한다.

본 논문은 5개의 장으로 구성한다. 1장은 서론으로 연구 목적과 필요성을 기술하고, 2장은 관련 연구와 기존 시스템의 문제점과 성능 개선 요소를 분석한다. 3장은 웹 서비스의 성능 개선을 위한 제안 시스템의 구성에 대하여 기술한다. 4장은 제안 시스템의 설계와 구현을 통하여 기존 시스템인 SMSP[3]와 성능을 비교, 평가한다. 마지막으로 5장은 결론과 향후 연구 방향을 기술한다.

Ⅱ. 관련연구

2.1 웹 서비스의 개요

웹 서비스는 서비스 지향 아키텍처(SOA: Service Oriented Architecture)를 실현하며, 비즈니스 프로세스 실행과 어플리케이션 통합을 위한 기술로 구별한다. 웹 서비스는 개방형 표준인 XML을 사용하는 인터넷 기반(HTTP 프로토콜)의 자료 교환에 대한 컴포넌트이며, 서비스 교환과 기술(description), 등록(registration)과 발견(discovery) 단계로 구성한다. XML에 의한 메시지 교환으로 상호운용성(interoperability)이 높다. 웹 서비스의 상호운용성은 다양한 웹 서비스의 동적인 발견과 합성으로 부가 가치를 가진 새로운 형태의 웹 서비스로 나타난다[4].

다양한 하드웨어와 소프트웨어의 독립성과 상호 운용성을 위한 웹 서비스 프로토콜은 SOAP, WSDL(Web Services Description Language), UDDI(Universal Description, Discovery and Integration of Business for Web) 등과 같은 개방형 기술로 구성되며, SOAP은 웹 서비스의 요청/응답을 위한 메시지 형식을 정의하고, WSDL은 웹 서비스 이용에 필요한 인터페이스와 입/출력 메시지의 형식 등을 기술하고, UDDI는 웹 서비스의 디렉토리 서비스를 위한 표준으로, 웹 서비스를 등록하고 검색/발견 메커니즘을 제공한다[5].

다음의 [그림 1]은 웹 서비스와 관련된 3가지 표준인 SOAP(웹 서비스 호출), WSDL(서비스 기술), UDDI(웹 서비스 등록/검색)의 상호 관계를 나타낸다[6].

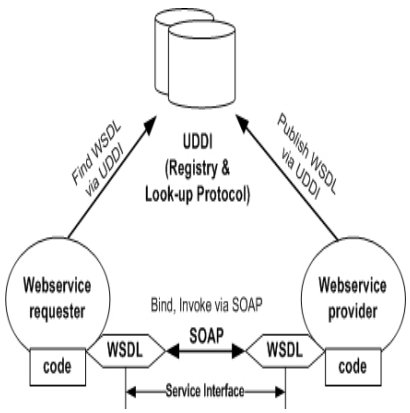

그림 1. 웹 서비스의 동작 개념
Figure 1. Control Architecture of Web Services

www.kci.go.kr

2.2 아파치 웹 서비스

아파치 웹 서버는 NCSA(National Center for Supercomputing Applications) httpd 1.3을 기반으로 개발된 웹 서버이다. httpd 서버는 트래픽 증가에 비효과적이고, 여러 웹 사이트의 관리 문제가 발생한다. 아파치 1.3 버전까지 개발 목표는 안정성과 풍부한 기능성의 제공이지만, 아파치 2.0은 빠른 속도, 다양한 플랫폼 지원, 프로그래밍 인터페이스를 통한 견고한 모듈 개발을 목표로 한다.

초기의 아파치 1.3 버전까지는 멀티프로세스 아키텍처 기반의 프리포크 기법의 사용으로 안정적이지만, 프로세스의 생성과 스케줄링에 많은 오버헤드로 응답 시간이 느리다. 아파치 2.0 버전은 멀티쓰레딩 기능의 추가로 기존의 한계인 확장성과 유연함을 갖는 하이브리드 웹 서버를 구축한다. 또한 쓰레드 지원으로 웹 서버의 확장성을 증가시키고, 프로세스와 쓰레드의 혼합 실행이 가능하다[7].

2.3 웹2.0 환경의 웹 서비스 구조

웹2.0은 개념적으로 많은 부분의 추가로 기술적인 측면에서 기존의 다양한 XML 응용, SOA, 브라우저 확장 기술, RIA, 웹 서비스 응용, 시맨틱 웹 응용 등과 같은 차세대 웹 기술과 응용들을 포함한다. 웹2.0을 대표하는 기술은 AJAX이며, 자바 스크립트 언어와 기타 웹 표준을 사용하는 기술이며, 보안을 강화하고, 다양한 운영체제와 브라우저를 제공한다. [그림 2]는 전통적인 웹 서비스와 웹2.0 웹 서비스의 구조도이며 차이점은 Ajax 엔진이 추가로 필요하다.

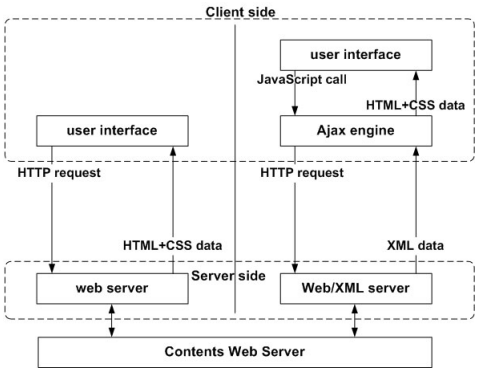

그림 2. 웹2.0에 대한 웹 서비스 구조
Figure 2. Web service structure for the Web 2.0

웹2.0의 대표적인 기술인 AJAX을 활용한 웹 응용은 ActiveX 기반의 응용과 달리 XML을 처리하는 DOM 엔진과 자바 스크립트 엔진을 가진 브라우저나 플랫폼에서 호환되는 장점을 가진다. 또한 비동기적인 데이터 교환에 의해 요청에 대한 서버의 응답을 대기하지 않고 다음 작업이 가능하므로, 대기 시간과 서버의 부담을 줄이고, 사용자 체감 속도를 향상과 이벤트 기반의 처리에 의해 효과적인 사용자 인터페이스 구현이 가능한 장점을 갖는다[8,9].

웹 서비스를 요청할 때 전통적인 웹은 HTTP 기반이지만 웹2.0은 자바 스크립트에 의해 호출하고, Ajax 엔진이 HTTP 요구로 변환하여 서버로 전송한다. 또한 응답할 때에도 서버는 XML 데이터를 Ajax 엔진으로 전송하면 HTML+CSS 데이터로 변환하여 브라우저로 전송한다.

다음의 [그림 3]은 전통적인 웹과 웹2.0의 통신 방식을 나타낸다. 전통적인 웹은 단일선의 연결 형태로 동기화 통신하며, 데이터 전송 요구는 서버가 접수/처리하고 클라이언트로 반환까지 사용자 브라우저는 대기 상태에 있다[8,10].

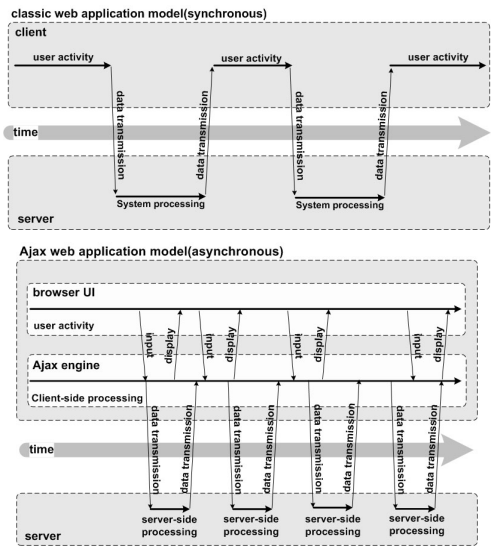

그림 3. 전통적인 웹과 웹2.0의 통신 방식
Figure 3. Communication Methods of Web 2.0, and traditional Web

웹2.0은 비동기 통신하고, 클라이언트의 행위는 이벤트 통합 제어를 담당하는 Ajax 엔진이 인식하고 브라우저 출력 사항과 서버 처리 사항을 구분하여, 브라우저 출력은 즉시 처리하고, 서버 처리 사항은 서버로 전송/처리한다. 이벤트가 완료되면 이벤트는 Ajax으로 인계하므로 서버의 처리가 완료되지 않아도 브라우저는 다른 작업의 수행이 가능하다.

www.kci.go.kr

2.4 기존 웹 서비스 시스템의 문제와 성능 분석

톰캣 AXIS의 사용으로 웹 서비스를 쉽고 빠른 구현이 가능하지만, 문제점은 아파치 웹 서버 외에 톰캣을 설치하여 사용한다. 톰캣 서블릿 엔진을 사용하면, 웹 서버 설치와 추가적인 인터넷 포트가 필요하고 처리 시간을 요구한다. 따라서 서버의 보안과 관리에 부가적인 추가 요소가 발생한다. 이러한 단점을 보안하기 위해 웹 서버에 부가적인 웹 서비스 모듈을 직접 구현하여 추가하는 작업이 필요하다.

그리고 클라이언트의 요청 처리를 위해 실시간으로 쓰레드를 생성하기 때문에 시간과 자원을 많이 사용하고, 다수의 클라이언트가 동시 접속할 때 정상적인 동작이 불가능하다. 또한 기존의 클라이언트와 서버의 통신 방법은 블록킹 I/O를 사용하여, 입/출력 과정에서 쓰레드의 블록킹이 발생하고, 블록킹으로 제외된 쓰레드는 클라이언트 요청으로 다시 활성화되어 대기 상태로 변경한다. 쓰레드 활성화 단계는, 많은 시간과 자원을 요구하며, 클라이언트의 요청에 대해 접속 시간과 응답 시간이 길어지는 단점이 발생한다[11]. 또한, 기존의 다양한 연구 결과는 [12]에서 나타나는 것처럼 아파치 웹 서비스의 성능 결과는 다른 웹 서비스보다 하락함을 나타낸다.

III. 모바일 웹 서비스의 성능 향상을 위한 프레임워크 설계와 세션 정보 관리

3.1 제안 서버 구조의 설계

본 논문은 톰캣 AXIS 기반의 웹 서비스에서 클라이언트 요청에 대한 접속 시간과 응답 시간 지연으로 인한 처리 지연과 웹 서비스 요청 증가에 대한 처리량 감소를 보완하기 위해 서블릿 엔진을 제거하고 직접 SOAP 요구와 응답 메시지를 처리하는 모바일 웹 서비스 시스템을 설계 구현한다.

본 논문에서는 AXIS 모듈을 그대로 사용하면서 톰캣의 오버헤드 감소를 위하여 톰캣 서블릿 컨테이너를 제거하고 직접 SOAP 요구와 응답 메시지를 처리하는 시스템을 설계하고 구현하는 방법을 선택한다. 웹 서비스의 성능 향상을 위해 웹 서버에 부가적인 웹 서비스 모듈을 직접 구현한다. 추가하는 모듈은 사용자 요구 수신기, 웹 문서 처리기, SOAP 프로토콜 처리기, NBTM 관리자, 세션 관리자 등이다. 웹 서비스를 직접 구현하기 위하여 크게 SOAP 메시지 핸들링, 메시지 체인, WSDL과 관련된 부분과 쓰레드 풀을 이용하는 논-블록킹 I/O 부분과 클라이언트의 요청 수신 부분을 구현한다. 또한 웹 서비스 구현을 위해 표준의 WSDL 문서와 SOAP 메시지를 사용한다.

[그림 4]는 XML 기반의 웹 서비스를 위해 제안한 모바일 웹 서비스 시스템의 구조를 나타낸다. 모바일 웹 서비스 서버 시스템은 HTTP 연결 처리부를 통해 클라이언트의 웹 서비스 요구를 처리한다. HTTP 연결 처리부는 접속부와 사용자 요구 수신기, 웹 문서 처리기, SOAP 프로토콜 처리기로 구성한다. 웹 서비스의 성능 향상을 위한 추가 모듈은 서버에 구현한다. 메시지 조작, 메시지 체인, WSDL 처리에 대한 기능, 그리고 수신한 SOAP 요구 처리는 서버에서 처리한다. 이를 위해 모바일 웹 서비스 시스템의 구성 요소에 다음과 같은 모듈을 구현한다.

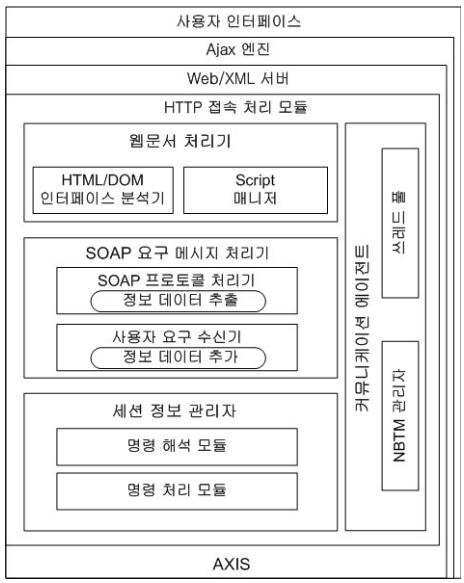

그림 4. 모바일 실험 시스템의 구현 구조도
Figure 4. Implementation Structure of a Mobile Experimental System

사용자 요구 수신기(SOAP 요구 수신기)
웹 문서 처리기(Web Document Processor)
SOAP 프로토콜 처리기(SOAP Protocol Processor)
NBTM 관리자(Non-blocking and thread manager)
세션 정보 관리자(Session information manager)

접속부는 클라이언트 요청을 접수/분석하며, WSDL 요청은 웹 문서 처리기로 전달하고, 웹 서비스 요청은 SOAP 메시지 분석부로 전달한다. 웹 문서 처리기는 특정 스크립트에 대한 WSDL을 생성하고, SOAP 메시지 분석부는 클라이언트 요청을 분석하고, SOAP 메시지 생성부는 서비스 처리부

www.kci.go.kr

의 처리 결과를 SOAP 메시지로 생성한다.

제안한 모바일 웹 서비스 시스템이 웹 서비스 처리를 위한 동작 절차는 다음과 같다.

① 클라이언트는 지정된 URL로 WSDL을 얻는다.
② 클라이언트가 WSDL에 맞는 SOAP 메시지를 모바일 웹 서비스 시스템으로 전송한다.
③ 모바일 웹 서비스 시스템은 클라이언트의 SOAP 메시지를 접수하여 클라이언트가 원하는 WSDL명, 서비스명, 입력값을 추출한다.
④ 추출한 값으로 서비스를 수행한다.
⑤ 서비스 결과를 WSDL 형태의 SOAP 메시지로 변환하여 클라이언트로 전송한다.

세션 유지를 위해 서버는 세션 아이디를 클라이언트에게 전송하고 클라이언트는 세션 아이디 이름을 서버로 전송하여 세션을 복구한다. 서블릿 엔진을 대체하기 위해서 모바일 웹 서비스 시스템과 웹 문서 처리기를 사용한다. 톰캣을 사용하는 대신 WSDL 파일은 웹 문서 처리기에 의해 직접 생성하고 SOAP 메시지는 제안 시스템에서 처리한다.

3.2 웹 서비스 프레임워크의 구성 요소 설계

클라이언트의 요청 메시지는 표준의 SOAP 메시지와 웹 서버로부터의 수신 메시지로 구성하기 때문에 SOAP 메시지 변경을 위해 요청된 SOAP 메시지를 분석하는 모듈과 모바일 웹 서비스 시스템으로부터의 수신 결과를 규격화된 SOAP 메시지로 구현하는 모듈이 필요하다. 따라서 웹 서비스 요청을 수신하는 접속부와 SOAP 메시지 분석기 그리고 SOAP 메시지 생성기로 구성한 사용자 요구 수신기를 구현한다. [그림 5]는 클라이언트가 웹 서비스 형태로 서버에 요청하는 경우의 처리 과정을 나타낸다.

클라이언트는 접근 URL에서 WSDL을 얻고, HTTP를 통해서 SOAP 형태의 XML 문서로 웹 서비스 요청을 위해서 요청 서비스명, 입력값, 세션아이디를 포함하는 SOAP 메시지를 서버로 전송한다. 서버는 클라이언트의 요청 형태가 웹 서비스인 경우는 SOAP 메시지 분석부로 SOAP 메시지를 전달한다. SOAP 메시지 분석부는 요청 서비스의 WSDL명을 추출하고 전달된 SOAP 메시지에서 정보를 생성하고 서비스명과 입력값들을 추출하고 서비스 처리부에 서비스 처리를 요청한다.

서비스 처리부의 처리 결과는 SOAP 메시지 생성부에서 응답 SOAP 메시지를 생성한다. 먼저 응답 헤더 정보, 응답 SOAP 메시지의 정보를 생성하고, 처리 결과가 정상인 경우는 서비스 결과값과 세션 아이디를 추가하고, 비정상이면 실패 정보를 추가한다. SOAP 메시지를 생성하여 접속부로 보내면, 접속부는 클라이언트로 전송한다.

그리고 무선 웹 서비스 통신을 위해서는 WSDL 언어가 필요하다. 기존의 웹에 존재하는 HTML 데이터를 무선 통신 데이터 형식인 WSDL 데이터 형식으로 변환하기 위하여 웹 문서 처리기를 설계한다. 웹 문서 처리기는 클라이언트의 웹 서비스에 대한 요구를 WSDL 파일로 자동으로 생성시키고 클라이언트의 요청에 의해 활성화한다.

```
Procedure Client()
{   call request server(SOAP_Message);
  wait for return of transaction result from server;
        the receipt of a transmission result;
  termination request of web service;  }
Procedure Server()
{     Procedure Connection()
      {   connection_acceptance
          analysis of request a kind;
          If (kind of request = request web service) then
            {  call SOAP_Message_Analysis;
               Wait for result return SOAP message;
               transmission transaction result to Client;  }
       }
      Procedure SOAP_Message_Aalysis()
      {   request analysis;
          an extract of WSDL name;
          creation of SOAP message information
          an extract of service name;
          an extract of input value
          request of service transaction to SOAP_Message_Creation;
          call SOAP_Message_Creation();   }

      Procedure SOAP_Message_Creation()
      {
          creation of response header information
          creation of SOAP message information
            If (transaction result = normal) then
               { an addition of service result value
                  an addition of session ID; }
            else
               an addition of a failure information
            creation of a result SOAP message;
            call Connection()
            transmission transaction result to Connection  }
}
```

그림 5. 클라이언트의 웹 서비스 요청 처리 과정
Figure 5. Web Service Client's Request Process

본 논문에서 제안한 WSDL 생성기는 HTML/XML 파서를 구현하여 WSDL 생성을 지원하고, HTML/XML 파서를 통하여 클라이언트로부터 요청된 메시지를 WSDL로 간단하게 변환한다. 웹 문서 처리기는 Java 1.4의 org.w3c.dom 라이브러리를 이용하여 AXIS에 독립적으로 실행하며, Dom의 Document 객체를 사용하고, 모든 태그는 element 객체를 사용한다.

서비스 처리부는 클라이언트가 요청한 서비스에 대한 정보를 접수하여 서비스를 처리하기 위하여 먼저 클라이언트가 전송한 정보에 세션 정보의 존재를 확인하여 세션 정보가 존재하면 세션을 복구하고, 세션 정보가 존재하지 않으면 세션을 생성하고, 요청한 서비스를 수행하고 결과를 SOAP 메시지

www.kci.go.kr

생성부로 전달한다. 다음의 〔그림 6〕은 제안 시스템의 실행 흐름도를 나타낸다.

세션 정보 관리자는 클라이언트가 접속하면 서버와 클라이언트 간의 접속 상태를 유지하기 위해 필요한 모든 정보를 저장하고 관리하여 세션 복구 요청이 있으면 클라이언트의 세션 복구 요청을 처리하고, 사용하지 않는 세션 정보의 관리와 처리를 담당한다.

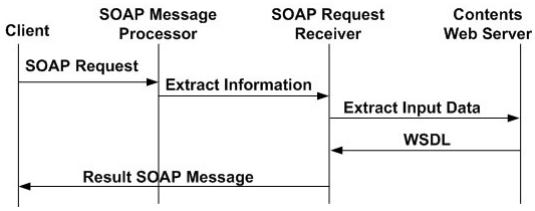

그림 6. 실험 시스템의 실행 흐름도
Figure 6. Execution Flow of the Test System

그리고 NBTM 관리자는 셀렉터에 의해 등록된 서버와 클라이언트의 네트워크 연결에서 발생하는 I/O 이벤트(connect, accept, read, write)를 감지하여 즉각 처리 가능한 I/O 이벤트가 발생한 서버와 클라이언트의 네트워크를 연결하고, 클라이언트의 작업 요청에 적합한 쓰레드를 쓰레드풀에서 선택하며 주기적으로 쓰레드풀을 관리하여 추가/생성을 담당한다.

Ⅳ. 구현 및 평가

4.1 실험 환경

본 논문의 실험을 위해 윈도우 서버와 톰캣 5.5 기반의 웹 서비스 시스템 2개를 구축하고, 제안 시스템인 모바일 웹 서비스 시스템과 비교한다. 실험을 위해서 SOAP 요구 메시지 생성기는 2GB RAM, 2-way Intel XEON 2.4GHz의 클라이언트 4대를 사용한다. 여러 개의 문자 전송과 다수의 동시 요청에 대한 실험을 통해서 웹2.0 웹 서버와 제안한 모바일 시스템의 성능 평가를 실시한다.

모바일 웹 서비스 시스템의 성능 평가를 위해 기존 방식의 웹2.0 웹 서비스 시스템과 본 논문에서 제안한 웹2.0 환경의 실험 시스템을 구성하였다. 각각의 실험 시스템을 위해 2GB RAM, 4-way Intel Xeon 3.0GHz Windows 2000 서버와 Java SDK 1.5 그리고 Tomcat 5.5와 AXIS,IE와 Firefox 브라우저를 사용하였다.

실험을 위한 가정으로 SOAP 요구는 4대의 모바일 클라이언트 시뮬레이션 프로그램에 의해 제안한 모바일 웹 서비스 시스템에 연결하고, 시간 간격은 1초에서 10초이며, 여러 개의 SOAP 요구 메시지를 반복 요청한다. 연결이 종료되면 시뮬레이션 프로그램은 모바일 웹 서비스 시스템에 연결을 위해 반복적인 연결 시도를 한다. 시스템에 접근하는 사용자는 동시에 200명으로 가정하고, SOAP 요구는 4대의 클라이언트에 의해 50개의 쓰레드를 생성시켰다. 본 논문의 실험에 이용된 콘텐트 웹 서버는, 두 웹 사이트(www.naver.com(사전)과 www.yahoo.co.kr(주식)를 선택하였다. 서버 타임아웃은 각각 30초이다.

본 논문의 실험 과정은 〔그림 7〕에 나타난다. 클라이언트가 SOAP 형식으로 웹 서비스를 요청하는 경우, 모바일 웹 서비스 시스템은 SOAP 메시지와 내용을 HTTP 호출로 웹 서버에 전달한다. 웹2.0 서버가 전송한 HTTP 응답을 모바일 웹 서비스 시스템이 수신하면, 제안 시스템은 WSDL을 생성시키고 클라이언트인 모바일 장치에게 SOAP 응답 메시지를 전송한다.

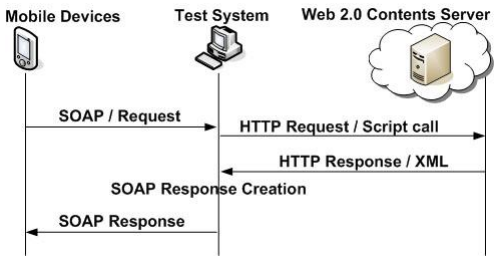

그림 7. 실험 시스템의 사용자 요구처리 과정
Figure 7. User Requests Processes of the Experimental System

4.2 평가

본 논문의 실험은 하나의 연결을 가지고 다수의 문자를 전송한다. 본 논문의 실험 방법은 100개의 문자를 전송할 때 초기 TCP, HTTP 연결을 설정하고 SOAP 메시지로 문자를 전송한다. 수신측은 네트워크 버퍼를 사용한다.

다음의 〔표 1〕은 실험에 대한 요약으로 실험에 사용된 시간과 실험 시간동안 클라이언트의 요청 개수를 나타낸다. 본 논문의 실험은 모바일 웹 서비스 시스템의 실행 평가에 초점을 맞췄다.

표 1. 동시 요청을 위한 실험 정보
Table 1. Test Information for Simultaneous Requests

테스트 정보	제안 시스템	SMSP 시스템	표준 시스템
실험 시간(단위: sec)	2,300	2,500	3,600
총 요구 횟수	41,138	40,121	34,752

www.kci.go.kr

〔그림 8〕은 실험 시스템들의 트랜잭션 효과를 분석한 것으로 실험을 위한 가정은 1초에 50개의 접속을 시도한다. 실험 결과 요구 시간초과의 경우 동시 요청이 10개 이하의 request는 3가지 시스템 모두 제한 시간 내에 요구가 성공하였으며, 50개의 경우 SMSP 시스템과 제안 시스템은 request timeout은 1개 미만의 발생 확률을 갖지만 웹2.0 표준 시스템은 11.3개의 발생 확률을 나타낸다.

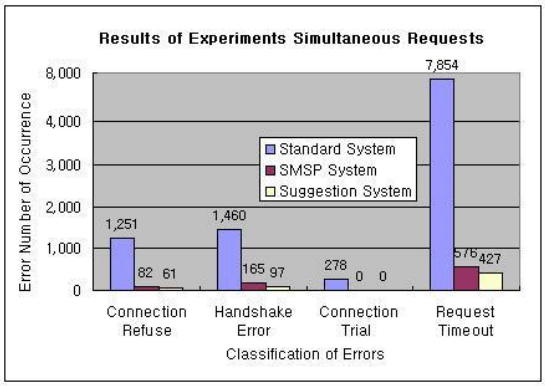

그림 8. 클라이언트 요청에 대한 실험 결과(단위: 개수)
Figure 8. Experiment results for the client requests

〔그림 9〕는 제안 시스템과 표준 시스템에서 클라이언트 개수를 증가시키고 매초 응답 비율을 나타낸다. 〔그림 9〕와 같은 결과는 표준 시스템은 응답 비율이 선형으로 감소시키고 SMSP는 제안 시스템과 비슷한 결과를 나타내다 클라이언트 개수가 증가하면서 감소하지만 제안 시스템은 거의 일정하게 유지한다. 따라서 시스템 성능은 Tomcat을 사용하는 웹 서버보다 좋고 더욱이 클라이언트가 증가됨에 따라 차이가 더 커지는 것은 나타낸다. 제안 시스템의 성능 향상은 non-blocking I/O(NIO) 아키텍처를 사용으로 표준 톰캣 서버보다 더 높은 성능을 제공함을 나타낸다.

본 논문에서 제안한 시스템의 실험 결과는 서블릿 컨테이너를 사용하는 것보다 본 논문의 시스템인 모바일 웹 서비스 시스템의 효율이 높다는 것을 의미하며, 전체 시스템의 성능 향상으로 연계되어 누적되는 연결 신호 처리의 향상을 의미한다. 따라서 연속적인 요청을 빠르게 처리할 수 있고, 성능 차이는 요청이 증가할수록 기존 시스템과 SMSP 시스템은 더 큰 격차를 가져온다. 그러므로 웹 서비스는 시스템의 응답 속도 개선이 전체적인 성능에 많은 영향을 끼치는 것을 의미한다. 결국 연결 요청 횟수가 많아질수록 기존의 웹 서비스 시스템의 오류 횟수는 기하급수적으로 증가한다.

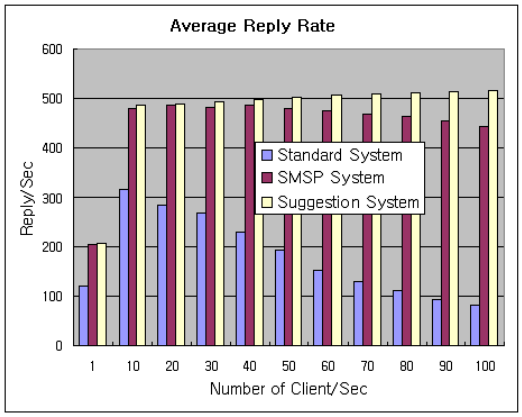

그림 9 제안 시스템과 표준 시스템의 응답 비율
Figure 9. Standard system and the system's response rate proposal

Ⅵ. 결론

본 논문은 변화하는 웹 환경에 맞추어 웹2.0 기반의 웹 서비스에 대한 연구를 진행하였다. 웹 서비스의 성능 향상 문제와 지연 처리 해결이 중요한 이슈가 되고 있는 상황에서 모바일 웹 서비스의 성능 향상과 지연 처리 해결을 위하여 새로운 모바일 웹 서비스 아키텍처를 제안하였다.

본 논문에서 제안한 웹 서비스 아키텍처는 톰캣 서블릿 컨테이너를 제거하여 SOAP에 존재하는 처리 지연을 줄이고 SOAP 메시지를 처리한다. SOAP 요구와 응답은 제안 시스템에 의해 직접 처리된다. 제안한 웹 서버에 추가되는 모듈들은 사용자 요구 수신기, 웹 문서 처리기, SOAP 프로토콜 처리기, 논-블록킹 쓰레드 관리자, 세션 관리자 모듈을 직접 구현하여 추가하였다.

그리고 연결 성공 비율을 증가시켜 개선된 서버 성능의 향상을 위하여 세션에 근거한 표준 작업 부하에서 클라이언트의 세션 정보 관리를 주기적으로 실시하였다. 이 논-블록킹 쓰레드 매니저는 연결 요청의 증가에 따른 연결 오류의 비율을 감소시킨다.

본 논문에서는 웹 서비스 표준 프로토콜을 기반으로 하는 2개 이상의 모바일 웹 서버 아키텍처를 구현하여 일관성 있게 시험했다. 실험 결과 모바일 환경에서 아파치 웹2.0 보다 6.7%, 전통적인 오리지널 아파치 웹 서비스 보다 9.7%의 성능 향상을 얻었다. 또한 동시 요청이 50개 이상의 경우 접속 성능은 약 38% 향상 되었다. 따라서 실험 결과는 접근의 수행을 처리하는 SOAP 요구에서 표준 웹 서비스 구현보다 성능이 향상되었음을 증명하였다. 이 연구는 매우 작은 연결 오류를 가지고 있으므로 전형적인 웹 서비스 구현보다 효율적인

www.kci.go.kr

서비스 요구를 처리할 수 있다. 향후 연구의 중점 사항으로 커뮤니케이션 보안을 위한 메시지 암호화 작업 알고리즘에 관한 연구가 필요하다.

참고문헌

〔1〕 Yoshio Turner, Tim Brecht, Greg Regnier, Vikram Saletore, "Scalable Networking for Next-Generation computing platforms", Proceedings of SAN, 2004.

〔2〕 J. Guitart, V. Beltran, D. Carrera, J. Torres, and E. Ayguad'e. "Characterizing secure dynamic web applications scalability", In 19th International Parallel and Distributed Processing Symposium, Denver, Colorado(USA). April 4-8, 2005, 2005.

〔3〕 김용태, 박길철, 김석수, 이상호, "웹 서비스의 서버 구조 단순화를 통한 웹2.0 웹서비스 성능 향상", 한국정보처리학회 논문지 VOL. 14-D NO. 04. pp. 421-426, 2007.06.

〔4〕 Eric Newcomer and Greg Lomow, Under standing SOA with Web Services, Sddison-Wesey pp. 51-197, 2005.

〔5〕 이규철, "e-비즈니스용 웹 서비스 등록저장소의 안정적 운영을 위한 관리 및 운영지침 개발", 한국전산원, 2005. 12.

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

〔7〕 The Apach Software Foundation, http://tomcat.apache.org/

〔8〕 http://adaptivepath.com/ideas/essays/archives/000385.php

〔9〕 Jesse James Garrett, "Ajax: A New Approach to Web Applications", 2005.

〔10〕 Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Candidate Recommendation, 2006.

〔11〕 Annie P. Foong, Thomas R. Huff, Hervert H. Hum, Jaidev P. Patwardhan, and Greg J. Regnier. "TCP Performance Re-Visited", In Proc. Of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS-2003), pp. 70-79, Austin, Texas, March 2003.

〔12〕 D. Davis and M. Parashar, "Latency Performance of SOAP Implementations", Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 407-412, 2002.〕

저 자 소 개

김 용 태

1988. 숭실대학교 전산학과 석사.
2008년 충북대학교 전산학과 이학박사
2002년-2006년 가림정보기술 이사
2006.3-현재 한남대학교 멀티미디어학부 강의전담교수
〈관심분야〉 모바일 웹서비스, 정보보안, 센서 웹, 모바일 통신 보안, 멀티미디어

정 윤 수

2000. 2. : 충북대학교 전산학과 석사
2008년 2월 : 충북대학교 대학원 전자계산학 이학박사
〈관심분야〉 센서 보안, 암호 이론, Network Security, 이동통신 보안

박 길 철

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

www.kci.go.kr