Mobile Web Service Performance Enhancement by Simplifying Architecture

Soo-Lyul Oh* · Gab-Sang Ryu** · Chul Kim***
Mokpo National University* · Dongshin University** ·
Gwangju National University of Education***

ABSTRACT

In this paper, we suggest performance enhanced Mobile Web service architecture. Mobile Web Services are new technology that integrates different applications to provide interoperability. SOAP is a lightweight message exchanging protocol and is a principal factor that decides the Web Services performance. When a mobile Web Service is implemented with the Java technology, it should be implemented with both a SOAP message processor like AXIS, and Java Servlet container (e.g. Tomcat). This typical implementation is not efficient because it requires an additional communication port and process. In this paper, we suggest a new method that replaces this typical approach to enhance Web Service performance.

Keywords: Mobile Web Service, Performance Evaluation, SOAP, Java Servlet

모바일 웹서비스 성능 향상 처리기 설계

오수열* · 류갑상** · 김 철***
목포대학 컴퓨터공학과* · 동신대학 컴퓨터학과** · 광주교육대학 컴퓨터교육과***

요 약

본 논문에서는 모바일 웹서비스 구현 환경을 최소화하면서 서비스 성능을 향상시키는 개선된 처리기를 설계 제안하였다. 웹 서비스를 구현하기 위해서는 표준에 규정된 WSDL코드를 가지고 SOAP 통신을 해야 한다. 그러므로 WSDL 및 SOAP를 구현하면 기본적인 웹 서비스가 가능하다. 또한 웹 통신을 위해 Tomcat과 AXIS 라이브러리를 사용해야한다. 그러나 Tomcat의 사용은 추가적인 네트워크 사용과 시스템에 프로세스로 상주하여 시스템 자원을 사용하는 문제가 있기 때문에 구현시 오버헤드를 발생시킬 수 있다.

본 논문은 오버헤드의 원인인 Tomcat을 사용하지 않고 모바일 웹 서비스를 구현함으로써 네트워크 자원과 시스템 자원을 절약할 수 있으면서 성능을 향상시키는 효율적 처리기를 설계 구현하였다.

키워드: 모바일 웹 서비스, 성능 평가, SOAP, 자바 서블랫

---

* 교신저자: 류갑상, 동신대학교 컴퓨터학과
- This paper was supported by Research Funds of Mokpo National University in 2009.
논문투고: 2011-03-11
논문심사: 2011-06-13
심사완료: 2011-06-14

1. Introduction

Internet services enable users to access the internet from any location at any time providing flexible personalized information according to users' location and their information needs[14]. The Internet can deliver various value added services in addition to basic communication services. As internet capabilities are widely understood and wireless technologies advance, mobile internet services will soon be a major mediator in information delivery and in business transactions[15]. Furthermore, integrations with other backend applications are required to provide seamless services across the organization. Nowadays XML based Web Service is regarded as a promising solution for this requirement, because it provides seamless interoperability between different computing frameworks such as Microsoft .NET and J2EE solutions and between different computing platforms such as Microsoft Windows, Linux, and others. The Web Service enables faster and cheaper integration between different applications, including third party applications, and also increases application reuse throughout an organization[6].

The Web Service is an alternative of distributed object middleware such as Java RMI (Remote Method Invocation) [9], CORBA (Common Object Request Broker Architecture)[13], and DCOM (Distributed Component Object Model). CORBA, developed by the OMG (Object Management Group), is designed to allow distributed objects to interoperate in different programming languages, such as C++, Java, Smalltalk, and etc. Java Remote Method Invocation (RMI), developed by Microsoft, is a java application programming interface for performing Remote Procedure Calls (RPC). CORBA, Java RMI and DCOM use optimized connection-oriented communications protocols that are either language specific, or have detailed rules defining how data structures and interface should be realized.

2. Background and Related Works

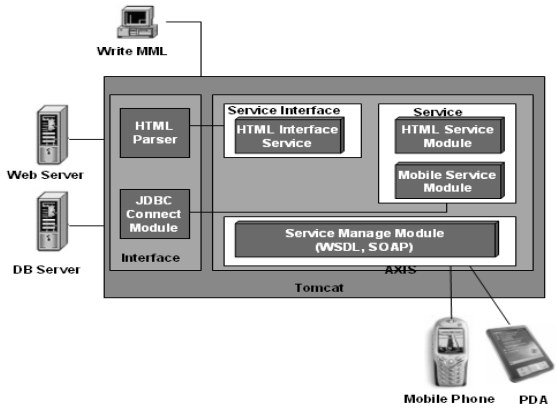
The SOAP and WSDL(Web Service Description Language) are a new messaging protocol and an interface language that supports Web Services[8, 1]. They are increasingly accepted as means of supporting distributed applications on the Web. Therefore, the performance of SOAP becomes a critical issue. Several studies have evaluated the performance of SOAP by comparing it to other technologies like Java RMI[10], CORBA[7] and DCOM[5], and to a different implementation environment like real-time trading system[12], scientific computing[13] and non-trivial peer-to-peer communication[3]. All this research suggests that the performance of Web Service has underperformed compared to other technologies.

compression can reduce the SOAP message length, but real-time gzip compression is computationally expensive. However, XML compression can yield a performance gain if the overhead of the compression algorithm is relatively low compared to XML serialization [17]. Kohlhoff and Steele [11] illustrated that compression can enhance the SOAP performance, but may only be useful in a considerably slower network because of CPU time spent compressing and decompressing. Tian et al. [16] also revealed that compression of mobile Web Service can improve the performance by employing a simple dynamic scheme. Differential deserialization is another method that improves the Web Services performance.

Contrary to these approaches, our method concentrates on the implementation architecture of the Web Service. There are several Web Service implementation methods, which differ in their support for class binding, ease of use and performance [4]. (Figure 1) illustrates an example of the typical Java-based mobile Web Services implementation. SOAP messages are processed by using AXIS (Apache eXtensible Interaction System). AXIS is a SOAP processor developed by the Apache project (http://ws.apache.org/axis/) and supports HTTP

SOAP request/response generation, SOAP message monitoring, dynamic invocation, Web service deployment, and automatic WSDL generation for Web Service. A Web Servlet container, like Tomcat, is required to provide mobile Web Service with AXIS. To provide wireless internet service, the server administrator also needs to write MML (Made Markup Language) to parse Web contents by using the administrative tool. A MML generates service request forms, or service results, by dynamically parsing the existing Web contents and sending them to relevant servers and clients. When a client, whether it is wireless or wired client, requests Web service via SOAP request, Apache Tomcat transfers it to AXIS. AXIS interfaces the SOAP request message into a relevant Web Service server by using the service management function. Service providing servers are interfaced by using a WSDL module, which is provided by AXIS. By implementing SOAP and distributed computing service, the system architecture can have a lightweight thin client structure and the service can be provided in a flexible way.

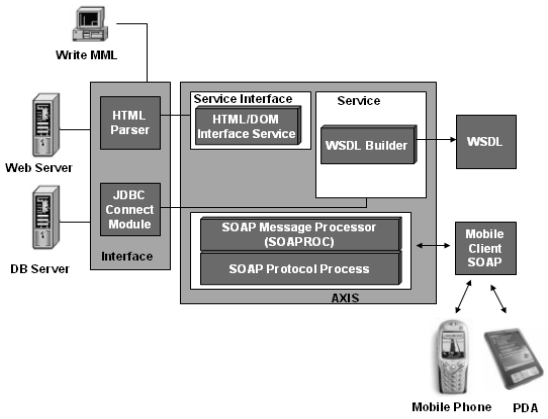
However, this implementation is not efficient because it requires additional process for Web Servlet engine (Tomcat) and the communication port. For this reason, we propose an alternative architecture that can process SOAP messages without using Web Servlet engine.



(Figure 1) Typical Mobile Web Service Implementation

3. Mobile Web Services Architecture without Java Servlet Container

(Figure 2) illustrates our mobile Web Services implementation architecture. The main difference between our approach and the general approach is that our implementation does not use the Tomcat Servlet engine for Web Service processing. Instead, a specialized SOAP processing system, called SOAProc, and WSDL builder are run on the AXIS.



(Figure 2) Mobile Web Services Architecture without Java Servlet Container

4. Implementations

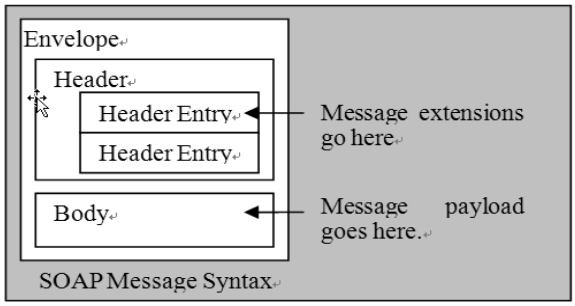
4.1 SOAP Message Structure

A SOAP message consists of the following elements [21]:

- Envelope: The Envelope element serves as a container for the other elements of the SOAP message. As it is the top element, the Envelope is the message.
- Header: The Header element encapsulates extensions to the message format without having to couple it to the payload or to modify the fundamental structure of SOAP. This allows extensions like transactions, encryption, object

references, billing, and others to be added over time without breaking the specification. The Header element is optional and may be eliminated.

- Body: The Body element of a SOAP message is the location for application-specific data. It contains the payload of the message, carrying the data that represent the purpose of the message. It could be a remote procedure call, a purchase order, a style sheet, or any XML which requires to be exchanged using a message.



(Figure 3) SOAP Message Syntax

(Figure 3) illustrates an example of a SOAP message. The Header element is intentionally omitted in this example. <ns1: IntranetLogin …> indicates IntranetLogin method that will be called. The tags between <ns1:IntranetLogin…> tag are parameters of method IntranetLogin, such as <userid> … </userid>, <pass> … </pass>, and <sessionidtag> … </sessionidtag>.

4.2 Analysing SOAP Request Message with SOAProc

The algorithm that analyses the method and its parameters of SOAP request messages are presented in (Figure 4) The algorithm is as follows:

RequestSoapMassage.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv =
"http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<soapenv:Body>
    <ns1:IntranetLogin soapenv:encodingStyle
    =http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns1="urn:wireserver">
        <useridxsi:type="xsd:string">
            test
        </userid>
        <pass xsi:type = "xsd:string">
            pass
        </pass>
        <sessionidtag xsi:type="xsd:string">
            sessionidtag
        </sessionidtag>
    </ns1:IntranetLogin >
</soapenv:Body>
</soapenv:Envelope>
```

(Figure 4) SOAP Message Example

Step 1: Gets the SOAP messages

The system generates a FileInputStream of RequestSoapMessage.xml (fis).

```
FileInputStream fis = new
FileInputStream("RequestSoapMassage.xml")
SOAPEnvelope env = new SOAPEnvelope(fis);
```

Then the system gets the SOAP message from the above FileInputStream.

```
SOAPBodyElement sbe = env.getFirstBody();
```

Step 2: Gets the SOAP Body

The system extracts the SOAP Body from the SOAP message.

```
sbe.getName();
```

Step 3: Analysing SOAP Body

The system finds IntranetLogin part of <ns: IntranetLogin …> from the Body of the SOAP message.

```
ArrayList al = sbe.getChildren();
```

The system creates array list of items between <ns:IntranetLogin…></ ns:IntranetLogin> in the Body of the SOAP message.

```
for(int i = 0 ; i < al.size() ; i++){
   MessageElement me =
MessageElement)(al.get(i));
   System.out.println((i+1)+" th " +
   me.getName()+"'s value is " +
   me.getValue());
}
```

The system iteratively analyses the item list to get MessageElement like <userid> … </userid>, <pass> … </pass>, and <sessionidtag> … </sessionidtag>. In each iteration, the item's name and value are obtained by me.getName() and me.getValue()methods. For example, if the system uses the example in <Figure 5>, <userid … >test </userid> is in the first item of item list and 'userid' and 'test' are the name and value, which can be obtained by using iterative analysis. The system can generate a response message to the clients by using this result.

4.3 Generating SOAP Response Message with SOAProc

Our system analyses the client's SOAP request message and sends the analysing result to the Web server. When the Web server system generates a HTTP response message, our system generates a SOAP response message by using it.

SoapMassageToServer.java

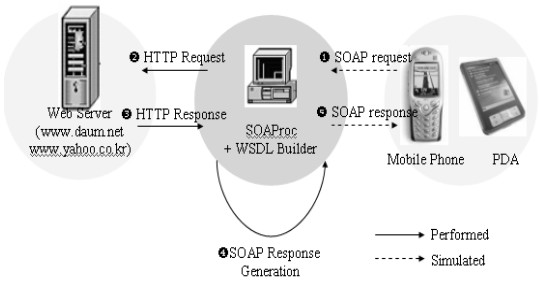
```
import org.apache.axis.message.SOAPEnvelope;
import org.apache.axis.message.SOAPBodyElement;
import org.apache.axis.message.MessageElement;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.ArrayList;
class SoapMassageToServer{
   public static void main(String[] args){
      try{
          FileInputStream fis = new
          FileInputStream("RequestSoapMassage.xml"
          );
          SOAPEnvelope env = new
   SOAPEnvelope(fis);
          system.out.println("soap Meassage");
          system.out.println(env);
          System.out.println();
          SOAPBodyElement sbe =
   env.getFirstBody();
          System.out.println("Service Name : " +
          sbe.getName());
          System.out.println("NamespaceURI : " +
          sbe.getNamespaceURI());
          ArrayList al = sbe.getChildren();
          for(int i = 0 ; i < al.size() ; i++){
             MessageElement me =
             MessageElement)(al.get(i));
             System.out.println((i+1)+" th " +
             me.getName()+"'s value is " +
             me.getValue());
          }
      }
      catch (IOException e){
          System.out.println(e);
      }
      catch (org.xml.sax.SAXException e){
         System.out.println(e);
      }
   }
}
```

(Figure 5) SOAP Message Processing Algorithm

5. Experiment

5.1 Method

The experiment is focused on the performance evaluation of our mobile Web service system. Two sets of systems are prepared for our experiment. The first system is implemented with standard Web Services architecture as explained in Section 2. This implementation requires a Tomcat Servlet container with AXIS. The second implementation is based on our approach, where there is no Servelet container with the SOAP message processing performed by the SOAProc system, and WSDL is created by the WSDL builder.



(Filgure 6) Experiment System Procedure

We conducted a simulated performance comparison experiment. (Figure 6) illustrates the experimental process. If a client requests Web services by submitting a SOAP request, the experimental system analyses the SOAP message and sends a HTTP request to the content Web servers. If the experiment system receives a HTTP response message form the Web server, it generates WSDL and sends a SOAP response message to the client's mobile device.

SOAP requests are simulated by the mobile client simulation program, which connects to the experiment system and sends several SOAP request messages. There are time intervals, from 1 to 10 seconds between SOAP requests. If the connection is closed, the simulation program continually tries to connect to the experimental system. We assumed that there were 200 users on the system at the same time. SOAP requests were created by four client programs and each program generated 50 immediate threads.

The following criteria were examined to compare two experimental systems:

- Test time: the number of seconds consumed for the test.
- Number of Requests: the number of requests generated within test time.
- Connection Timeout: the connection numbers that were not connected within the request timeout.
- Connection Refuse: the request numbers that could not be connected because the server was busy.
- Connection Handshake Error: the number of session configuration failures after connection
- Connection Trial Time: the number of times the client could not connect to the server.
- Request Timeout: the number of times the timeout was exceeded.

5.2 Results

<Table 1> summarizes the experimental results, which illustrate an enhanced performance in all categories. Though the test time of our system is shorter than that of the standard system, the total number of requests is greater than that of standard system and the timeout number is less than that of the standard system. For example, whist the average request per second of our system is 17.94, that of standard system is 9.64. But there are many connection errors in the standard system. Only some portion of 200 requests were successfully connected to the server while the others get a "refused" message from the server. However, those kinds of connection failures do not happen in our system.

<Table 1> Experiment Result

	SOAProc System	Standard System
Test time	29,400	46,200
Total Request	524,573	445,422
Connection Timeout	0	435
Connection Refused	0	18,960
Connection Handshake Error	0	513
Connection Trials	243	22,534

6. Conclusions

Mobile Web Service is a critical solution in the internet service integration architecture. In this research we proposed a new Web Service architecture by implementing two significant systems. Firstly, the HTML/WSDL converter can support reusing current HTML based contents. This is essential for saving

developing or maintenance costs and serving seamless internet services both wired and wireless. Secondly, we proposed a new SOAP message processing system to diminish SOAP latency problems by eliminating the Tomcat Servelet container in the Web Services implementation. The SOAP request and response messages are directly processed by the SOAProc system.

We can implement an alternative mobile Web Services system by using these two systems without violating standard Web Services protocols. Our experimental results demonstrate that the SOAP request processing performance of our approach is significantly better than that of the standard Web service implementation. Our system can process more service requests and is more efficient than that of typical Web service implantation while exhibiting only very small connection errors.

References

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

[2] Chiu, K., M. Govindaraju, and R. Bramley (2002), Investigating the limits of SOAP performance for scientific computing. Proceedings 11th IEEE International Symposium on High Performance Distributed Computing, 246-254.

[3] Cutsem, T.V., et al (2004), On the Performance of SOAP in a Non-Trivial Peer-to-Peer Experiment. in 2nd International Conference of Component Deployment.

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

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

[6] Doculabs (2003), @Bench Web Services Performance Benchmark Study, oculabs: LaSalle, Chicago.

[7] Elfing, R., U. Paulsson, and L. Lundberg (2002), Performance of SOAP in Web Services Environment Compared to CORBA. in 9th Aisa-Pacific Software Engineering Conference (APSEC '02).

[8] Gudgin, M., et al. (2003), SOAP Version 1.2., W3C.

[9] JavaSoft (1998), Java Remote Method Invocation Specification, revision 1.5, JDK 1.2 edition.

[10] Juric, M.B., et al. (2004), Java RMI, RMI tunneling and Web services comparison and performance analysis. SIGPLAN Notices, 39-5; 58-65.

[11] Kohlhoff, C. and R. Steele (2004), Evaluating SOAP for high performance applications in capital markets. Computer Systems Science and Engineering, 19-4, 241-251.

[12] Kohlhoff, C. and R. Steele (2003), Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems. in WWW2003. Budapest,Hungary.

[13] OMG (1995), Common Object Request Broker Archictecture.

[14] Siau, K., E.P. Lim, and Z. Shen (2000), Mobilecommerce: Journal of Database Management,. 12-3, 4-13.

[15] Senn, J.A. (2000), The emergence of m-commerce. Computer, 33-12, 148-150.

[16] Tian, M., et al. (2004), Performance considerations for mobile Web services. Computer Communications, 27-11. 1097-1105.

[17] van Engelen (2003), R.A. Pushing the SOAP Envelope with Web Services for Scientific Computing. in the International Conference on Web Services (ICWS). LasVegas.

[18] W3C (2003), SOAP Version 1.2. SOAP Version 1.2,

Author

Soo Lyul Oh

1981. B.S. Department of Materials Engineering, Chonnam University
1986. M.S. Department of Computer Science, Chosun University
1994. Ph. D. Department of Computer Science, Chonnam University
1988~Present. Professor Department of Computer Engineering, Mokpo University
Area of Interest: software engineering, web service, cluding computing
e-mail: syoh@mokpo.ac.kr

Gab Sang Ryu

1983. B.S. Department of Computer Science, Chonnam University
1985. M.S. Department of Computer Science, Chonnam University
2006. Ph. D. Department of Computer Science and Engineering, Korea University
1995~2005. Research Engineer, Korea Institude of Machine and Metal
1996~Present. Professor Department of Computer Science, Dongshin University
Area of Interest: wireless mobile Internet, ICT education. CAD/CAM
e-mail: gsryu@dsu.ac.kr

Chul Kim

1997. Ph. D. Department of Computer Science, Chunnam University
1998. Guest Professor, University of Washington
1992~Present. Professor Department of Computer Education, Gwangju National University of Education
Area of Interest: internet resource management, education contents, e-Learning
e-mail: chkim@gnue.ac.kr