# MWSMF: A Mediation Framework Realizing Scalable Mobile Web Service Provisioning

Satish Narayana Srirama [1], Matthias Jarke [1,2]
[1] RWTH Aachen, Informatik V
Ahornstrasse 55,
52056 Aachen, Germany
{srirama, jarke}@cs.rwth-aachen.de

Wolfgang Prinz [1,2]
[2] Fraunhofer FIT
Schloss Birlinghoven
53754 Sankt Augustin, Germany
wolfgang.prinz@fit.fraunhofer.de

## ABSTRACT

It is now feasible to invoke basic web services on a smart phone due to the advances in wireless devices and mobile communication technologies. While mobile web service clients are common these days, we have studied the scope of providing web services from smart phones. Although the applications possible with Mobile Host are quite welcoming, the scalability of such a Mobile Host is observed to be considerably low. In the scalability analysis of our Mobile Host, we have observed that binary compression of SOAP messages being exchanged over cellular network have greatly improved the performance of the Mobile Host. While binary compression is observed to be very efficient, the mechanism has raised the need for an intermediary in the mobile web service invocation cycle. The paper addresses our mobile web service message optimization scenario, at the enterprise service bus technology based mediation framework, with evaluation results.

## 1. INTRODUCTION

From the view-point of information systems engineering, the Internet has lead the evolution from static content to web services. Web services are software components that can be accessed over the Internet using well established web mechanisms, XML-based open standards and transport protocols such as SOAP and HTTP. Public interfaces of web services are defined and described using Web Service Description Language (WSDL), regardless of their platforms, implementation details. Web services have wide range of applications and primarily used for integration of different organizations. The biggest advantage of web services technology lies in its simplicity in expression, communication and servicing. The componentized architecture of web services also makes them reusable, thereby reducing the development time and costs. [10]

Concurrently, the capabilities of high-end mobile phones and PDAs have increased significantly, both in terms of processing powers and memory capabilities. The smart phones are becoming pervasive and are being used in wide range of applications like location based services, mobile banking services, ubiquitous computing etc. The market capture of such smart phones is quite evident. The higher data transmission rates achieved in wireless domains with 3G and 4G technologies and the fast creeping of all-ip broadband based mobile networks also boosted this growth in the cellular market. The situation brings out a large scope and demand for software applications for such high-end smart phones.

To meet this demand of the cellular domain and to reap the benefits of the fast growing web services domain and standards, the scope of the mobile terminals as both web services clients and providers is being observed. While mobile web service clients are common these days [7, 4], we have studied the scope of *mobile web service provisioning* in one of our previous projects. In this project, we have developed a *Mobile Host* [19], capable of providing basic web services from smart phones. Extensive performance analysis of the Mobile Host was conducted and many applications were designed and developed proving the feasibility of concept.

While the applications possible with Mobile Host are quite welcoming, during the performance analysis of the Mobile Host, we have observed the scalability of the Mobile Host to be considerably low. We define *scalability* as the Mobile Host's ability to process reasonable number of clients, over long durations, without failure and without seriously impeding normal functioning of the smart phone for the user. For improving the scalability of the Mobile Host the mobile web service messages exchanged over the radio link are to be compressed without seriously affecting their interoperability. Different compression techniques are studied in detail and BinXML [8] was identified to be the best possible compression option and the binary encoding was adapted for the mobile web service invocation cycle.

While BinXML is observed to be very efficient for the mobile web service message compression, the encoding mechanism has raised the need for an intermediary or a middleware framework in the mobile web service invocation cycle. The mediation framework should encode/decode the mobile web service messages to/from XML/BinXML formats in the mobile operator proprietary networks. During our Mobile Host's application, QoS and discovery research, we

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MOBILWARE 2008, February 13-15, Innsbruck, Austria
Copyright © 2008 ICST 978-1-59593-984-5
DOI 10.4108/ICST.MOBILWARE2008.2797

have identified the final deployment scenario of Mobile Hosts in the cellular networks. The Mobile Web Services Mediation Framework (MWSMF) is established as an intermediary between the web service clients and the Mobile Hosts based on the Enterprise Service Bus (ESB) technology. The features, realization details and performance analysis of the mediation framework, specific to maintaining scalability of the Mobile Host are addressed in this paper. The rest of the paper is organized as follows:

Section 2 discusses the concept of mobile web service provisioning. Section 3 addresses scalability analysis of the Mobile Host. Section 4 discusses the components and realization details of the mobile web services mediation framework, while section 5 provides the evaluation of the MWSMF. Section 6 summarizes the results and concludes the paper.

## 2. MOBILE WEB SERVICE PROVISIONING

Service Oriented Architecture (SOA) [6] is a component model that delivers application functionality as services to end-user applications and other services, bringing the benefits of loose coupling and encapsulation to the enterprise application integration. SOA is not a new notion and many technologies like CORBA and DCOM at least partly represent this idea. Web services are newest of these developments and by far the best means of achieving SOA. Using web services for SOA provides certain advantages over other technologies. Specifically, web services are based on a set of still evolving, though well-defined W3C standards that allow much more than just defining interfaces.

The quest for enabling these open XML web service interfaces and standardized protocols also on the radio link, with the latest developments in cellular domain, lead to a new domain of applications, mobile web services. The developments in cellular world are two folded; firstly there is a significant improvement in device capabilities like better memory and processing power and secondly with the latest developments in mobile communication technologies with 3G and 4G technologies, higher data transmission rates in the order of few mbs were achieved. In the mobile web services domain, the resource constrained mobile devices are used as both web service clients and providers, still preserving the basic web services architecture in the wireless environments. While mobile web service clients are quite common these days [7, 4], the research with providing web services from smart phones is still sparse. In our *mobile web service provisioning* project one such Mobile Host was developed proving the feasibility of concept [19]. Figure 1 shows the deployment scenario of mobile web services, where mobile devices are used as both web service providers and clients.

Mobile Host is a light weight web service provider built for resource constrained devices like cellular phones. It has been developed as a web service handler built on top of a normal Web server. The SOAP based web service requests sent by HTTP tunneling are diverted and handled by the web service handler component. The Mobile Host was developed in PersonalJava on a SonyEricsson P800 smart phone. The footprint of the fully functional prototype is only 130 KB. Open source kSOAP2 [12] was used for creating and handling the SOAP messages.

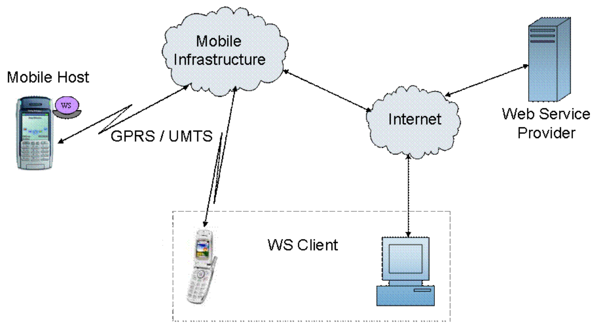

**Figure 1: Mobile terminals as web service providers and clients**

The detailed evaluation of this Mobile Host clearly showed that service delivery as well as service administration can be done with reasonable ergonomic quality by normal mobile phone users. As the most important result, it turns out that the total web service processing time at the Mobile Host is only a small fraction of the total request-response time (<10%) and rest all being transmission delay. This makes the performance of the Mobile Host directly proportional to achievable higher data transmission rates. Similarly the regression analysis of the Mobile Host showed that the Mobile Host can handle up to 8 concurrent requests for reasonable services of message sizes approximately 2 kb. Mobile Host is also possible with other Java variants like J2ME, for smart phones. We also have developed a J2ME based Mobile Host and its performance was observed to be not so significantly different from that of the PersonalJava version. The J2ME based implementation is used in analyzing the scalability of the Mobile Host.

Mobile Host opens up a new set of applications and it finds its use in many domains like mobile community support, collaborative learning, social systems and etc. Primarily, the smart phone can act as a multi-user device without additional manual effort on part of the mobile carrier. Many applications were developed and demonstrated, for example in a distress call; the mobile terminal could provide a geographical description of its location (as pictures) along with location details. Another interesting application scenario involves the smooth co-ordination between journalists and their respective organizations. Most recently the scope of the Mobile Host in m-learning (mobile learning) domain is also being studied with applications like podcasting, mobile blogging, expertise finders and etc. The Mobile Host in a cellular domain is of significant use in any scenario which requires polling that exchanges significant amount of data with a standard server, for example a mobile checking for the updates of RSS feeds provided by a server. The Mobile Host can eliminate polling process as the RSS feeds can now be directly sent to the Mobile Host, when the RSS feeds updated [19].

## 3. SCALABILITY ANALYSIS OF MOBILE WEB SERVICE PROVISIONING

While the applications possible with Mobile Host are quite welcoming, during the performance analysis of the Mobile Host, we have observed the scalability of the Mobile Host to be considerably low. From the regression analysis of Mobile Host for checking its scalability, Mobile Host was successful in handling 8 concurrent accesses for reasonable service like *location data provisioning service* with response size of approximately 2Kb. The main reason for not being able to process more mobile web service clients was due to the transmission delay which constituted 90% of the mobile web service invocation cycle time. Similar analysis conducted with the *mobile picture service*, where the response size is approximately 40 kb, further supported this point. So the Mobile Host's scalability is inversely proportional to increased transmission delays. The transmission delays can be reduced in two ways. 1.) By achieving higher data transmission rates with current generation telecommunication technologies. 2.) By reducing the size of the message. In the scalability analysis of the Mobile Host, we mainly concentrated at the second issue i.e. reducing the size of the message being transmitted over the radio link.

Web services communication is a layered communication and across different protocols. Considering SOAP over HTTP, at the lowest level is the transportation protocol, TCP. On top of TCP lies the HTTP communication. Then SOAP communication is over the HTTP protocol. The application communication and protocols for example WS-Security lies on top of SOAP. So any message exchanged over the web service communication, consists some overhead across all the different layers. Since we have considered wireless environments, and the message exchange is over the cellular network, the size of the message has to be reduced to the minimum possible level [13]. The size of the mobile web service message is shown in equation 1.

$$B_{msg} = B_{tp} + B_{mtp} + B_{soap} + B_{app} \tag{1}$$

Where $B_{tp}$, $B_{mtp}$, $B_{soap}$, $B_{app}$ are the message overheads over transportation, message transportation, SOAP, application protocols respectively. So to exchange the messages effectively over the radio link $B_{msg}$ has to be minimized. For this the messages are to be compressed/encoded in the optimal way. The minimal encoding may not always be the best solution. First reason for this is that the encoding should be efficient, both in terms of message size reduced and extra performance penalties added to the devices. For example if the size of message is reduced by 50% and the processing of the encoding takes more than half the time of actual message exchange cycle, the encoding mechanism is not efficient. Secondly the encoding mechanism should not affect the interoperability. If an attempt is made to reduce the overload at $B_{tp}$ or $B_{mtp}$, the interoperability of the web services is seriously impeded. So the best position to target the encoding process is at the $B_{soap}$ and upper levels. So the XML based SOAP messages are to be compressed.

Web service messages can be compressed with standard compression techniques like *Gzip* or XML-specific compression techniques like *XMill* to obtain smaller message sizes. Canonical XML [5] standard targets the logical equivalence of these compressed XML messages. Recently there is an effort with the Fast Web Services [15], Fast Infoset standard draft [16], Efficient XML [1], BinXML [8] etc. to specify a binary format for XML data that is an efficient alternative to XML in resource constrained environments. Most recently there is also some effort with BiM (Binary Format for Metadata) [11] standard for the binary encoding of MPEG-7 Metadata. BiM is designed in a way that it allows fast parsing and filtering of the XML data at the binary level itself, without having to decompress again. [9] gives a comparison of different compression technologies for XML data and specifies the best scenario for the web service message exchange across smart phones. The analysis suggests that BinXML is the best option (BiM was not considered in this analysis) to compress web service messages considering compression ratio, processing time and resource usage.

Based on the analysis at [9] we have adapted BinXML for compressing the mobile web service messages. BinXML is a very simple binary representation of XML data. It replaces each tag and attribute with a unique byte value and replaces each end tag with 0xFF. By using a state machine and 6 special byte values including 0xFF, any XML data with circa 245 tags can be represented in this format. The approach is specifically designed to target SOAP messages across radio links. So the mobile web service messages are exchanged in the BinXML format in radio link and the approach has improved the performance of the Mobile Host significantly.

To analyze the effects of BinXML compression for mobile web services, we have used two Sony Ericsson P990i smart phones as web service requestor and the Mobile Host. The phones have an internal flash shared memory of 64 Mb. The devices support MIDP2.0 with CLDC1.1 configuration. The two mobile phones were connected to the Internet using GPRS connections. The services of the *Expertise Finder Modules* are deployed on the smart phones and the mobile web service clients tried to invoke these services. *Expertise Finder scenario* is developed with the Mobile Host and is used in m-learning domain. Using the Expertise Finder Modules, learners can collaborate among each other to find an expert, who can solve a specific problem. Once the expert is found, the expert can rate himself to the client, using the *expert rating service*. The Expert Rating service is used in the scalability analysis of the Mobile Host. The SOAP request message of this service includes the actual expert finder request message, the intermediaries to whom the message is forwarded before reaching the expert, and the rating and details of the expert. The size of the request message is observed to be 2544 bytes, with 4 forwards. The response just shows the acknowledgement from the client, and its size is 570 bytes. With the BinXML encoding, the size of the message to be transmitted over the radio link has reduced significantly. The size of the request reduced to 1591 bytes while the size of the response reduced to 495 bytes. This reduction in size of the messages to be transmitted has caused significant reduction in transmission delays of the request and response messages.

The actual gain in mobile web service invocation cycle time, i.e. mobile web services compression gain with the BinXML encoding is:

$$T_{mwscg} = \delta T_{reqt} + \delta T_{rest} - T_{reqenc} - T_{reqdec} - T_{resenc} - T_{resdec} \tag{2}$$

Where $T_{reqenc}$ and $T_{reqdec}$ are the encoding and decoding delays of the SOAP request message, while $T_{resenc}$ and $T_{resdec}$ are the encoding and decoding delays of the SOAP response

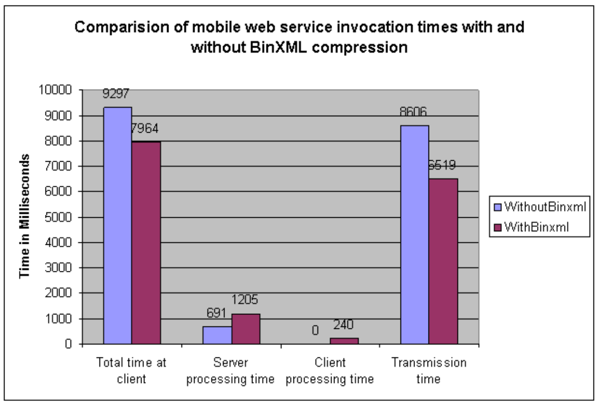

**Figure 2: Comparison of timestamps for the expert rating service with and without BinXML encoding**

message, and $\delta T_{reqt}$, $\delta T_{rest}$ are the respective reductions in request transmission and response transmission delays.

Figure 2 shows the comparison of delays in mobile web service invocation cycle, with and without BinXML encoding. From this diagram we can derive that there is approximately 1333 milliseconds ($\sim$15%) gain in performance of the Mobile Host with the BinXML encoding. We could also conclude that the performance gain is directly proportional to the compression gain achieved with the binary encoding. Alternative compression mechanisms can also be verified for the Mobile Host's performance gain, with the architecture we have proposed in this study. As long as $T_{mwscg}$ value is positive, the compression mechanism is efficient.

But BinXML is not an open standard. Hence not all the messages transmitted over the radio link can be based on this standard. If a client sends an uncompressed message for the Mobile Host, the transmission is not very efficient, even though the Mobile Host can process such a request. In such a scenario a valid mediation framework should be established at the gateway of the mobile operator proprietary networks, helping in encoding/decoding the mobile web service messages to/from XML/BinXML formats sent into the cellular domain.

## 4. MOBILE WEB SERVICES MEDIATION FRAMEWORK (MWSMF)

The Mobile Web Services Mediation Framework (MWSMF) is established as an intermediary between the web service clients and the Mobile Hosts. The mobile web service clients in the Internet can thus invoke the services deployed with the Mobile Host, via the MWSMF. The architecture and the features of the MWSMF are addressed in [18]. The detailed architecture of the MWSMF includes many Peer to Peer (P2P) concepts and thus beyond the scope of this paper. A perusal of the deployment scenario is recommended for clear understanding of the framework, though the scalability concepts discussed further in this paper does not require such clarity. The mediation framework ensures the QoS of the mobile web service messages and transforms them as and when necessary and routes the messages based on their content to the respective Mobile Hosts. Apart from handling security and improvements to scalability the QoS provisioning features of MWSMF also includes message persistence, guaranteed delivery, failure handling and transaction support. Enterprise Service Bus (ESB) is the most recent development in enterprise integration domain and a standards-based ESB solves the integration problems elevated by the MWSMF. Gartner et al. defines enterprise service bus as a new architecture that exploits web services, messaging middleware, intelligent routing, and transformation [17]. We tried to realize the MWSMF based on ESB technology and implemented the middleware framework using Java Business Integration (JBI) based open source ServiceMix ESB.

ServiceMix by following the JBI architecture supports two types of components - *Service Engine Components* and *Binding Components*. Service engines are components responsible for implementing business logic and they can be service providers/consumers. The binding components marshall and unmarshall messages to and from protocol-specific data formats to normalized messages. Thus they allow the JBI environment to process only *normalized messages*. The normalized message consists of the message content also called payload, message properties or metadata and optional message attachments referenced by the payload. No two components in the framework communicate directly and the messages are exchanged over the *Normalized Message Router (NMR)*. The components are deployed into the framework using spring based XML configuration file. Spring is a lightweight container, with wrappers that make it easy to use many different services and frameworks. Lightweight containers accept any Java Bean, instead of specific types of components [20]. The configuration uses the WS-Addressing for routing the messages across the components via the normalized message router. WS-Addressing is a specification of transport-neutral mechanisms that allow web services to communicate addressing information. It essentially consists of two parts: a structure for communicating a reference to a web service endpoint, and a set of Message Addressing Properties, which associate addressing information with a particular message.

### 4.1 Components developed for scalability maintenance of the Mobile Host

The following components are developed at the mediation framework for improving the scalability of the Mobile Hosts.

***HttpReceiver***: The HttpReceiver component receives the web service requests (SOAP over HTTP) over a specific port and forwards them to the Broker component via NMR. The component thus acts as the gateway to the mediation framework and as a proxy for the Mobile Hosts. The component is configured into the mediation framework by adding the following xml chunk to the configuration file.

```
<sm:activationSpec componentName="httpReceiver" ser-
vice="ssn:httpBinding" endpoint="httpReceiver" des-
tinationService="ssn:mwsmfBroker">
<sm:component>
<bean class="org.apache.servicemix.components.http.
HttpConnector">
<property name="host" value="localhost"/>
<property name="port" value="8912"/>
</bean>
```

```
</sm:component>
</sm:activationSpec>
```

The `componentName` attribute of the `activationSpec` specifies the name of the component, while `service` indicates the service name of the proxied endpoint, the `endpoint` specifies the endpoint name of the proxied endpoint and the `destinationService` attribute specifies the service name of the target endpoint, here it is the Broker component. The remaining properties of the bean component are specified in the `<sm:component>` element.

The support for this component is most recently deprecated in ServiceMix and ServiceMix ESB now ships with a JBI compliant HTTP/SOAP binding component named `servicemix-http`. The component can be used as both a requester and a provider [2]. The component also has the support for WS-Addressing specification. But as most of the message flow analyses of the MWSMF were conducted by considering the HttpReceiver component, here I still go with the old architecture. ServiceMix is an open source project and currently under rigorous development and hence the modification to the old components and support for new components are quite obvious and regular.

***HttpInvoker***: The binding component generates a web server request, if the message is a normal HTTP request. The component can also invoke web services by transferring the SOAP messages as HTTP body. The component is developed by us, while evaluating the mediation framework. As discussed already and similar to HttpReceiver component, the HttpInvoker component can theoretically be replaced by servicemix-http binding component.

***Broker***: This component serves as a hub for all the communication that happens with other components, in the mediation framework. It receives the client-supplied message from the HttpInvoker component and hosts the main integration logic of the mediation framework. In case of the scalability maintenance, the messages received by Broker are verified for mobile web service/BinXML messages. The component also interfaces with other components and provides the result to the client.

***BinaryTransformer***: The service engine component transforms the mobile web services messages to and from the BinXML format. If the request message is in the XML format it encodes the message to BinXML format and decodes to XML if it receives a BinXML message. The component always acts as a provider and it receives the messages from the Broker.

Apart from the components discussed above, the MWSMF also provides many alternative components that help in providing QoS and discovery for mobile web services deployed with the Mobile Host. For example QoSVerifier, XSLT-Transformer help in maintaining the security of the mobile web services. The mediation framework also provides features like helping in automatic startup of the Mobile Hosts conserving the resources of smart phones and UDDI server for publishing the mobile web services. Further details of the MWSMF are provided at [18].

### 4.2 Mobile web service message optimization

Once the MWSMF is established, for improving the scalability of the Mobile Hosts, the components discussed in the previous subsection are deployed with the mediation framework. The messages received by the mediation framework from external clients are compressed using BinXML encoding, and the binary messages are sent over the radio link. The scenario is named as *mobile web service message optimization*. The message flow of the scenario is shown and numbered in figure 3. The service engine components are shown as straight lined rectangles, while the binding components are shown as dashed rectangles.

1. The HttpReceiver component receives the mobile web service request or the HTTP Request from the client.
2. The HttpReceiver sends the message to the Broker through the NMR
3. The Broker examines the message for web service request and transfers the message to HttpInvoker, if it is a normal HTTP request. The Broker transfers the message to the BinaryTransformer component through the NMR, if the message comprises a mobile web service request.
4. The BinaryTransformer component BinXML encodes the message and transfers the response message back to the Broker component.
5. The Broker component sends the message to the HttpInvoker via the NMR.
6. The HttpInvoker generates the request to the Mobile Host by setting the BinXML data to the body of the HTTP message.
7. The Mobile Host processes the request using a BinXML adapter and sends the response message back to the HttpInvoker.
8. The HttpInvoker sends the response back to the Broker via the NMR.
9. The Broker transfers the response to the BinaryTransformer through the NMR.
10. The BinaryTransformer decodes the BinXML data to the XML format and transfers the response message back to the Broker component.
11. The Broker returns the response to the HttpReceiver component through the NMR.
12. The HttpReceiver component returns the response back to the client.

## 5. EVALUATION OF THE MWSMF

Once the mobile web service message optimization was designed and established, the MWSMF was extensively tested for its performance and scalability issues, using load test principles. A huge number of clients were generated for the mediation framework, simulating real-time mobile operator network load. The expert rating service considered in the scalability analysis of the Mobile Host is again considered for the performance evaluation of the MWSMF.

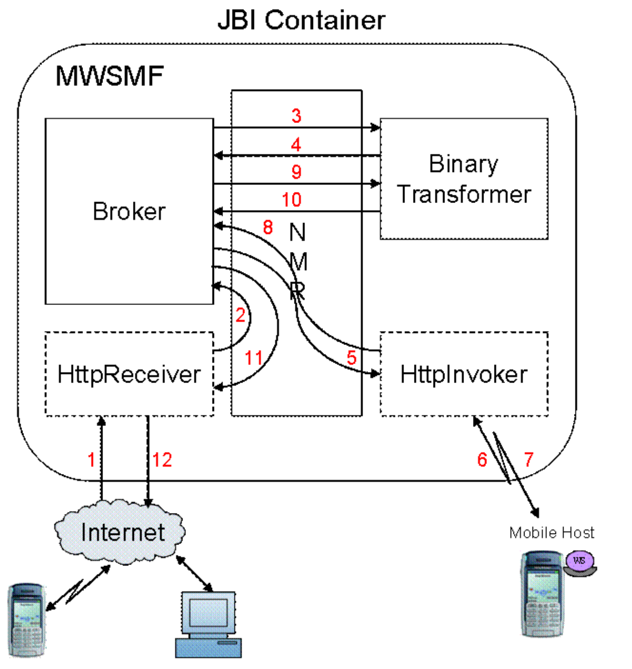

**Figure 3: Message flows in mobile web service message optimization scenario**

## 5.1 Test setup

The prototype of the ServiceMix based mediation framework is established on a HP Compaq nw8240 laptop. The laptop has an Intel(R) Pentium(R) M Processor 2.00GHz / 1GB RAM. A java based server was developed and run on the same laptop on an arbitrary port (4444), mocking the Mobile Host. The server receives the expert rating service request from the client and populates the standard response. The response is then BinXML encoded and the compressed response is sent back to the client, in the HTTP response message format. By considering this simple server, we can eliminate the pure performance delays of the Mobile Host and the transmission delays of the radio link, and thus getting the actual performance analysis of the MWSMF. For the load generation we used a Java clone of the popular ApacheBench load generator from WSO2 ESB [3, 21]. The load generator can initiate a large number of concurrent web service invocations simulating multiple parallel clients. The command line executable `benchmark.jar` also provides a detailed statistics of the invocations, like the number of concurrent request, successful transactions per second, mean of the client invocation times etc. The `benchmark.jar` and `commons-cli-1.0.jar` are downloaded to a working directory and are used to simulate huge number of concurrent requests. The following sample shows a command that simulates 200 concurrent clients for the expert rating service with each client generating 10 requests for the same service.

```
java -jar benchmark.jar -p ExpertRatingRequest.xml -
n 10 -c 200 -H "SOAPAction:  urn:expertRating" -T "
text/xml; charset=UTF-8" http://localhost:8912/soap
/Service
```

## 5.2 Test results

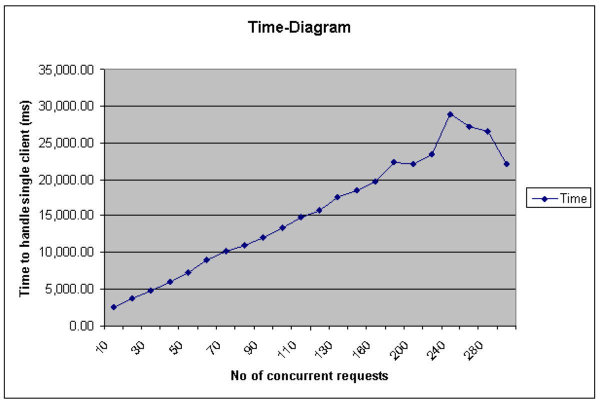

**Figure 4: Average times taken to handle clients under different concurrency levels, at the mediation framework**

Figure 4 shows the time taken for handling a client request under multiple concurrent requests generated for the mediation framework. The mediation framework was successful in handling up to 110 concurrent requests without any connection refusals. Higher numbers of concurrent requests were also possible, but some of the requests failed as the mediation framework generated '`connection refused IO error`'. The main reason for this connection refusal is as: The ServiceMix transport is based on blocking code which means that the ESB can handle only as many concurrent requests as the number of threads configured in the system [14]. Figure 4 also shows a steady increase in the average time taken for handling a client request with the increase in number of concurrent requests. The figure also shows a sharp decline in the time taken to handle a client after 240 concurrent requests. The decline is because of a large number of failed requests at this concurrency level. More than 300 concurrent requests are not considered as already at this high concurrency level, the number of failed requests is more than 50% of the total requests. The increase in average duration to handle a client is quite normal and the mean duration of handling a single request still remains mostly constant. The mean is calculated considering the performance of the MWSMF over long durations, including parameters like the number of service requests failed. The mean value is in the range 100-150 milliseconds, and it improved slightly with the increase in concurrency levels. This shows the performance of mediation framework is actually improving when there are large numbers of clients to handle.

The results from this analysis show that the mediation framework has reasonable levels of performance and the MWSMF can scale to handling large number of concurrent clients, possible in the deployment scenario addressed in [18]. This conclusion is also evident from figure 5, which shows that the number of transactions handled by the mediation framework per second almost remains steady (in fact growing) even under such heavy load conditions. The mediation framework is successful in handling 6-8 mobile web service invocations per second, with the mobile web service message optimization scenario. The values can significantly grow, when the deployment scenario is established on reasonable servers, with high resource and performance capabilities.

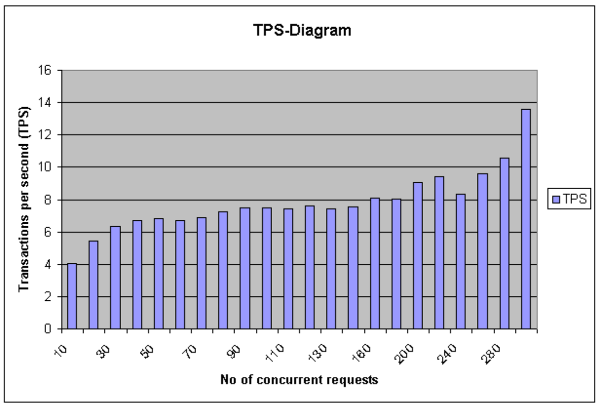

**Figure 5: The number of transactions handled per second by the MWSMF at different concurrency levels**

## 6. CONCLUSIONS

This paper addressed our scalability analysis of the Mobile Host, which has identified that Mobile Host's scalability is inversely proportional to increased transmission delays. In order to reduce the transmission delays, we tried to reduce the size of the mobile web service messages being exchanged, using BinXML encoding mechanism. The performance gain to the Mobile Host with binary compression is quite significant in terms of improved scalability. The study also raised the necessity for an intermediary in the mobile web service invocation cycle. We have developed an enterprise service bus technology based mobile web services mediation framework, acting as a proxy for mobile web service provisioning. The paper also addressed the features, components and realization details of the MWSMF. The regression analysis of the mediation framework conducted with the mobile web service message optimization scenario, clearly showed that the mediation framework has reasonable levels of performance and the MWSMF can scale to handling large number of concurrent clients, possible in mobile operator networks.

## 7. ACKNOWLEDGMENTS

This work is supported by the Research Cluster Ultra High-Speed Mobile Information and Communication (UMIC) at RWTH Aachen University (http://www.umic.rwth-aachen.de/). The authors would also like to thank R. Levenshteyn and M. Gerdes of Ericsson Research for their help and support.

## 8. REFERENCES

[1] AgileDelta. Efficient XML: Lightning-Fast Delivery of XML to More Devices in More Locations, 2007.

[2] Apache ServiceMix. Apache servicemix 3.x users' guide. Apache ServiceMix community, 2007.

[3] Apache Software Foundation. Apache benchmark, 2007.

[4] B. Benatallah and Z. Maamar. Introduction to the special issue on m-services. *IEEE transactions on systems, man, and cybernetics - part a: systems and humans*, 33(6):665–666, November 2003.

[5] J. Boyer. Canonical xml, version 1.0. Technical report, W3C Recommendation, March 2001.

[6] S. Burbeck. The Tao of e-business services - The evolution of Web applications into service-oriented components with Web services. IBM DeveloperWorks, October 2000.

[7] J. Ellis and M. Young. J2ME Web Services 1.0 - Final Draft (JSR 172). Technical Report 11, Sun Microsystems, Inc., October 2003.

[8] M. Ericsson. A study of compression of XML-based messaging. Technical report, Växjö University, 2003.

[9] M. Ericsson and R. Levenshteyn. On optimization of XML-based messaging. In *Second Nordic Conference on Web Services (NCWS 2003)*, pages 167–179, November 2003.

[10] K. Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to web services architecture. *IBM Systems Journal: New Developments in Web Services and E-commerce*, 41(2):178–198, 2002.

[11] J. Heuer, C. Thienot, and M. Wollborn. *Introduction to MPEG-7: Multimedia Content Description Interface*, chapter Binary Format, pages 61–80. Jon Wiley and Son, 2002.

[12] kSOAP2. kSOAP2 - An efficient, lean, Java SOAP library for constrained devices. SourceForge.net.

[13] M. Laukkanen and H. Helin. Web services in wireless networks: What happened to the performance. In *proceedings of the Int. Conf. on Web Services (ICWS '03)*, pages 278–284. CSREA Press., 2003.

[14] A. Perera. Wso2 esb performance testing round 2, July 2007.

[15] P. Sandoz, S. Pericas-Geertsen, K. Kawaguchi, M. Hadley, and E. Pelegri-Llopart. Fast Web Services, August 2003.

[16] P. Sandoz, A. Triglia, and S. Pericas-Geertsen. Fast Infoset, June 2004.

[17] R. W. Schulte. The Enterprise Service Bus: Communication Backbone for SOA. Gartner Inc., May 2007.

[18] S. N. Srirama, M. Jarke, and W. Prinz. A mediation framework for mobile web service provisioning. *2006 Middleware for Web Services (MWS 2006) Workshop, 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (edocw)*, 0:14, 2006.

[19] S. N. Srirama, M. Jarke, and W. Prinz. Mobile web service provisioning. In *AICT-ICIW '06: Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, page 120. IEEE Computer Society, 2006.

[20] B. A. Tate and J. Gehtland. *Spring: A Developer's Notebook*. O'Reilly, April 2005.

[21] WSO2. Wso2 esb benchmark, 2007.