A Hybrid Paradigm Aggregating Web-Service and Mobile Agent

Hatem A. Khater, Mohamed El Kholy,
A. Baith Mohamed, Magdy Saeb

Arab Academy for Science, Technology & Maritime Transport
Alexandria, Egypt

Abstract: One main goal of new Software communities is to achieve the vision of ubiquitous knowledge in highly dynamic and heterogeneous environments. Software communities have different concepts but they all share some common goals such as flexibility, autonomy and software reusability. But they differ in some other goals such as mobile agent which lead to faster applications and reduced network bandwidth demands. Another promising community is a web service which leads to a platform independent environment via the messaging system. In this paper we aim to integrate the web service with the mobile agent .This integration will allow us to use the benefit of the platform independence of the web service paradigm with bandwidth reduction of the mobile agent .In other words we tried to utilize the mobile agent technology to implement an enhanced web service.

Key-Words: Web service, Mobile agent, network traffic Universal Description, Discovery and Integration (UDDI), EXtensible Markup Language (XML).

1. Introduction

The web service is a software system designed to support interoperable machine-to-machine interaction over a network. It supports a hardware and language independent where the service provider can write his code in any language and uses a web server software to translate it to a web service which is a platform independent and can be consumed by any client without constrains any language he is using.

Web service mainly relies on the messaging system in which a service consumer "client" sends a request then the service provider "server" replies with a response. This messaging system increases the network traffic especially when these

messages carry a lot number of bytes. Nevertheless the searching mechanism to explore and find the required service may delay the agent which wants to rapidly find and consume this service. On the other hand, the mobile agent is a software paradigm in which the program code can migrate from one host to another over the network. It carries its data and execution status to complete its task away from its original host autonomously without any interaction from its home or visited host. Therefore this code transfer and client side execution reduces the total network traffic because there is no need to transfer data between the server and the client. In turn, this will have another important benefit which is the ability of mobile agent applications to overcome network latency. This latency increases by increasing the network traffic especially when we have a mobile host that is needed to forward the network packets for it while this host changes its network address continuously. Nevertheless, mobile agents can provide enhancement to all applications which need parallel computing such as search which may improve the web service discovery. However, the mobile agent requires a common platform to interact with hosts which they visit during their journey to execute their tasks or to communicate with each other's .We aim to combine the fast execution, bandwidth demand and parallel search with the platform independent environment that is integrating the web service and the mobile agent.

The remaining of the paper is organized as follows section 2 introduces the web service and its architecture.

Received 01/23/2012 Reviewed 02/20/2012

Section 3 the mobile agent, section 4 motivation scenario. Section 5 the integration between the web service and the mobile agent, section 6 integrating web services with mobile agent. Section 7 experimental results while section 8 is our conclusion.

2. Web service

Service oriented architecture (SOC) is a new software paradigm in which the basic construct is the service instead of the object in the object oriented paradigm. It aims to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. Web service can implement a service-oriented architecture. It provides functional building-blocks accessible over standard Internet protocols independent of platforms and programming languages. The basic Web services platform is XML (EXtensible Markup Language) and HTTP (Hyper Text Markup Language). XML provides a language independent which can be used between different platforms and programming languages and still expresses complex messages and functions. The HTTP protocol is the most used Internet protocols and is machine independent. When all major platforms could access the Web using Web browsers, different platforms could interact. For these platforms to work together, Web-applications were developed.
Main web service elements are:

- SOAP (Simple Object Access Protocol)
- WSDL (Web Services Description Language)
- UDDI (Universal Description, Discovery and Integration)

UDDI allows the client to search the service .Then SOAP will bind him to the provider to exchange data according to the WSDL description.

2.1 SOAP (Simple Object Access Protocol)

SOAP is a framework for the exchange of XML documents in which the client sends a request to a server and its input data and then the server responds with the result of this service and so on. It is the main messaging system between the service provider and service consumer. Also the SOAP engine which is used at the server and client sides facilitates:
Serializing objects from a programming language into SOAP messages and De-serializing SOAP messages into objects in a programming language.

2.2 WSDL (Web Service Description Language)

It is the interface of a Web Service which is defined using the XML format. The interface is independent of the Web Service implementation. A WSDL document defines the one or several operations, and their messages (received by the operation and returned back). WSDL documents consist of the following parts WSDL types, WSDL message, WSDL port type, WSDL binding, WSDL service.

1 WSDL types

This can be considered as the declaration of the WSDL document including the document schema and all the input and output type declaration.

2 WSDL message

This indicates all messages that are transferred between client and provider “SOAP in” and the other direction from the provider back to the client “SOAP out.

3 WSDL port type

This contains one or more operation. Each Operation specifies an action supported by the service.

4 WSDL binding

This provides the mapping from WSDL to SOAP of each service. This is done by defining each SOAP action.

5 WSDL service

This contains the service name and the location of this service. WSDL must provide the user with all what he must know about the service in other words a client does not need any knowledge about the Web Service besides what he can find in the WSDL.

2.3 UDDI (Universal Discovery and Direction Interface)

UDDI provides a method for publishing and finding service descriptions. The service description information defined in WSDL is complementary to the information found in a UDDI registry. The provider uses it to publish his service to a broker on the other hand the client uses it to search for the desired service. UDDI searches for the service by its semantic meaning and its key words when the service is finding the replay is the WSDL of the service which enables the client to interact with the service.

2.4 Web service mechanism

The provider will write his service by any language and any platform and the web server program will convert his program to a web service by creating its WSDL and then the provider will publish his service via a broker , a consumer wants to use this service so he will search for it using UDDI and when this service is found the broker will send him back the WSDL of the service which enables the consumer to interact with this service , during run time messaging system SOAP is responsible of carrying XML message between the provider and consumer containing different elements and results.

3. Mobile agents

Mobile agent is a software paradigm in which the program code can migrate from one host to another across a network. It performs its task locally on the client side on machines that provide agent hosting. Mobile agents are intelligent enough to take the decision of migrating autonomously without any interaction of the original host or any visited one. MA includes creation, initialization, migration, monitoring, deletion, communication with other agents and termination. A Mobile Agent System (MAS) must be able to support all the above functionalities.

3.1 Mobility

For a process is to migrate from one host to another, it needs the following things which allow this migration.

Common execution language

If the mobile agent migrates from one host to another, both hosts must share a common execution language. This is easy to be found in a homogenous networking environment. In this case, assembly language or machine code could be sent across the network for execution. Another scenario for mobile agency is a heterogeneous environment, where many different system architectures are connected. To solve this problem, an interpreted scripting language is used .Therefore the migration of the mobile agent still has restrictions and must be with a host which has the capability to interact with this agent.

Communication mechanism

Some communication mechanism must exist to transfer agents across a network. An agent might be transferred in the network layer using TCP/IP, or by using a higher level of communication such as HTTP. Note that in web service SOAP which is XML message is transferred using HTTP.

Security

In this environment where the code can transfer to any host in the network. We have to secure the host against malicious code and also protect the code against malicious hosts.

3.2 Mobile agent goals

Mobile agent allows us to achieve many goals such as bandwidth conservation which became a very important demand especially in wireless connection. It also allows delegating tasks to agents when not connected which over come poor network connections. It helps us to increase the computing efficiency by using parallel computing.

Bandwidth conservation

One of the main goals of mobile agents is to conserve network bandwidth; this is done by placing the code directly at the point of information, rather than sending many messages across the network.

Delegate tasks to agents when not connected

Mobile agent has the ability of Off-line computing which enables it to work remotely without an uninterruptable network connection to the user, because after the mobile agent is dispatched from his original it is able to fulfill his task without any connection to his original host until it migrates back to it to deliver the result avoiding the problems caused by poor networking conditions.

Enabling a new type of interaction

Mobile agent has the ability of self-cloning which allows it to increase the parallelism of a task by making clones visit numerous data sources simultaneously. This property is very important in searching mechanism. In this case the mobile agent produces a number of agents having the same code and features to visit every host to find the desired data or service.

4. Motivation scenario

In the traditional web service, the service is delivered in the client / server model. The server provides a set of operations that the client can invoke. After the client searches for his service, he starts the SOAP messaging with the provider to invoke the service. These messages are the data transferred from the client then the result is sent back from the provider to the client. One can consider if the data send by the client includes a large number of bytes and the client invokes this service simultaneously. This condition will badly increase the network traffic and the problem will grow more if this client is connected via a wireless connection which has a narrow bandwidth. It may be unstable for one reason or another which will lead to a number of data losses or errors in sent data. This will force the node to resend its data again over the same narrow bandwidth. Another more high-risk condition takes place if this client is mobile and changes his address continuously. At the same time one wants to want to forward the results of his invoked data and continue receiving his SOAP messages including his data to be processed. The widespread use of wireless communication devices such as laptop, PDA and cell phones makes this scenario often more likely to take place. Therefore if we can deliver the service its self to the client and allow it to access the client data locally .We will then reduce the execution time of sending SOAP messages or resending it. We also decrease the network traffic by sending only the code once to the client. Moreover, this will overcomes network disconnection because there is no need for the user to still connect with the provider. This performance of this web service is enhanced depending on the size of data transferred. There is another important scenario motivating us to move the web service code towards the client if the client. This is the case if the client data have high security requirements and the client does not want to transfer his data over the open environment of the internet.

5. An enhanced web service based on mobile agent

Mobile agents can provide an enhanced implementing of web services by decreasing execution time, reduce bandwidth requirements and overcome the effects of unstable network connections. Also mobile agents are free to move between cooperating web servers to implement the desired service functionality by performing a web service composition. This may lead to the new perspective on cloud computing in which the mobile agent will search and compose all the desired functionality required by the client. Then these services are delivered as only one complete service to the client. In addition mobile agent carries the data and execution status if the execution condition requires him to complete his task at another host. Implementing web services with mobile agents needs careful discussion of many issues. For example, the code and data mobility is exclusively a mobile agent-related issue. On the other hand data transfer in the form of SOAP messages is exclusively a Web service-related issue. SOAP messages are XML document which transfer over HTTP, the mobile agent also utilizes HTTP to travel from one host to another. We will trace the web service in the implementation discussion step by step describing the role of the mobile agent in each step.

5.1 Service implementation and description

In web service this code is translated to web service by producing its WSDL. This in turn describes the service as a set of operations over a messaging system at a network endpoint which performs the functionality of the service. In the mobile agent paradigm the code is pushed towards the client and carrying its data and execution status .To integrate the web service and the mobile agent paradigm, this code must   provide its service description. That is done by producing the WSDL describing all operations and functions. The mobile agent code is translated to be language and platform independent.

5.2 Mapping web service request to mobile agent

At the provider side there is a main issue which is how to assign an incoming web service request to a particular mobile agent that fulfills that request. We have two main approaches to do this. The first is to use a one-to-one mapping between services and mobile agents. In this case all requests to the same service are delivered to one mobile agent. This approach will have a complex problem when a service request is sent will the mobile agent is away from this endpoint. The problem is how to push this service to the location of the mobile agent and what about the states of this mobile agent at this time. The second solution is to generate a new mobile agent for each incoming request and limit the lifetime of a particular mobile agent to the time that the requested tasks end. So on the provider side there will be a number of copies of this mobile agent according to the frequency of this service demand. When one mobile agent is dispatched from the server another copy will be produced. However, it is a simple model because multiple, concurrent requests can easily be handled by generating more mobile agents. In this approach we guarantee the availability of the web service at any time under the condition of multiple request, it is a fact that many users may request the same service at the same time.

5.3 Service discovery

In web service the main searching mechanism based on UDDI which is compared to yellow pages on the internet UDDI allows service providers to list themselves by name, product, location, or the web services they offer. When a client wants to run a particular service he search the internet using UDDI including the service name or his desired SOAP interaction or some of this service profiles this is sent to a broker which has a UDDI registry in which services information from multiple service providers is stored and when service searching ends the result is the WSDL of the service to be sent back to the client or a not available annotation if the desired service is not found at this broker. This mechanism can be enhanced by using mobile agent which can has cloning behavior allowing faster search because multiple copies of mobile agents carrying the same UDDI will search this service at multiple brokers at the same time. Mobile agents can perform a more important task which is service composition in which the mobile agent will visit more than one provider to compose the required service.

6. Integrating web service with mobile agent

Our method of the web service migration will be valuable if we decreased the amount of data transferred and the execution time. This will be true if the code size of the mobile agent is less than the size of SOAP messages transferred between provider and client. We will discuss our approach from the beginning when a client needs a service until this service is executed and we will test our execution time using SOAPUI program. In step one the client will search for the service .In step two the client will invoke the service and the provider will replay with a mobile agent carrying this service.

6.1 Service search

The first point a user needs a service to complete his software implementation is performed by defining this service and starting his search using traditional UDDI. This client is connected to our interface which will create a mobile agent to carry this UDDI. This mobile agent starts the search, with its cloning ability, reducing the search time and the number of messages between different brokers and the client. The first mobile agent which finds the service will carry the result back to the client that is the WSDL file of this service. This process is shown in Figure1.

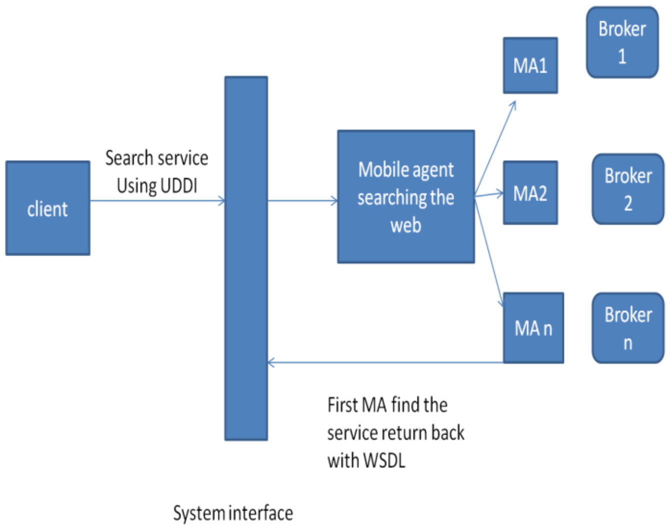

client
Search service Using UDDI
Mobile agent searching the web
MA1
MA2
MA n
Broker 1
Broker 2
Broker n
First MA find the service return back with WSDL
System interface

Figure 1: Service search using mobile agent

The incoming UDDI from the client will put its required data in the mobile agent which is designed to search the web using the client data mobile agent can speed up the service search by using his cloning features to perform a parallel search, the first mobile agent finds the required service at any broker will return to the client with the service description WSDL. Note that mobile agent can also perform service composition if the service is not completely implemented at only one provider, but in this paper we focus on the performance of the web service when it is found or composed at one server, and discuss the performance of this single hope although our

method can extend to another hopes the mobile agent can perform to complete his task if he needs so.
This method of service discovery will be allowed only for brokers who can run mobile agents.

6.2 Service invocation

The client now needs to invoke his required service so he sends the SOAP request to the service provider .The provider instead of sending his SOAP respond back to the client and start a SOAP conversation with the client consuming the network bandwidth by many messages. The provider instead passes this request to his mobile agent engine which is responsible for mapping this request to a mobile agent. This mobile agent which satisfies this request will be dispatched and send back to the client. The mobile agent interacts with the client at its side and locally accesses its data as shown in Figure 2 until it carries out the required task.

6.3 Service migration

The code is sent over the HTTP protocol just as SOAP message which ensure that we will utilize the internet infrastructure just like the traditional web service. The implementation of service migration is not difficult to achieve and there are many ideas of realizing it. There are two main questions that must be resolved to implement the service migration. The first is how to transfer the service from one machine to another. The second is how to receive this service at the client side and interact with it. The web service provider must be supported with a function that is able to transfer an object over the TCP connection. We consider the service here as an object which can be transferred over the HTTP protocol. Many languages such as JAVA and .NET is capable to transfer objects via various networks. There are other ways to transfer an object over the network using a TCP Connection. Regarding the second question, the client should be aware of the fact that it receives such an object “service”. Therefore, it must be supported with a function to start this object upon receiving it. The mobile agent is transferred to the client who already has the WSDL of this service .The mobile agent starts execution accessing the data from the client in its same form SOAP messages put locally from the port these messages was to be sent away. In this way there is no different from the traditional web service mechanism.

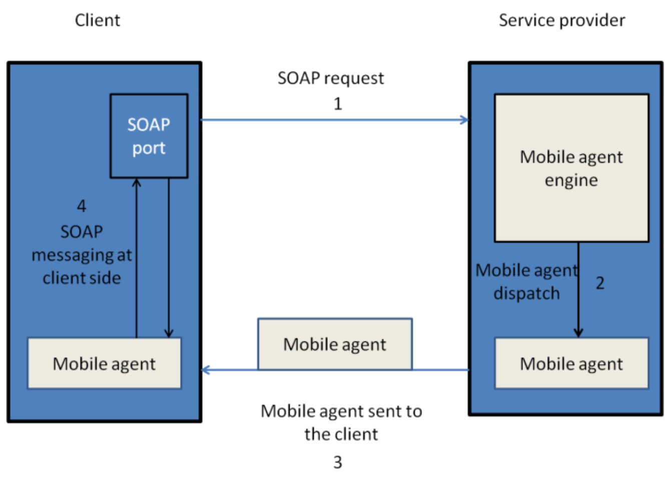


Figure 2: Service provider sends mobile agents to the client side and access client data locally

As we have mentioned before, this paper focuses on the only single hope for the mobile agent to access the data at the client side but this service may be composed from more than one provider or the mobile agent after executing the data on the client side needs to visit another provider to complete his task. We will discuss a simple example where a client needs to perform a service which is already implemented at the provider side or the provider composed it according the client requirements. Assume that this service needs a huge amount of data which is located at the client side. An example of this service is matrix multiplication which is a common program in digital signal processing and image processing and many other application. Assume that we have two matrices each 50*50 so we have 2500 elements in each matrix each element may be complex type with a big number of bytes. We have to transfer 5000 elements of complex type and not only once the client may want to use this service simultaneously for a period of time. The problem is not in the matrix multiplication or addition which will not take a lot of time the problem is transferring the data from the client to the provider via SOAP messages. These messages will take a lot of time and increase the network traffic. Another bad condition if this client uses a wireless connection which is unstable. This situation will suffer from a narrow bandwidth as well as simultaneous disconnection which will force us to resend the data from client to provider or in the opposite direction, instead of this we can use our method to send this service to the client side and access this data locally at the client side.

6.3 Drawbacks of our method

Our system will not be efficient if the total number of bytes transferred in the traditional web service is less than the code transferred. Thus, our system is advantageous to be used in applications where a large chunk of data is located at the client side. Another drawback an take place if we send a program to the client side which may be a malicious code. So we must support the client with security against probable attacks.

7. Experimental results and comparison

We have verified our result by implementing this matrix multiplication as a web service by using .Net programming language and applying it to a simulated client. This client will invoke the service and we will calculate the result using SOAPUI. We here have two parameters to test the first is the execution time and the second is the total number of bytes transferred. Then will calculate the execution time if this service is transferred as a mobile agent to the client side and access the data locally. We implemented the service to multiply two 2*2 matrices with elements of type real and calculate the response time and total SOAP bytes transferred. We got the following results the total number of bytes transferred was 549 bytes and the execution time was 10 m sec. We repeated this test for 10 times the total number of bytes was between (530-550) bytes and the execution time was (8-10) m sec.

Then we implemented another web service to multiply two 3*3 matrices the result after 10 times was (11-17) m sec and the total number of bytes was 748 bytes. There is a snapshot of these results in the appendix of this paper in FigureA.1, FigureA.2 (Appendix). We now can see how the SOAP messages increase with increasing the matrix element.

Then we calculate the execution time of this service if it is delivered as a mobile agent who accesses the data locally on the client side. The result was as follows: 2*2 matrix less than 1 m sec, 3*3 matrix 1m sec, 50*50 matrix 7 m sec, 70*70 matrix 9 m sec which is the same as 2*2 matrix in the traditional web service.

The size of the code which will be transferred was only 6000 bytes for the matrix multiplication program which is written in .Net programming language. This is less than the total bytes of 3*3 matrix multiplication for only ten times.

From these results we recognized that our method can help to decrease network traffic as well as decreasing the execution time. This method may help real time and wireless applications.

8. Summary and Conclusion

We have introduced an unexampled paradigm which combines the universal interoperability features of the web services with the bandwidth conservation features of the mobile agent. We have used existing infrastructure to enhance the performance of the web services and conserving network resources. We have shown that our system can be implemented by using the same underlying communication protocol. Our method utilizes the HTTP to move the mobile agent to the client side. In addition, the same protocol was used to transfer web service messages. We have examined two parameters to demonstrate the resulting performance enhancement. This enhancement resulted in reduction of the execution time and the total size of data transferred. We have shown that the total execution time of traditional web service to multiply two 2 by 2 matrices were 8 to 10 msec. By repeating this calculation with our method the execution time was less than 1 m sec. In addition, the method provided a reduction in the size of data transferred to 6 Kbytes for any matrix size for any number of the multiplication operation. Without the proposed technique, the traditional web service consumes 550 bytes for multiplying two 2 by 2 matrices for and 740 bytes to multiply two 3 by 3 matrices every time the operation is used. The idea behind this fact is that the code is transferred only once and can be executed several times. The proposed paradigm may prove its efficiency in real-time applications such as in the case of multimedia and image processing techniques.

Based on the above argument, we conclude that our paradigm reduces network traffic and decreases the execution time.

References

[1] Dominic Cooney and Paul Roe, "Mobile Agents Make for Flexible Web Services," The Ninth Australian World Wide Web Conference, Queensland, Australia, July 2003.

[2] Mydhili K.Nair1 and V. Gopalakrishna. "Applying web services with Mobile Agent for Computer Network Management," International Journal of Computer Networks & Communications (IJCNC), March 2011.

[3] Magdy Saeb, Cherine Fathy, " Performance Evaluation of Mobile Agent-based Dynamic Load Balancing Algorithm, " 9th International Conference on Distributed Multimedia Systems, DMS_Conference, Miami, Florida, USA, 2003.

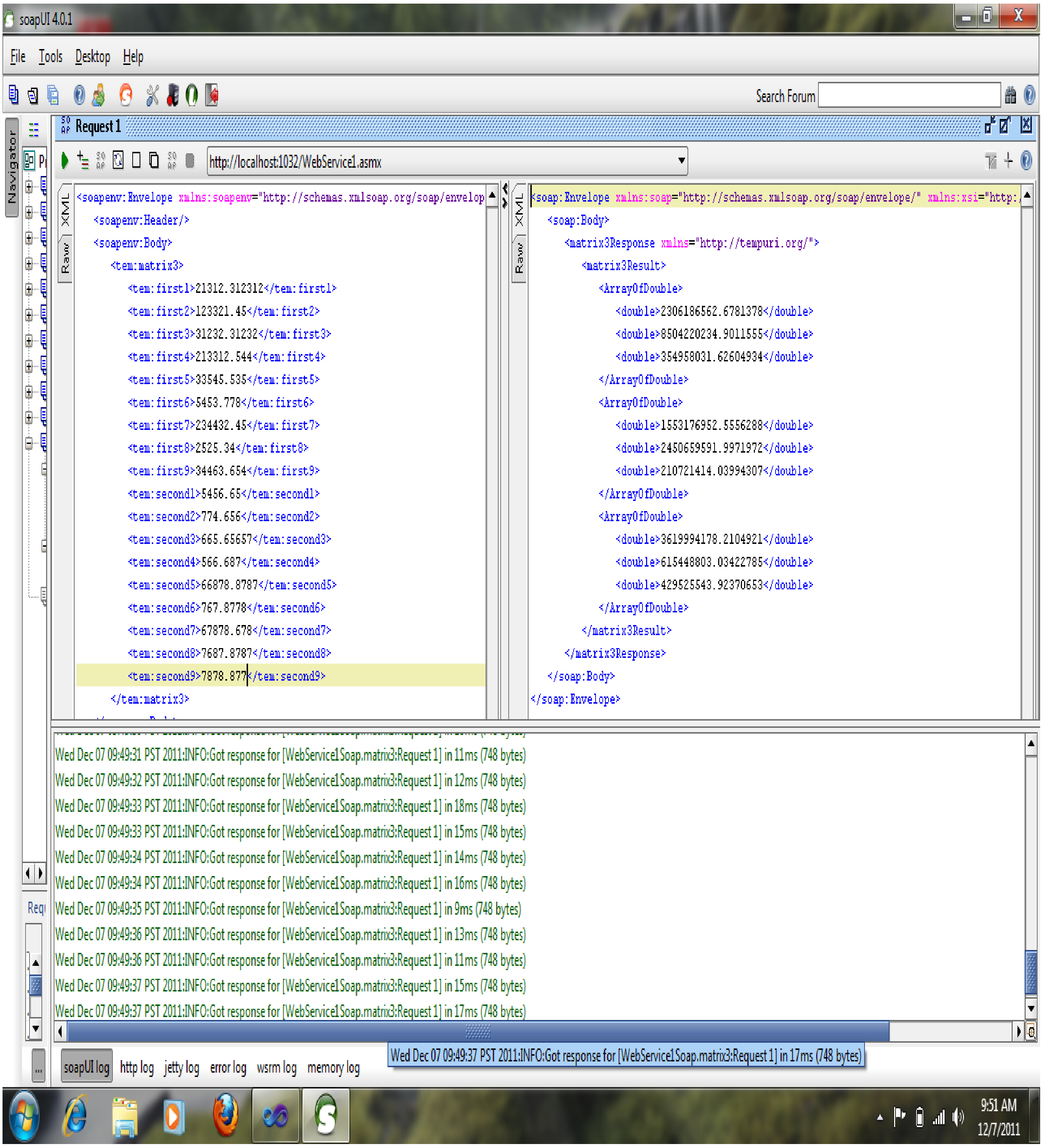
[4] Amir Padovitz, Shonali Krishnaswamy, and Seng Wai Loke, "Toward Efficient and Smart Selection of Web Service," In AAMAS'2003 Workshop on Web Services and Agent-based Engineering, 2003.

[5] R. Montanari and G. Tonti, "A policy-based infrastructure for the dynamic control of agent mobility," In 3rd International Workshop on Policies for Distributed Systems 2002.

[6] Beda Christoph Hammerschmidt , Volker Linnemann,
" MIGRATING STATEFUL WEB SERVICES Using APACHE AXIS and P2P," IADIS International association for development of the information society , 2005.

[7] World Wide Web Consortium W3C " Simple Object Access Protocol " (SOAP) http://www.w3.org/tr/SOAP . Site accessed on December 2011.

Appendix:
Images of execution of the SOAP UI program



```
http://localhost:1032/WebService1.asmx
```

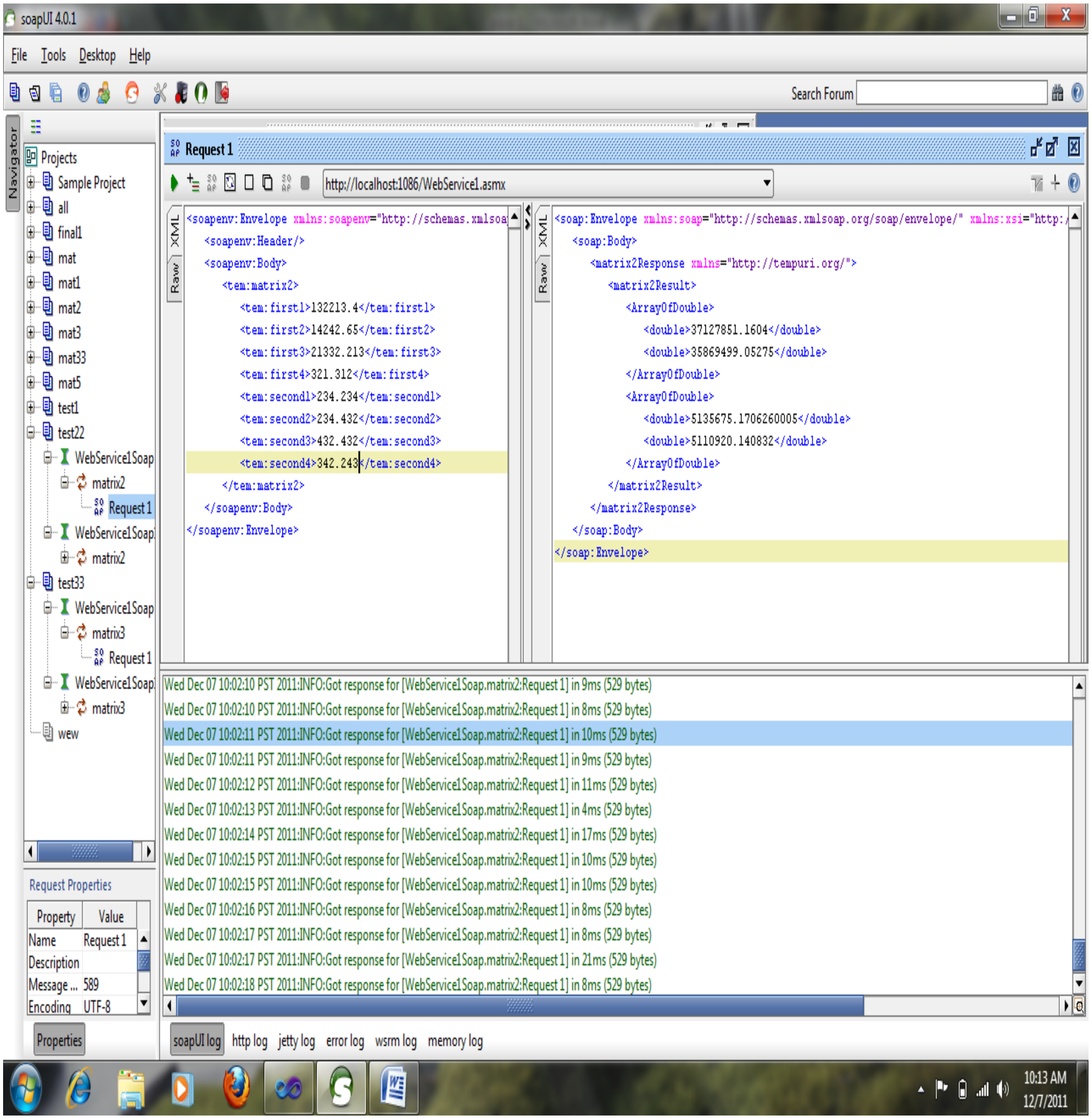
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelop
   <soapenv:Header/>
   <soapenv:Body>
      <tem:matrix3>
         <tem:first1>21312.312312</tem:first1>
         <tem:first2>123321.45</tem:first2>
         <tem:first3>31232.31232</tem:first3>
         <tem:first4>213312.544</tem:first4>
         <tem:first5>33545.535</tem:first5>
         <tem:first6>5453.778</tem:first6>
         <tem:first7>234432.45</tem:first7>
         <tem:first8>2525.34</tem:first8>
         <tem:first9>34463.654</tem:first9>
         <tem:second1>5456.65</tem:second1>
         <tem:second2>774.656</tem:second2>
         <tem:second3>665.65657</tem:second3>
         <tem:second4>566.687</tem:second4>
         <tem:second5>66878.8787</tem:second5>
         <tem:second6>767.8778</tem:second6>
         <tem:second7>67878.678</tem:second7>
         <tem:second8>7687.8787</tem:second8>
         <tem:second9>7878.877</tem:second9>
      </tem:matrix3>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http:/
   <soap:Body>
      <matrix3Response xmlns="http://tempuri.org/">
         <matrix3Result>
            <ArrayOfDouble>
               <double>2306186562.6781378</double>
               <double>8504220234.9011555</double>
               <double>354958031.62604934</double>
            </ArrayOfDouble>
            <ArrayOfDouble>
               <double>1553176952.5556288</double>
               <double>2450659591.9971972</double>
               <double>210721414.03994307</double>
            </ArrayOfDouble>
            <ArrayOfDouble>
               <double>3619994178.2104921</double>
               <double>615448803.03422785</double>
               <double>429525543.92370653</double>
            </ArrayOfDouble>
         </matrix3Result>
      </matrix3Response>
   </soap:Body>
</soap:Envelope>
```

```
Wed Dec 07 09:49:31 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 11ms (748 bytes)
Wed Dec 07 09:49:32 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 12ms (748 bytes)
Wed Dec 07 09:49:33 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 18ms (748 bytes)
Wed Dec 07 09:49:33 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 15ms (748 bytes)
Wed Dec 07 09:49:34 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 14ms (748 bytes)
Wed Dec 07 09:49:34 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 16ms (748 bytes)
Wed Dec 07 09:49:35 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 9ms (748 bytes)
Wed Dec 07 09:49:36 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 13ms (748 bytes)
Wed Dec 07 09:49:36 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 11ms (748 bytes)
Wed Dec 07 09:49:37 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 15ms (748 bytes)
Wed Dec 07 09:49:37 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 17ms (748 bytes)
```

```
Wed Dec 07 09:49:37 PST 2011:INFO:Got response for [WebService1Soap.matrix3:Request 1] in 17ms (748 bytes)
```

Figure A.1: Snapshot of SOAP UI program showing the result of applying a web service which multiplies two 3by3matrices including execution time and total number of bytes transferred in SOAP messages.



http://localhost:1086/WebService1.asmx

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoa
   <soapenv:Header/>
   <soapenv:Body>
      <tem:matrix2>
         <tem:first1>132213.4</tem:first1>
         <tem:first2>14242.65</tem:first2>
         <tem:first3>21332.213</tem:first3>
         <tem:first4>321.312</tem:first4>
         <tem:second1>234.234</tem:second1>
         <tem:second2>234.432</tem:second2>
         <tem:second3>432.432</tem:second3>
         <tem:second4>342.243</tem:second4>
      </tem:matrix2>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http:/
   <soap:Body>
      <matrix2Response xmlns="http://tempuri.org/">
         <matrix2Result>
            <ArrayOfDouble>
               <double>37127851.1604</double>
               <double>35869499.05275</double>
            </ArrayOfDouble>
            <ArrayOfDouble>
               <double>5135675.1706260005</double>
               <double>5110920.140832</double>
            </ArrayOfDouble>
         </matrix2Result>
      </matrix2Response>
   </soap:Body>
</soap:Envelope>
```

```
Wed Dec 07 10:02:10 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 9ms (529 bytes)
Wed Dec 07 10:02:10 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 8ms (529 bytes)
Wed Dec 07 10:02:11 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 10ms (529 bytes)
Wed Dec 07 10:02:11 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 9ms (529 bytes)
Wed Dec 07 10:02:12 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 11ms (529 bytes)
Wed Dec 07 10:02:13 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 4ms (529 bytes)
Wed Dec 07 10:02:14 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 17ms (529 bytes)
Wed Dec 07 10:02:15 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 10ms (529 bytes)
Wed Dec 07 10:02:15 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 10ms (529 bytes)
Wed Dec 07 10:02:16 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 8ms (529 bytes)
Wed Dec 07 10:02:17 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 8ms (529 bytes)
Wed Dec 07 10:02:17 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 21ms (529 bytes)
Wed Dec 07 10:02:18 PST 2011:INFO:Got response for [WebService1Soap.matrix2:Request 1] in 8ms (529 bytes)
```

Request Properties

Property	Value
Name	Request 1
Description	
Message ...	589
Encoding	UTF-8

Figure A.2: Snapshot of SOAP UI program showing the result of applying a web service which multiplies two 2by2matrices including execution time and total number of bytes transferred in SOAP messages