Web Service Performance using SOAP Compression

G M Tere
Department of Computer Science,
Shivaji University, Kolhapur,
Maharashtra - 416004, India
girish.tere@gmail.com

R. R. Mudholkar
Department of Electronics,
Shivaji University,
Kolhapur,
Maharashtra – 416004, India
rrm_eln@unishivaji.ac.in

B T Jadhav
Y.C. Institute of Science,
Satara,
Maharashtra - 415001, India
btj21875@gmail.com

Abstract— **Recent distributed computing applications are developed using different platforms working on different Operating Systems and these applications are running on heterogeneous hardware. SOAP protocol is mainly used for communicating among heterogeneous processes developed in platforms like .Net, Java, Android and so on. SOAP messages are basically XML documents and thus because of use of XML messages for interprocess communication there is a chance that performance of distributed computing applications will suffer. Client's request and server's response is sent in SOAP format and the size of SOAP message may be large if it is carrying information from databases. To transport such a large SOAP message more time will be required resulting in increase in response time and decrease in throughput. One solution is to use run time compression technique to compress SOAP message. Our experiments show that response time is reduced by 56% while using compression technique.**

Keywords-Compression, performance, SOAP, SOAP Extension, Web service

I. Introduction

On more complex XML Web services, the SOAP response could be a very large dataset. For instance, when it is serialized to XML, a dataset that contains all columns of the table "orders" from Northwind database is about 454 KB. We developed an application that retrieves this dataset by invoking an XML Web service; the SOAP response would contain all of that data. *Web services* describe a standardized way of integrating web based applications using the XML, SOAP, WSDL and UDDI techniques. They are used for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall. Web service is suitable for distributed system because it use SOAP protocol. SOAP a simple and extensible for exchange of message, is the most widely used communication protocol in the web service. Beside of advantage, web services have basic problem and that is low performance [7]. Some of approach introduced to improve performance. In this paper, attempt is made to improved performance of web services using runtime compression technique [8].

II. SOAP Extensions

SOAP Extensions allow us to plug into the SOAP architecture of the .NET Framework in order to perform some pre or post-processing on SOAP messages. When a SOAP message is received by the Web Service's HTTP handler [1], the SOAP message is de-serialized into objects (for example, SoapHeader for header information and SoapMessage for the body), and eventually passed into some Web Method. After the Web Method has finished, the result as well as any SoapHeader, SoapMessage, or SoapException objects are serialized into a SOAP message [5], [6] and sent back up the chain and to the client. SOAP extensions are a Microsoft ASP.NET Web method interception mechanism that can be used to manipulate SOAP requests/responses before they are sent on the network. Developers can write code that executes before and after the serialization and deserialization of messages. SOAP extensions provide a low-level API for implementing serialization and deserialization of a SOAP message.

As SOAP message is basically XML document, it contain lot of redundant data. Therefore we selected compression technique to compress the SOAP response. Using SOAP extensions, we can reduce the size of SOAP messages traveling on the network when a consumer invokes a method from an XML Web service. In most cases SOAP requests are much smaller in size than SOAP responses (for example, a large dataset returned by server), therefore we have compressed only SOAP responses. We used mechanism called object serialization where an object is represented as a sequence of bytes that includes the object's data as well as information about the object's type and the types of data stored in the object. The reverse method is called as serialization. As shown in Fig. 1 on server side and AfterSerialize stage the SOAP response is compressed (shown as '1' in Fig. 1) and travels through network as a compressed SOAP message and on client side and BeforeDeserialize stage the compressed SOAP message is decompressed (shown as '2' in Fig. 1). After uncompressing SOAP message deserialization process is performed.

SOAP extensions must be running on both client and server. If the size of SOAP message is large, the sending of data will be difficult because of limited bandwidth. In presented approach before sending SOAP message in media, message should be compressed in order to decrease transfer time. An important point in compressing of SOAP message is that a whole of SOAP message is not compressed but only the body of SOAP message is compressed, because the header of SOAP message is required for correct function.

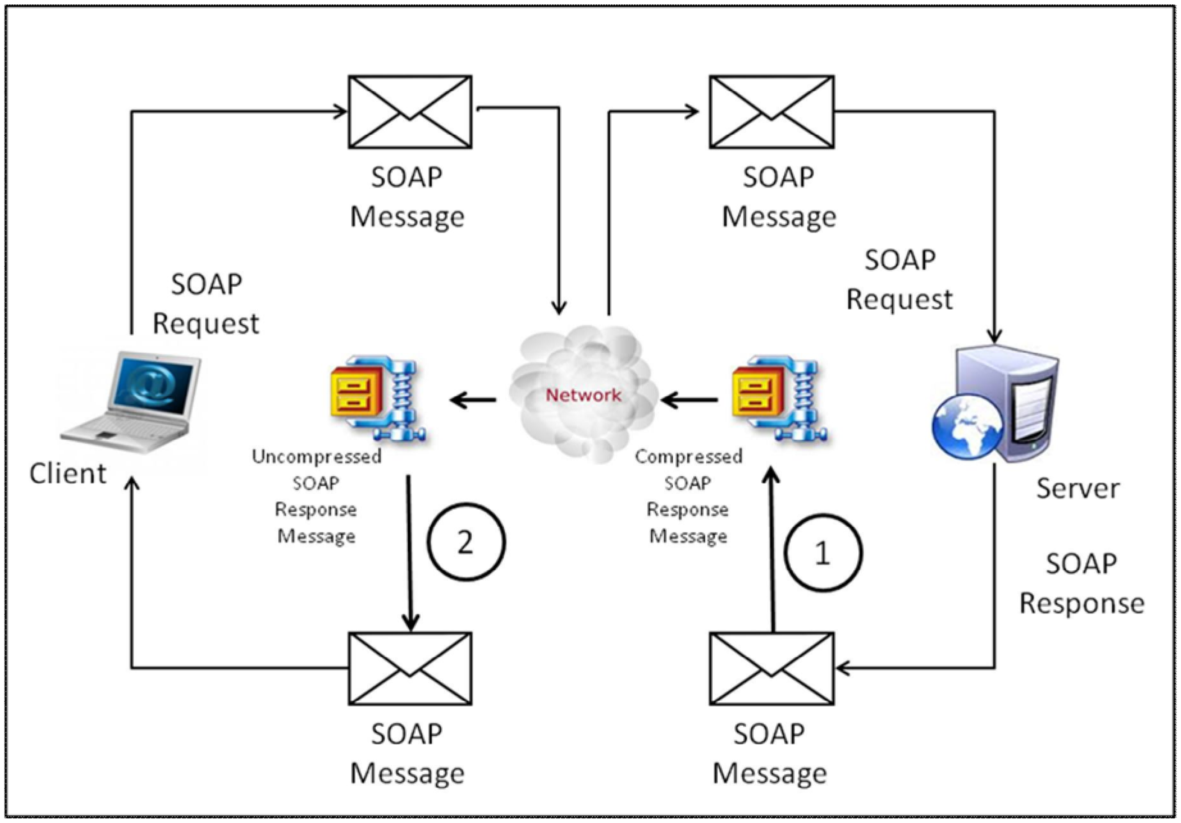


Figure 1. Runtime compression of SOAP response

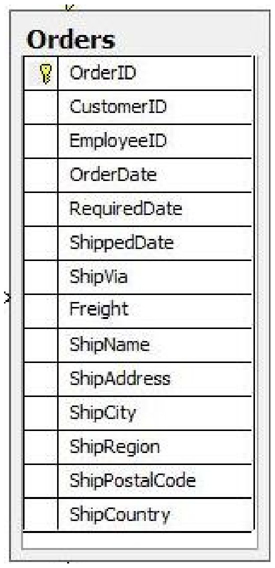
III. Experiment Setup

- Server side: DELL Inspron N5110 Intel Core i5, 4GB RAM and Windows 7 OS
- Consumer side: Dell Inspiron N51545 Intel Core duo with 4 GB RAM and Windows XP.
- Database used: Microsoft SQL Express Server 2005
- IDE used for application development: Visual Studio 8
- Both laptops were connected using Wi-Fi router.

To compress Web service's SOAP responses, following steps are used:

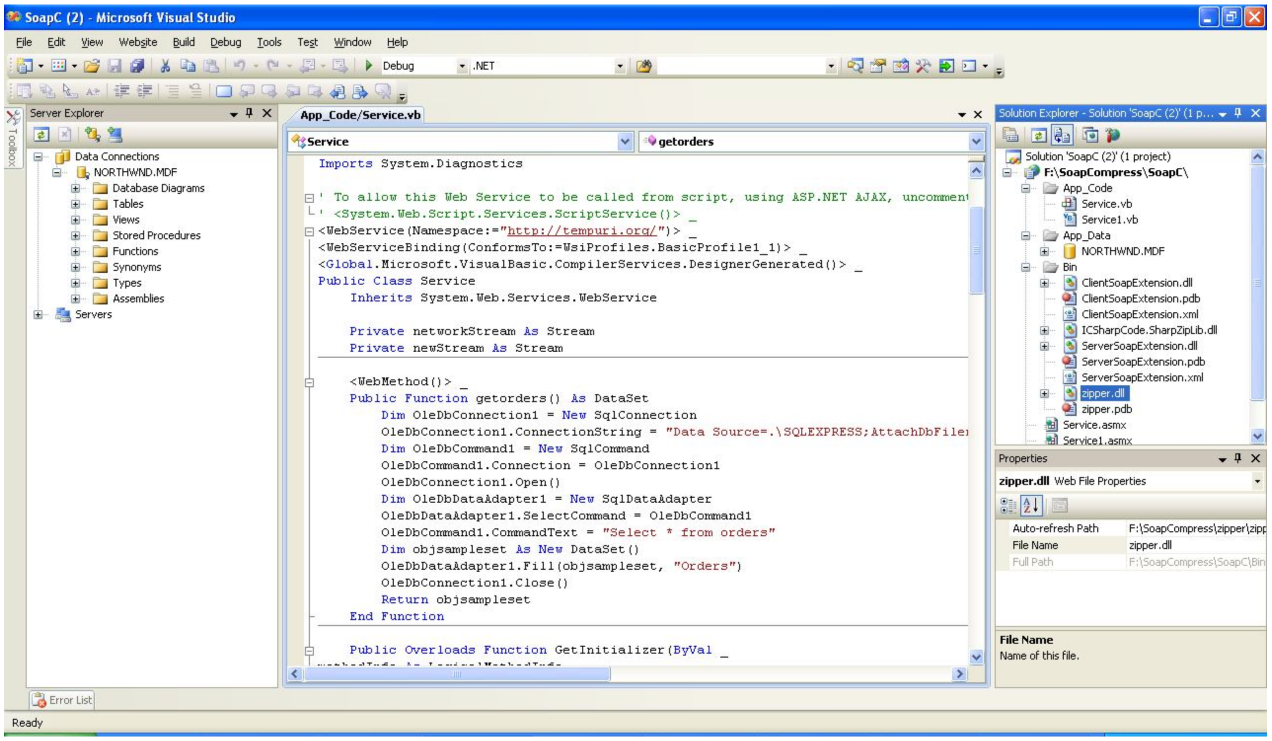
- Compress the SOAP response message after serialization on the server.
- Decompress the SOAP response message before deserialization on the client.

The application uses SOAP extension. We developed XML Web service that returns a large dataset created from Orders table of Northwind database. Structure of Orders tables is shown in Fig. 2. The developed application using Microsoft Visual Studio 8 is shown in Fig. 3.



Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry

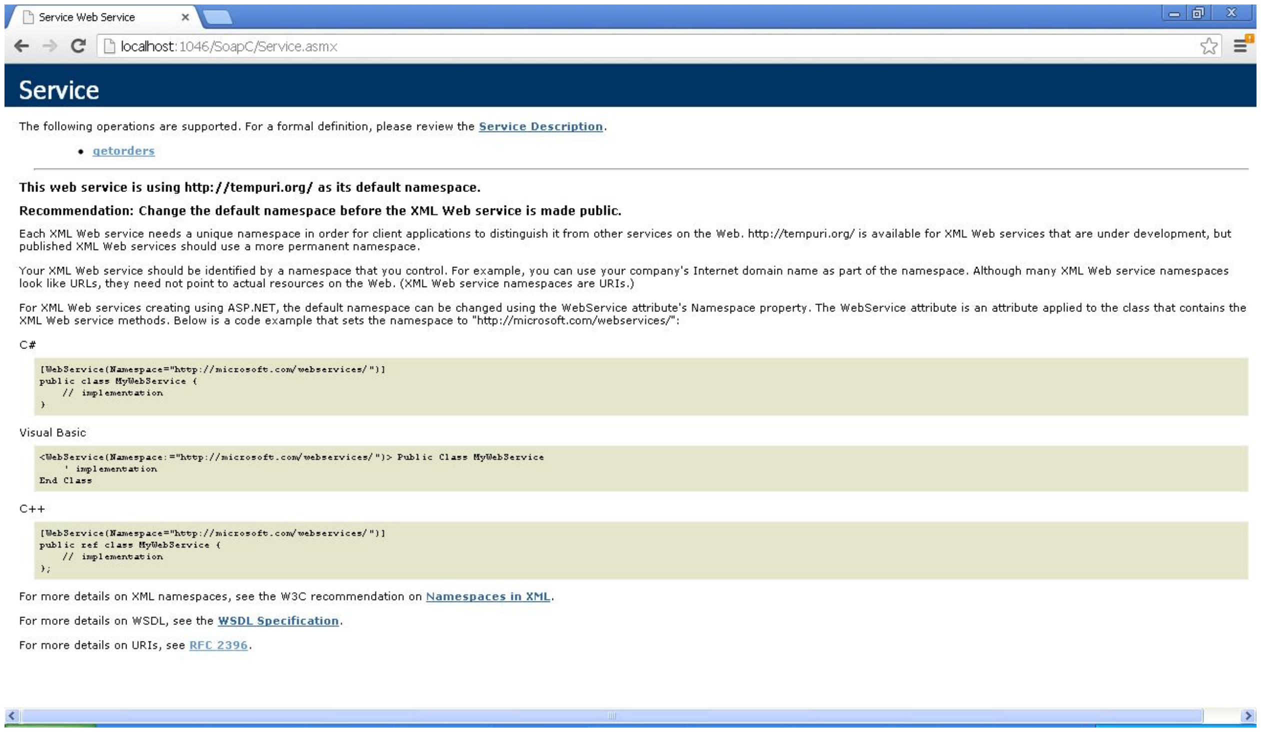
Figure 2: Structure of Orders Table



```
Imports System.Diagnostics

' To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment
' <System.Web.Script.Services.ScriptService()> _
<WebService(Namespace:="http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class Service
    Inherits System.Web.Services.WebService

    Private networkStream As Stream
    Private newStream As Stream

    <WebMethod()> _
    Public Function getorders() As DataSet
        Dim OleDbConnection1 = New SqlConnection
        OleDbConnection1.ConnectionString = "Data Source=.\SQLEXPRESS;AttachDbFile
        Dim OleDbCommand1 = New SqlCommand
        OleDbCommand1.Connection = OleDbConnection1
        OleDbConnection1.Open()
        Dim OleDbDataAdapter1 = New SqlDataAdapter
        OleDbDataAdapter1.SelectCommand = OleDbCommand1
        OleDbCommand1.CommandText = "Select * from orders"
        Dim objsampleset As New DataSet()
        OleDbDataAdapter1.Fill(objsampleset, "Orders")
        OleDbConnection1.Close()
        Return objsampleset
    End Function

    Public Overloads Function GetInitializer(ByVal _
```

Figure 3. Application developed in .NET to demonstrate use of compression technique

Fig. 4 shows running of Web service ‘Service’, which contains ‘getorders’ method .



Service

The following operations are supported. For a formal definition, please review the Service Description.

- getorders

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```

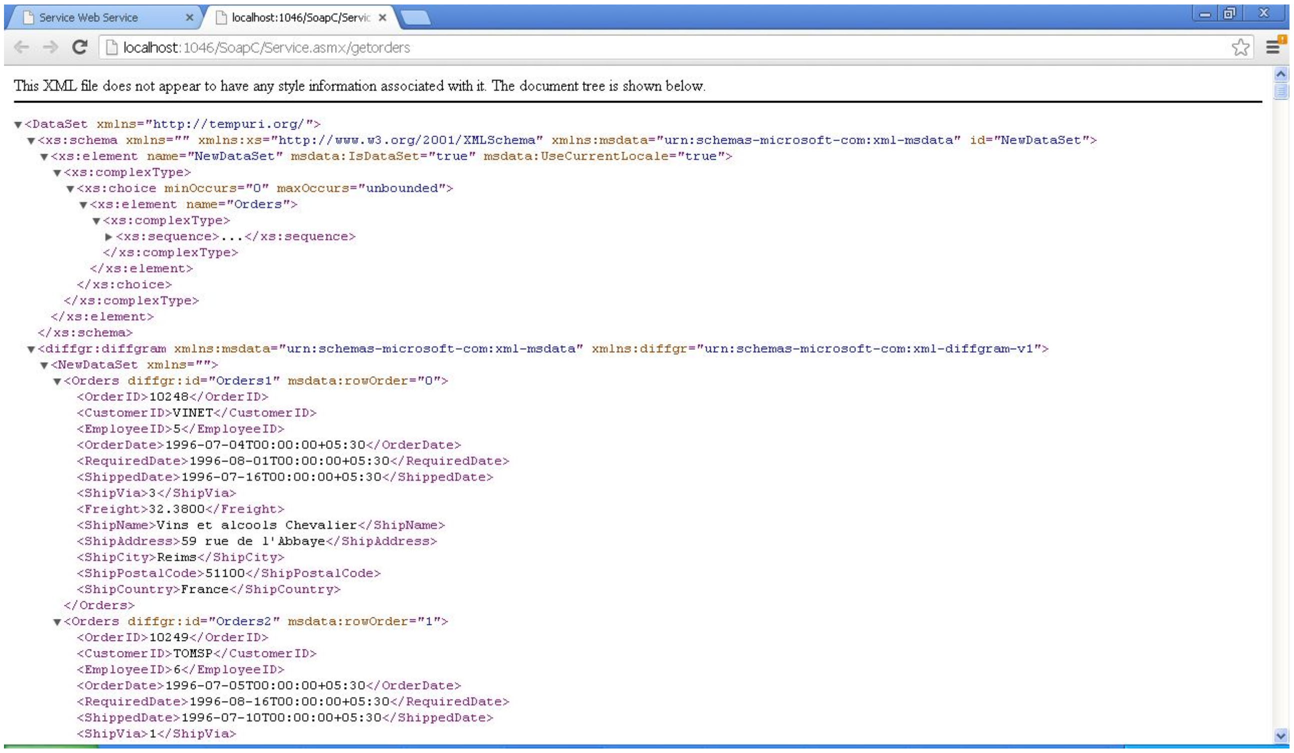
For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.

For more details on WSDL, see the WSDL Specification.

For more details on URIs, see RFC 2396.

Figure 4. Web service ‘Service’

Invoking getorders method give output as shown in Fig. 5.


Service Web Service
localhost:1046/SoapC/Servic
localhost:1046/SoapC/Service.asmx/getorders

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<DataSet xmlns="http://tempuri.org/">
  ▼<xs:schema xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" id="NewDataSet">
    ▼<xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
      ▼<xs:complexType>
        ▼<xs:choice minOccurs="0" maxOccurs="unbounded">
          ▼<xs:element name="Orders">
            ▼<xs:complexType>
              ▶<xs:sequence>...</xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  ▼<diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
    ▼<NewDataSet xmlns="">
      ▼<Orders diffgr:id="Orders1" msdata:rowOrder="0">
          <OrderID>10248</OrderID>
          <CustomerID>VINET</CustomerID>
          <EmployeeID>5</EmployeeID>
          <OrderDate>1996-07-04T00:00:00+05:30</OrderDate>
          <RequiredDate>1996-08-01T00:00:00+05:30</RequiredDate>
          <ShippedDate>1996-07-16T00:00:00+05:30</ShippedDate>
          <ShipVia>3</ShipVia>
          <Freight>32.3800</Freight>
          <ShipName>Vins et alcools Chevalier</ShipName>
          <ShipAddress>59 rue de l'Abbaye</ShipAddress>
          <ShipCity>Reims</ShipCity>
          <ShipPostalCode>51100</ShipPostalCode>
          <ShipCountry>France</ShipCountry>
        </Orders>
      ▼<Orders diffgr:id="Orders2" msdata:rowOrder="1">
          <OrderID>10249</OrderID>
          <CustomerID>TOMSP</CustomerID>
          <EmployeeID>6</EmployeeID>
          <OrderDate>1996-07-05T00:00:00+05:30</OrderDate>
          <RequiredDate>1996-08-16T00:00:00+05:30</RequiredDate>
          <ShippedDate>1996-07-10T00:00:00+05:30</ShippedDate>
          <ShipVia>1</ShipVia>
```

Figure 5. Output showing the invocation of a Web service

On the consumer side, we developed a Microsoft Windows application that invokes the above XML Web service, retrieves the dataset, and can be presented in proper UI or can be processed further as per requirement.

The consumer invokes two identical XML Web services, only one of which uses SOAP compression. The Timer class below is used to measure invocation time:

```
Public Class ClsTimer
  Private Declare Function timeGetTime
Lib "winmm" () As Long

  Private lngStartTime As Integer
  Private lngTotalTime As Integer
  Private lngCurTime As Integer

  Public ReadOnly Property TotalTime() As
String
    Get
      TotalTime = lngTotalTime
    End Get
  End Property

  Public Sub StartTiming()
    lngTotalTime = 0
    lngStartTime = timeGetTime()
  End Sub

  Public Sub StopTiming()
    lngCurTime = timeGetTime()
    lngTotalTime = (lngCurTime -
lngStartTime)
  End Sub
End Class
```

On the server side, the SOAP response from the server is compressed to reduce its size. Following steps are completed for compressing SOAP message at runtime.

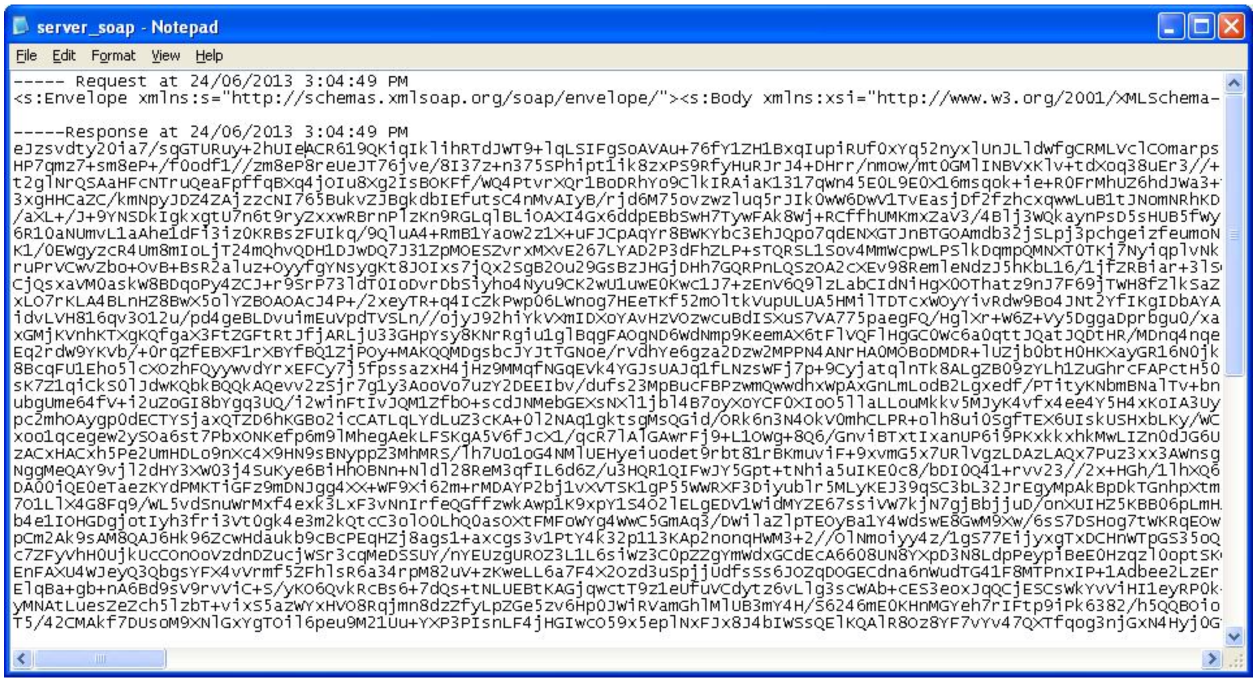
1. Using Microsoft Visual Studio .NET, we created a new Microsoft Visual Basic .NET class library project which is extended from “SoapExtension" class.

2. We added the ServerSoapExtension.dll assembly as reference and declare the SOAP extension on web.config of the XML Web service:

   ```
   <?xml version="1.0" encoding="utf-8"
   ?>
   ```

```
<configuration>
   <system.web>
     <webServices>
     <soapExtensionTypes>
        <add
type="ServerSoapExtension.myextension,
        ServerSoapExtension"
priority="1" group="0"/>
     </soapExtensionTypes>
   </webServices>
      ...
   </system.web>
 </configuration>
```

We used a temporary folder ("c:\temp"), to capture as text file ("c:\temp\server_soap.txt") the SOAP request and the compressed SOAP response. The content of server_soap.txt are shown in Fig. 6.



```
----- Request at 24/06/2013 3:04:49 PM
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-

-----Response at 24/06/2013 3:04:49 PM
eJzsvdty2Oia7/sgGTURuy+2hUIeACR619QKiqIklihRTdJWT9+lqLSIFgSoAVAu+76fY1ZH1BxqIupiRUfOxYq52nyxlUnJLldwfgCRMLVclCOmarps
HP7qmz7+sm8eP+/fOodf1//zm8eP8reUeJT76jve/8I37z+n375SPhiptlik8zxPS9RfyHuRJrJ4+DHrr/nmow/mtOGMlINBVxKlv+tdXoq38uEr3//+
t2glNrQSAaHFcNTruQeaFpffqBXq4jOIu8Xg2IsBOKFf/WQ4PtvrXQr1BoDRhYo9ClkIRAiaK1317qWn45EOL9EOX16msqok+ie+ROFrMhUZ6hdJWa3+
3xgHHCaZC/kmNpyJDZ4ZAjzzcNI765BukvZJBgkdbIEfutsC4nMvAIyB/rjd6M75ovzwzluq5rJIkOWw6DwV1TvEasjDf2fzhcxqwwLuB1tJNomNRhKD
/aXL+/J+9YNSDkIgkxgtU7n6t9ryZxxwRBrnPlzKn9RGLqlBLiOAXI4Gx6ddpEBbSwH7TywFAk8Wj+RCffhUMKmxZaV3/4Blj3WQkaynPsD5sHUB5fwy
6R1OaNUmvL1aAhe1dFi3izOKRBszFUIkq/9QluA4+RmB1Yaow2z1X+uFJCpAqYr8BWKYbc3EhJQpo7qdENXGTJnBTGOAmdb32jSLpj3pchgeizfeumoN
K1/OEWgyzcR4Um8mIoLjT24mQhvQDH1DJwDQ7J31ZpMOESZvrxMXvE267LYAD2P3dFhZLP+sTQRSL1Sov4MmWcpwLPSlkDqmpQMNXTOTKj7NyiqplvNk
ruPrVCwvZbo+OvB+BsR2aluz+OyyfgYNsygKt8JOIxs7jQx2SgB2Ou29GsBzJHGjDHh7GQRPnLQSzOA2cXEv98RemleNdzJ5hKbL16/1jfZRBiar+3lS
CjQsxaVM0askW8BDqoPy4ZCJ+r9SrP73ldTOIoDvrDbSiyho4Nyu9CK2wU1uwEOKwc1J7+zEnV6Q9lzLabCIdNiHgxOOThatz9nJ7F69jTWH8fZlkSaZ
xLO7rKLA4BLnHZ8BwX5olYZBOAOAcJ4P+/2xeyTR+q4IcZkPwp06LWnog7HEeTKf52mOltkVupULUA5HMilTDTcxWOyYivRdw9Bo4JNt2YfIKgIDbAYA
idvLVH816qv3o12u/pd4geBLDvuimEuvpdTVSLn//ojyJ92hiYkVXmIDXoYAvHzVOzwcuBdISXuS7VA775paegFQ/HglXr+W6Z+Vy5DggaDprbgu0/xa
xGMjKVnhKTXgKQfgaX3FtZGFtRtJfjARLjU33GHpYsy8KNrRgiu1glBqgFAOgND6WdNmp9KeemAX6tFlVQFlHgGC0Wc6a0qttJQatJQDtHR/MDnq4nqe
Eq2rdw9YKVb/+0rqZfEBXF1rXBYfBQ1ZjPOy+MAKQQMDgsbcJYJtTGNoe/rvdhYe6gza2Dzw2MPPN4ANrHAOMOBoDMDR+lUZjb0btHOHKXayGR16N0jk
8BcqFU1Eho5lcXOzhFQyywvdYrxEFCy7j5fpssazxH4jHz9MMqfNGqEVk4YGJsUAJq1fLNzsWFj7p+9CyjatqlnTk8ALgZBO9zYLh1ZuGhrcFAPctH5O
sK7Z1qiCkS0lJdwKQbkBQQkAQevv2zSjr7g1y3AooVo7uzY2DEEIbv/dufs23MpBucFBPzwmQwwdhxWpAxGnLmLodB2Lgxedf/PTityKNbmBNalTv+bn
ubgUme64fV+i2uZoGI8bYgq3UQ/i2winFtIvJQM1Zfbo+scdJNMebGEXsNXl1jbl4B7oyXoYCFOXIoO5llaLLouMkkv5MJyK4vfx4ee4Y5H4xKoIA3Uy
pc2mhOAygp0dECTYSjaxQTZD6hKGBo2icCATLqLYdLuZ3cKA+Ol2NAq1gktsgMsQGid/ORk6n3N4OkV0mhCLPR+olh8uiOSgfTEX6UIskUSHxbLKy/WC
xoo1qcegew2ySOa6st7PbxONKefp6m9lMhegAekLFSKgA5V6fJcX1/qcR7lAlGAwrFj9+L1OWg+8Q6/GnviBTxtIxanUP6i9PKxkkxhkMwLIZn0dJG6U
zACxHACxh5Pe2UmHDLo9nXc4X9HN9sBNyppZ3MhMRS/lh7Uo1oG4NMlUEHyeiuodet9rbt81rBKmuviF+9xvmG5x7URlVgzLDAzLAQx7Puz3xx3Awnsg
NggMeQAY9vjl2dHY3XW03j4SuKye6BiHhOBNn+Nldl28ReM3qfIL6d6Z/u3HQR1QIFwJY5Gpt+tNhia5uIKE0c8/bDI0Q41+rvv23//2x+HGh/1lhXQ6
DA00iQE0eTaezKYdPMKTiGFz9mDNJgg4XX+WF9Xi62m+rMDAYP2bj1vXVTSK1gP55WWRXF3Diyublr5MLyKEJ39qSC3bL32JrEgyMpAkBpDkTGnhpXtm
7O1LlX4G8Fq9/WL5vdSnuWrMxf4exk3LxF3vNnIrfeQGffzwkAwp1K9xpY1S4O2lELgEDV1widMYZE67ssiVW7kjN7gjBbjjuD/onXUIHZ5KBB06pLmH
b4e1IOHGDgjotIyh3fri3Vt0gk4e3m2kQtcC3olO0LhQ0asOXtFMFoWYg4WWC5GmAq3/DWilaZlpTEOyBa1Y4WdswE8GwM9Xw/6sS7DSHog7tWKRqEOw
pCm2Ak9sAM8QAJ6Hk96ZcwHdaukb9cBcPEqHZj8ags1+axcgs3v1PtY4k32p113KAp2nonqHWM3+2//OlNmoiyy4z/1gS77EijyxgTxDCHnWTpGS35oQ
c7ZFyVhH0UjkUcCOnOoVzdnDZucjWSr3cqMeDSSUY/nYEUzgUROZ3L1L6siWz3C0pZZgYmWdxGCdEcA6608UN8YXpD3N8LdpPeypiBeE0HzqzlOoptSK
EnFAXU4WJeyQ3QbgsYFX4vVrmf5ZFhlsR6a34rpM82uV+zKweLL6a7F4X2Ozd3uSpjjUdfsSs6JOZqDOGECdna6nWudTG41F8MTPnxIP+1Adbee2LzEr
ElqBa+gb+nA6Bd9sV9rvViC+S/yKO6QvkRcBs6+7dQs+tNLUEBtKAGjqwctT9z1eUfuVCdytz6vLlg3scWAb+cES3eoxJqQCjESCsWkYvViHI1eyRP0k
yMNAtLuesZeZch5lzbT+vixS5aZWYxHVO8Rqjmn8dzZfyLpZGe5zv6Hp0JWiRVamGhlMlUB3mY4H/S6246mEOKHnMGYeh7rIFtp9iPk6382/h5QQBOio
T5/42CMAkf7DUsoM9XNlGxYgTOil6peu9M21Uu+YXP3PIsnLF4jHGIwcO59x5eplNxFJx8J4bIWSsQElKQAlR8Oz8YF7vYv47QXTfqog3njGxN4HyjOG
```

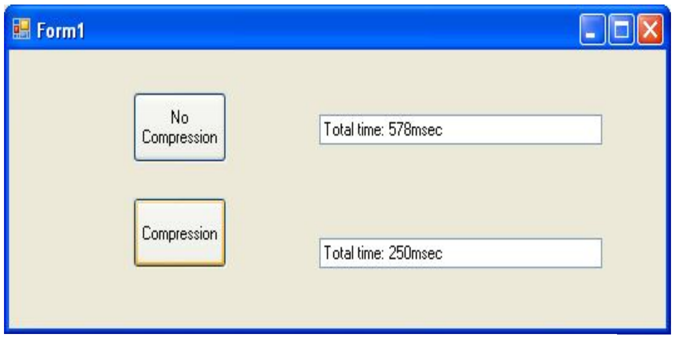
Figure 6. Content of text file, server_soap.txt

On the consumer side, the SOAP response from the server is uncompressed to retrieve the original content. To achieve this following steps are completed:

1. Using Visual Studio .NET, we created a new Visual Basic .NET class library project.
2. We added the ClientSoapExtension.dll assembly as reference and declare the SOAP extension on the XML Web service reference of our application:

IV. RESULT ANALYSIS

On the consumer side a Windows application invokes an XML Web service. The XML Web service returns a dataset and can be shown on the consumer side. Response time required for sending normal SOAP response message and compressed SOAP message is shown in Fig. 7.



Total time: 578msec

Total time: 250msec

Figure 7.Application which invokes the same XML Web Service with and without using SOAP Compression.

As shown in Fig. 7, we need 56% less time in retrieving the dataset, containing information of 800 orders, on the consumer side with very little impact on the CPU load. SOAP Compression can significantly increase the XML Web Services' performance when consumers and servers exchange large data [2]. There are a lot of solutions for improving performance on the web. SOAP Extensions is the way to improve XML Web Services' performance by compressing the exchanged data with small impact on CPU load [5]. We repeated the experiment by varying number of records in an orders table. Results obtained are shown in Table 1 and graphically in Fig. 8. From the experiments performed we conclude that technique of compressing SOAP response is useful when large data is need to be sent. For data less than 10 KB, data should not be compressed. For compressing SOAP we have used SharpZip library [9], which is open source software.

TABLE I. RESPONSE TIME REQUIRED FOR SENDING DIFFERENT DATA AS SOAP RESPONSE AND COMPRESSED SOAP RESPONSE MESSAGE

Rows in Orders	Time required to send normal SOAP (response in msec)	Time required to send compressed SOAP (response in msec)
300	362	198
400	396	204
500	428	215
600	468	222
700	517	234
800	578	250

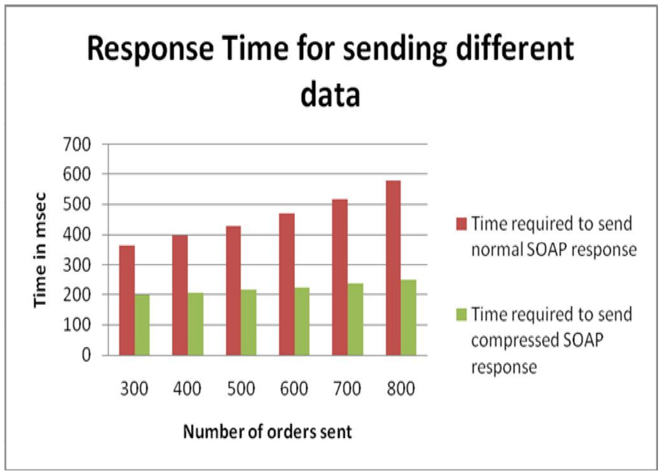


Figure 8. Response time required for sending different data with and without compression

V. CONCLUSIONS

As SOAP message contain large data, more time is required to transport SOAP message from server to client. This time to transport SOAP message can be reduced if we compress SOAP response message after serialization on the server. SharpZip library is used for compression. We developed .Net application which calls the Web service developed in .Net. If SOAP message is small than 2KB, then compression is not desirable. For larger SOAP size, time required to compress SOAP and uncompress SOAP is negligible as the time required to transport SOAP message. Our experiment show that we have saved nearly 58% of time using SOAP compression at run time.

ACKNOWLEDGMENT

We thank teachers of Department of Computer Science, Shivaji University, Kolhapur for motivating us for this research work. We wish to thank Principal of Thakur College of Science and Commerce, Mumbai and Principal, Y.C. Institute of Science, Satara, for providing resources required to perform experiments and to complete this paper.

REFERENCES

[1] Austin, D., Barbir, A., Ferris, C., et al. Web Services Architecture Requirements, W3C Working Group Note 11 February 2004

[2] Chiu, K., Govindaraju, M., and Bramley, R. Investigating the Limits of SOAP Performance for Scientific Computing. In Proceedings of 11th. IEEE International Symposium on High Performance Distributed Computing HPDC-11 2002 (HPDC'02). Edinburgh, Scotland, p. 246-254:IEEE, 2002

[3] Davis, D. and Parashar, M. Latency Performance of SOAP Implementations. In Proceedings of IEEE Cluster Computing and the GRID 2002 (CCGRID'02). Berlin, Germany, IEEE, 2002

[4] M.C. Rosu. Asoap:Adaptive soap message processing and compression. Proceedings of IEEE International Conference on Web Services (ICWS 2007), 2007.

[5] Madhusudhan Govindaraju, Aleksander Slominski, Kenneth Chiu, Pu Liu, Robert van Engelen, Michael J. Lewis, Toward Characterizing the Performance of SOAP Toolkits, Procceedings of 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, November 2004.

[6] Mitra, N. SOAP Version 1.2 Part 0: Primer, W3C Recommendation 24 June 2003 http://www.w3.org/TR/soap12-part0/

[7] N. Abu Ghazaleh, M. Lewis, and M. Govindaraju. Differential serialization for optimized soap performance. Proceedings of 13th International Symposium on High Performance Distributed Computing, 2004.

[8] Nair, S.S. XML Compression Techniques: A Survey, (on-line) Accessed 23 September 2004 http://www.cs.uiowa.edu/~rlawrenc/research/Students/SN_04_XMLCompress.pdf

[9] SharpZipLib, http://www.icsharpcode.net/opensource/sharpziplib/