A Performance Evaluation of Using SOAP with Attachments for e-Science

Ying Ying, Yan Huang, and David W. Walker
School of Computer Science, Cardiff University
5 The Parade, Roath, Cardiff CF24 3AA
{Y.Ying, Yan.Huang, David.W.Walker}@cs.cardiff.ac.uk

Abstract

SOAP is now commonly used as the main transport protocol in Service-Oriented Architectures (SOA), but it is debatable whether SOAP can really meet the performance needs of e-Science. This paper presents an extended experimental evaluation of the performance of SOAP with Attachments(SwA)[1]. The performance of different SOAP variants: standard SOAP, SwA using MIME(Multipurpose Internet Mail Extension), SWA using DIME(Direct Internet Message Encapsulation), and XSOAP, are evaluated in communicating multiple floating-point matrices and a number of complex data structs. The objective of this paper is to demonstrate that SOAP performance in communicating large volumes of data could be simply and effectively improved by adopting SwA. More over, SwA using DIME is identified to be a faster and more efficient message processing approach that using MIME.

1. Introduction

e-Science is based on the integration of distributed resources for solving challenging problems across a broad spectrum of scientific disciplines, and often involves the communication of large complex data objects over the network. The speed with which such data objects can be communicated is an important factor in determining how effective a Grid-based approach is for solving a particular problem. Grid computing is rapidly transitioning to a service-oriented architecture (SOA) based on a Web services framework, which offers the power of resource and service sharing, and a high degree of usability and interoperability. Thus, the underlying communication with and between services plays a critical role in e-Science.

SOAP (Simple Object Access Protocol) is commonly used for inter-process communication in distributed applications, in particular between Web services. SOAP is based on XML technology, and so inherits the inefficiencies of XML, requiring more bandwidth, more storage and more processing power than equivalent binary implementations. Standard SOAP represents the invocation parameters of a Web service in XML, hence it is widely perceived that service invocation can suffer severe performance penalties by adopting the XML-based SOAP protocol for transporting large volumes of data[2].

There are two approaches of SwA: using MIME, which is defined in SOAP Message with Attachments[3](SwA-MIME), and using DIME, which is defined in WS-Attachments[4](SwA-DIME). Both of them define an abstract model for SOAP attachments and the mechanism for encapsulating a SOAP message and zero or more attachments in either MIME or DIME message.

XSOAP (previously known as SOAPRMI) is designed and developed to provide a SOAP based RMI communication system that combines Java RMI with the ubiquity of HTTP and the platform and language independence of XML. The XML Pull Parser (XPP) included in XSOAP is a streaming pull parser, that has been found to have better performance than other current approaches to XML parsing[5].

Several studies have been done on the performance of using SOAP in Web services for both scientific computing and business applications. These include the investigation of the performance improvement gained through message chunking, compression, routing and streaming[2]; a comparison of the latency of SOAP implementations[6]; an evaluation of the limitations of SOAP in scientific computing[7]; an evaluation of the multi-protocol XSOAP in improving SOAP performance[8]; an evaluation of contemporary commercial SOAP implementations[9]; and an evaluation of SOAP performance in a real trading system[10]. However, these studies did not address the extent to which performance could be simply and effectively improved by reducing the payload of a SOAP message, which results in reducing a large amount of XML encoding/decoding. In this paper, we address this issue by transmitting most of the associated data as SOAP attachment(s) to achieve it.

The rest of this paper is organised as follows: a brief discussion MIME and DIME is given in section 2. Section 3 explains how we conduct the experimental evaluation, including the decisions we made and our metrics, and provides the results and analysis of the data collected in our measurements. Conclusions are presented in section 4.

2. MIME and DIME

SOAP Message with Attachments is an abstract model for SOAP attachments defined by W3C in SOAP 1.2 Attachment Features. It provides the basis for the creation of SOAP bindings that transmit attachments along with a SOAP envelope using the MIME mechanism, and provides a reference to the attachments from the envelope. SOAP attachments are described using the notion of a compound document structure consisting of a primary SOAP message part and zero or more secondary parts known as attachments[11]. The primary SOAP message provides the processing context for the compound SOAP structure. A secondary part is a resource that has identity and is identified by a URI.

WS-Attachments has been proposed by Microsoft and IBM to achieve the same general functionality as SOAP Message with Attachments. But it just defines an abstract model for SOAP attachments and based on this model defines a mechanism for encapsulating a SOAP message and zero or more attachments in a DIME message[12].

Both SOAP Message with Attachments and WS-Attachments attempt to overcome the following two problems associated with sending attachments in a SOAP message[13]:

- Encapsulation of binary data in the form of image files etc. cannot be done without additional encoding/decoding of the data.
- Encapsulation of other XML documents as well as XML fragments is cumbersome within a SOAP message – especially if the XML parts do not use the same character encoding.

But WS-Attachments has another approach to break a message into records and to provide random access to these records, which avoids the significant overhead in generic message processing as well as in parsing and memory allocation.

2.1 MIME

MIME was originally developed for emailing with attachments. Typically, e-mail messages with attachments are sent over the Internet using Simple Mail Transfer Protocol (SMTP) and MIME. SMTP is limited to 7-bit ASCII text with a maximum line length of a thousand characters which results in the inability to send attachments. MIME addresses these limitations by specifying message header fields and allowing different related objects such as attachments to be included in the message body in the form of a MIME multipart. RFC 2387 specifies the Multipart/Related Content-Type to provide a mechanism for representing an object that consists of related MIME body parts. In SOAP Messages with Attachments, this MIME multipart mechanism is used for encapsulation of compound documents to bundle attachments to the SOAP message.

A typical SOAP message with attachments using MIME is structured as follows:

```
Content-Type: multipart/related
type="text/xml"
start="<main>"
boundary="-----=Part_MIME"
....
-----=Part_MIME
Content-Type:text/xml; charset=UTF-8
Content-Transfer-Encoding: binary
Content-id=<main>

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelop ....>
...
   <in0 href="cid:attachment"/>
...
</soapenv:Envelope>
-----=Part_MIME
```

Content-Type:application/octet-stream
Content-Transfer-Encoding: binary
Content-id=<attachment>
…
In each multipart header, Content-Type, and Content-Transfer-Encoding declare the type and encoding style of this part, and Content-id identifies and references this part.

2.2 DIME

DIME is a new specification from Microsoft that provides a method for sending and receiving SOAP messages along with additional attachments, like binary files, XML fragments, and even other SOAP messages[4].

A DIME message consists of a series of one or more DIME records. Each record is self-describing – that is, the record header contains binary information used by a parser to interpret the message. The most significant fields in a DIME record are MB and ME, which specify whether this record is the first or the last record of the message. This feature is an important difference between MIME and DIME message processing. When parsing a MIME message, all of the data in the message must be read and interpreted to determine simple things like the number of attachments included in the message. However, when using DIME, a parser can simply use the data in the record headers to quickly walk through and count the number of records in the message without having to read any record data. This feature enables DIME to process messages faster and more efficiently.

A typical SOAP message with attachments using DIME is structured as follows:

Content-Type: application/dime
...
00001 1 0 0 0010 0000000000000000000
0000000000000000 0000000000101000
00000000000000000000000110110101
...
<soapenv:Envelop…>
...
   <in0 href="uuid:attachment"/>
...
</soapenv:Envelop>
00001 0 0 1 0001 0000000000000000000
0000000000101001 0000000000001010
00000000000101011010101011100000
uuid:attachment
application/octet-stream
...
In DIME records, a UUID (Universal Unique Identifier) or URI (Uniform Resource Identifier) are used as identifiers in referencing DIME records from the primary SOAP message. The SOAP envelope part is encapsulated together with all the attachments as the first record of the DIME message, while MIME encapsulates them as separate parts in the message.

3. Experimental Evaluation

3.1 Experimental Design

The tests described below are designed to evaluate the performance of different SOAP variants: standard SOAP, SwA-MIME, SwA-DIME, and XSOAP. Our main interest is in SOAP and SwA. But since XSOAP represents a very different approach to improve SOAP performance, we are also interested in how it compares with the other SOAP variants. The evaluation is based on two groups of tests: one communicates multiple floating-point matrices, and the other communicates a number of data structs.

The SOAP implementations used in our tests are Apache Axis 1.1 and XSOAP 1.2.29. The main reason for choosing Axis is that Axis supports standard SOAP, and both approaches of SwA. Java 1.4.2 is the programming language. Tomcat 5.0 is used as the container for web applications, along with the Xerces and Xalan XML parsers. XSOAP1.2.29 is used for the XSOAP tests and has its own HTTP server and XML parser. Nettool, a shareware TCP traffic monitor[14], is used to measure the roundtrip time and the size of SOAP messages. Ethereal, another shareware network traffic monitor[15], is used to monitor the TCP flow over the network. The tests are performed on a local LAN, with a typical bandwidth of 10Mbps. A laptop with a Pentium-III 900 MHz processor and 384 MB of RAM, and Windows XP Professional OS is used as the client machine. The server machine is a dual-processor desktop running Red Hat Linux. Two different platforms are chosen to use in the test for identifying the platform independence and interoperability of SwA.

3.1.1 Performance Metrics

The following performances metrics are used to evaluate the performance of the standard SOAP, SwA-MIME, SwA-DIME and XSOAP:

- **Roundtrip time.** This is the time to send a message from the client to the server and then get a response message back from the server to the client.

- **SOAP message size**. This is the actual size of the SOAP message transmitted on the wire.
- **Serialization and deserialization overheads.** The serialization of SOAP calls can be logically separated into the following phases: (1) traversing the data structures of the invocation parameters; (2) translating the stored values into ASCII representations as required by the XML specification; (3) copying the XML representation (including tags) into a buffer; and (4) sending the buffer over the network[16]. Deserialization is the reverse process to serialization. Some performance studies have pointed out that XML serialization and deserialization are the bottlenecks in SOAP processing[7]. So we quantify the overheads here as the time the client spends on serialization plus the time the server spends on deserialization.

3.1.2 Tests of Communicating Floating-Point Matrices

Since matrices are one of the most heavily used data structures in scientific computing, we test the performance of SOAP in transmitting floating-point matrices. Also, it has been reported[7] that the conversion of data from binary to ASCII and vice-versa is the major performance cost of XML and particularly for the case of floating point values and large arrays, which are found to constitute the major cost of XML encoding and decoding.

Tests are performed by sending one floating-point matrix of different sizes. The sizes of the matrices used are 8x8, 16x16, 32x32, 64x64, 128x128, 256x256 and 512x512. For the standard SOAP method, the float[][] matrix is passed to Axis to put into a SOAP message. Since two-dimensional arrays are not supported in XSOAP, the matrix has to be transformed into a vector to be passed into XSOAP. For both MIME and DIME, the float[][] object is serialized into a byte[] object and then put into a SwA message.

3.1.3 Struct Test

The struct we used in this test contains a String, an Integer and a Boolean. Tests are performed by communicating different numbers of elements of the struct. The size varies from 1, 10, 100, 100, 1000, 5000 to 10000. The purpose of this test is to test SOAP performance for sending a message with a differing number of complex struct types. In the test, although we sent a number of datatypes of the same struct type to simply the programming, in fact we treated each of them as a different struct type. Thus, each element is serialized into a byte[] object and put into the SOAP message as one attachment record. It should be noted that we do not present results for XSOAP for this test because XSOAP cannot transmit complex struct data.

3.2 Experimental Results and Analysis

In this section, we present the results of our tests, along with analysis and explanations. It should be noted that the results are illustrative and might be different under different conditions.

3.2.1 Tests of Communicating Floating-point Matrices of Different Sizes

The roundtrip time results of sending floating-point matrices of different sizes are presented in Figure 1. It should be noted there is no result of standard SOAP sending a 512x512 matrix presented in the figure because the message is over the size limit and could not be processed by Axis. Figure 1 shows that when the matrix sizes are small, XSOAP has the best performance and SwA-MIME has the worst performance, but the differences are small. Once the size of matrix is larger than 32x32, the roundtrip time of XSOAP and SOAP increase steeply as the matrix size increases, while both SwA-MIME and SwA-DIME increase at a much slower pace. When the matrix size is larger than 128x128, both SwA-MIME and SwA-DIME clearly have better performance than SOAP and XSOAP. For example, when the matrix size is 256x256, it takes SOAP 10.85 times longer than SwA-MIME and 25.41 times longer than SwA-DIME. Similarly it takes XSOAP 7.34 times longer than SwA-MIME and 3.13 times longer than SwA-DIME.

To explain the roundtrip time results we did some additional tests on the size of different SOAP messages and the serialization time on the client and deserialization time on the server when communicating one and two 128x128 floating-point matrices. The size of the different SOAP messages when sending different matrices is shown in Figure 2. This reveals an obvious explanation for the roundtrip time results shown in Figure 1, namely that the message size has a significant effect on roundtrip time. Thus, the method with smaller message size has better performance. When the matrix size is small, XSOAP has the smallest message and the best performance; standard

SOAP, with tags added in its message and hence a larger message, has lower performance compared with XSOAP. In the case of SwA-MIME and SwA-DIME, there is a fixed size of bytes at the head of each attachment that contains all the necessary information about the attachment such as its type, encoding method and size, etc. Since each matrix is sent as one attachment in the SOAP message and a fixed number of bytes are added to each attachment, when the matrix size is small, this part is relatively large; but when the matrix is large, it is relative small. That explains why both SwA-MIME and SwA-DIME have better performance than SOAP and XSOAP when the matrix size is larger than a certain size.

It should also be noted that the message sizes of both SwA-MIME and SwA-DIME are very close, but SwA-DIME always has better performance no matter what size the matrix is. The difference in SOAP message size and roundtrip time of using MIME and DIME in this case is always around 310 bytes and 650 ms, respectively.

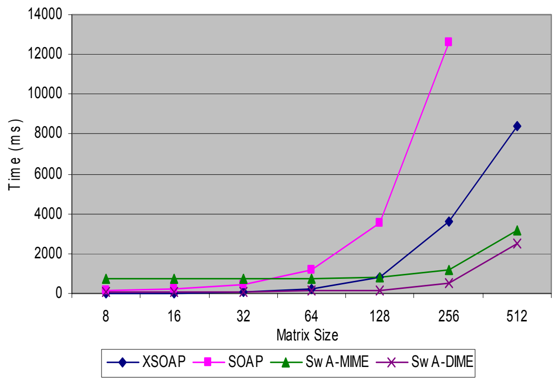

Figure 1: Roundtrip time of communicating one floating-point matrix.

The results of serialization and deserialization overhead are displayed in Table 1 and 2. This figure shows that standard SOAP has the most significant serialization/deserialization overhead. It is not surprising to see that SwA has much better performance than standard SOAP when the former have much smaller messages and spend less time on serialization /deserialization. At the same time, XSOAP has the lowest relative overhead of only 14% of its roundtrip time. In standard SOAP message, each element of the matrix is expressed in XML by adding a tag to it, so it is not surprising to see that SOAP takes longer to perform the serialization and deserialization. Because our tests are run on the same LAN, the network delays are not significant. Under this condition, XML encoding/decoding is a large factor in the performance. The large amount of XML encoding/decoding in standard SOAP messaging not only consumes more bandwidth and processing power but also results in a higher memory requirement. Standard SOAP can not complete the task of sending 512x512 matrix in the test obviously proves this. From our results, for standard SOAP with Axis, every 1 byte increase of the size of the raw data will result in about 7.2 bytes increase in the size of the SOAP message, and costs another 0.12 ms in serialization. Comparing with standard SOAP, SwA will only have an increase of 1.1 bytes in the SOAP message size, in the same circumstances. It is clear that both SwA approaches avoid large amounts of XML encoding/decoding. We can conclude that reducing the amount of XML encoding/decoding can improve SOAP performance by a large amount.

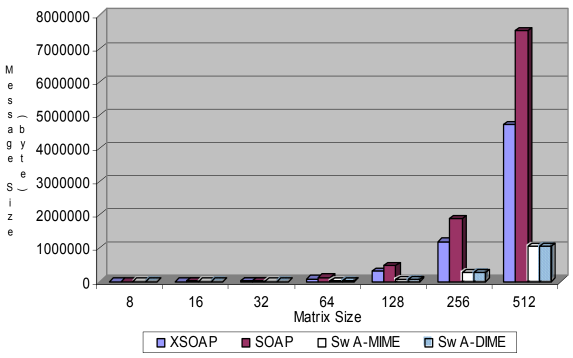

Figure 2: SOAP message size when communicating one floating-point matrix.

	One 128x128 matrix	Two 128x128 matrices
SOAP	1580 ms	2177 ms
SwA-MIME	571 ms	602 ms
SwA-DIME	75 ms	130 ms
XSOAP	90 ms	180 ms

Table 1: Client-side serialization time when communicating one and two 128x128 matrices

	One 128x128 matrix	Two 128x128 matrices
SOAP	1077 ms	1728 ms
SwA-MIME	39 ms	49 ms
SwA-DIME	13 ms	35 ms
XSOAP	20 ms	40 ms

Table 2: Server-side deserialization time when communicating one and two 128x128 matrices

The differences between MIME and DIME should make DIME faster and more efficient in message processing (see Section 2), and this is supported by our data. From the results shown in Figs. 1 and 2, SwA-MIME and SwA-DIME have similar message sizes, but SwA-DIME is always faster than SwA-MIME. In Table 1, we can see that it takes SwA-MIME more than 10 times longer than SwA-DIME to perform the serialization when communicating one 128x128 floating-point matrix, and 6 times longer when communicating two 128x128 floating-point matrices. In Table 2, we also find there is not much difference on server-side deserialization between SwA-MIME and SwA-DIME. We now consider the reasons making these differences and why SwA-DIME has better performance than SwA-MIME.

When we look at the roundtrip time and serialization and deserialization overheads, it is quite obvious that DIME provides a faster and more efficient message processing mechanism. In DIME the data record length for each record is specified in the message header, while in MIME the start and end of a data record is marked by the occurrence of a particular unique string. This string is defined at the beginning of the MIME message, and then an application scans the data in the message to find matching instances of the string. Thus, when the SOAP application serializes a message, it has to go through all the record data to look for the separators to determine the length of the message. While using DIME, it could be simply achieved by using the data in the record headers to quickly walk through and count the number of records and the length of the message without having to read any record data. Meanwhile memory allocation is also more straightforward for DIME than MIME. For DIME, because the lengths of the data are already known, it is simple to allocate buffers for the given size and stream the data into them. For MIME, which does not have to include a data length, the incoming buffers will be allocated without knowledge of how much data is being received. Applications will either have to guess at the size of the incoming data and deal with the inefficiencies of expanding heap allocations, or establish an additional requirement on the sorts of MIME messages the application will receive. But when the server deserialize the message, all these drawbacks of MIME could be avoided because the size of the message has already been defined in the SOAP message. These differences could explain why MIME has a significant serialization overhead, but not deserialization overhead, and also prove why SwA-DIME has better performance than SwA-MIME.

We have also tracked the actual TCP transactions for the MIME and DIME attachment formats to see the efficiency of their encapsulation mechanisms. We find that when sending a SOAP message with a 128x128 floating-point matrix as an attachment, MIME actually sends 13 packets over the network while DIME only sends 4 packets. As mentioned above, if there is only one attachment the difference in SOAP message size between MIME and DIME is always around 310 bytes no matter how large the attached matrix is. So why do MIME and DIME send different numbers of packets over the network? We notice that there is no difference between the first two packets, which is the HTTP binding request and HTTP binding information, which includes remote HTTP location, content-type, content-length, etc. But from the third packet on, we can see the difference. MIME sends a packet which is about 166 bytes, and then another 9 packets which represent the content of the attachment, and finally a packet which is about 43 bytes. The 166 bytes packet is actually the envelope part of the SOAP message, which includes all the processing information for the SOAP message and references to all the attachments. The last 43-bytes packet is the flag for the end of the attachment. Of the 9 packets containing the actual binary data of the attachment, we find the maximum size of each packet is 8192 bytes. Comparing with MIME, DIME encapsulates all the parts, including the SOAP envelope and attachments to send in one message. In this case, it is transmitted in 2 packets over the network, and we find that the maximum size of each packet is 65536 bytes. When we further track the TCP transactions when using both MIME and DIME to perform the task of sending a 256x256 and a 512x512 floating-point matrix, we find DIME only uses 7 and 19 packets, respectively to complete the transaction while MIME uses 38 and 133 packets, respectively. This also shows that the encapsulation mechanism provided by DIME supports faster and more efficient message processing and transactions.

3.2.2 Struct Tests

In the struct tests we again see the advantage of using attachments reflected in the roundtrip time in Figure 3, but there is not much difference in the size of the SOAP message shown in Figure 4. We have previously discussed the effect of XML encoding on the SOAP message size, but here we find that the overhead of constructing a SOAP message with a large number of attachments is significant compared with constructing the SOAP message directly, when the size of the raw data is not too big. For SwA-MIME and SwA-DIME we could not see much difference in SOAP message size when performing the test of sending floating-point number matrices with different sizes, because in this case we only use one attachment with each SOAP message. In the struct tests, each struct is sent as an attachment when using MIME and DIME. We find the overhead of constructing an attachment using MIME is around 540 bytes while using DIME it is around 440 bytes.

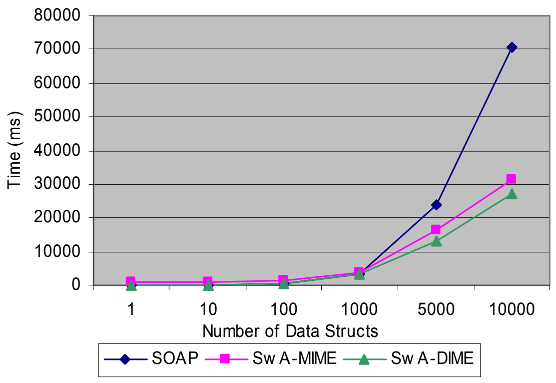

Figure 3: Roundtrip time of communicating different numbers of data structs.

The results of the roundtrip time are qualitatively similar to the timings for communicating floating-point matrices of different sizes. However, it is surprising that when we investigate the overhead of serialization on the client-side and of deserialization on the server-side when communicating 5000 and 7000 data structs, we find that standard SOAP has a significant deserialization overhead that accounts for up to 75% of the latency time. These results are displayed in Table 3 and 4. To find out the reason for this we examined the SOAP message itself. The data struct here is represented by a Vector object, and all the Vector objects are stored in a Vector array. The following is the XML document for representing an example data struct:

<multiRef id="id4212" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns194:Vector" xmlns:ns194="http://xml.apache.org/xml-soap" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item xsi:type="xsd:string">This is a Struct Test for SOAP</item>
    <item xsi:type="xsd:int">28</item>
    <item xsi:type="xsd:boolean">true</item>
</multiRef>

As pointed out previously, standard SOAP adds a tag to each element of the matrix after encoding when performing the test of communicating floating-point matrices. However, in this case, in addition to the tag for each data struct, another tag is added to represent the complex structure. We also notice that the extra tag has its own definition of encoding-style and namespace. When the server-side deserializes the SOAP message, every time it has to interpret the information in each tag and decide the right way to decode the specific tag. This is the reason why we have a significant deserialization overhead in this case, but not in the previous test.

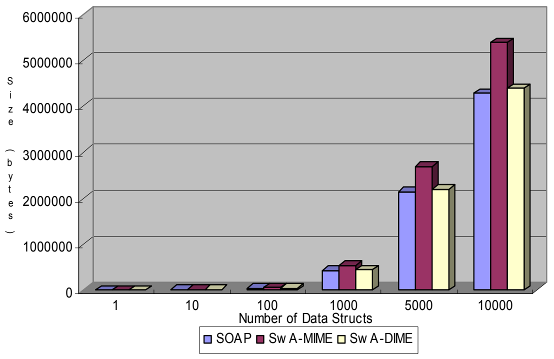

Figure 4: SOAP message size when communicating different numbers of data structs.

	5000 data structs	7000 data structs
SOAP	3584 ms	4180 ms
SwA-MIME	7366 ms	10274 ms
SwA-DIME	3874 ms	6388 ms

Table 3: Client-side serialization time when communicating 5000 and 7000 data structs

	5000 data structs	7000 data structs
SOAP	13534 ms	33520 ms
SwA-MIME	380 ms	614 ms
SwA-DIME	316 ms	540 ms

Table 4: Server-side deserialization time when communicating 5000 and 7000 data structs

4. Conclusions

We have undertaken performance studies of standard SOAP, SwA-MIME and SwA-DIME using workloads that reflect a range of typical complex data types for scientific computing.

Our tests indicate that SOAP performance when dealing with complex data types and large amounts of data could be improved by simply using SwA. XML encoding and decoding is both a time-consuming and memory-consuming task that can be reduced to a minimal level.

Our tests also indicate that SwA-DIME is a faster and more efficient message processing approach than using MIME even though they appear to provide the same functionality. The DIME approach to specifying data record lengths in the header, rather than using a string to mark the start and end of a record as in MIME, makes parsing faster and memory allocation more efficient. DIME has a small, fixed set of headers that provide the functionality required by a SOAP message. This makes it easier for tools for XML Web services to be developed to support DIME, while decreasing the risk of interoperability issues, and also makes processing fast and efficient.

As a new approach to sending SOAP messages, SwA-MIME is already supported by Web Service Definition Language (WSDL), and an extension to WSDL to support WS-Attachments using DIME has been proposed by Microsoft. This would allow the original type of data object to be properly defined in processing both attachment formats. Support in WSDL ensures the precision of converting data objects during transactions and also improves its interoperability with other implementations.

References:

[1] Ying Ying, Yan Huang and David Walker, Using SOAP with attachments for e-Science,[www] http://www.allhands.org.uk/2004/proceedings/papers/206.pdf

[2] Robert A. van Engelen, Pushing the SOAP Envelope with Web Services for Scientific Computing, In proceedings of the International Conference on Web Services (ICWS), 2003, pages 346-354.

[3] W3C, SOAP Messages with Attachments, [www] http://www.w3. org/TR/2000/NOTE-SOAP-attachments -20001211

[4] Microsoft, Direct Internet Message Encapsulation (DIME) Draft, [www] http://msdn.microsoft.com/library/en-us/dnglobspec/html/draft-nielsen-dime-02.txt

[5] Aleksander Slominski, Design of a Pull and Push Parser System for Streaming XML, [www] http://www.extreme.indiana.edu/xgws/papers/xml_push_pull/index.html

[6] Dan Davis and Manish Parashar, Latency Performance of SOAP implementation, In 2nd IEEE International Symposium on Cluster Computing and the Grid, 2002, [www] http://www.caip.rutgers.edu/TASSL/Papers/p2p-p2pws02-soap.pdf

[7] Kenneth Chiu, Madhusudhan Govindaraju, and Randall Bramley Investigating the Limits of SOAP Performance for Scientific Computing, In Proceeding of the 11th IEEE International Symposium on High-Performance Distributed Computing, 2002, [www] http://www.extreme.indiana.edu/xgws/papers/soap-hpdc2002/soap-hpdc2002.pdf

[8] Madhusudhan Govindaraju, Aleksander Slominski, Venkatesh Choppella, Randall Bramley and Dennis Gannon, Requirement Requirements for and Evaluation of RMI Protocols for Scientific Computing, [www] http://www.Extreme.indiana.edu/xgws/papers/sc00 paper/

[9] Alex Ng, Shiping Chen and Paul Greenfiled, An Evaluation of Contemporary Commercial SOAP Implementations, In Proceeding of the 5th Australasian Workshop on Software and System Architectures, [www] http://mercury.it.swin.edu.au/ctg/AWSA04/Papers/ng.pdf

[10] Christopher Kohlhoff and Robert Steele, Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems, Journal of Computer Systems, Science and Engineering, Vol 19 No 4 July 2004, pages 241—251

[11] Henrik Frystyk Nielsen and Hervé Ruellan, SOAP 1.2 Attachment Features,[www] http://www.w3.org/ TR/soap12-af/

[12] Henrik Frystyk Nielsen, Erik Christensen and Joel Farrell, Specification: WS-Attachments, [www] http://www-106.ibm.com/developerworks/webservices/library/ws-attach.html

[13] Jeannine Hall Gailey, Sending Files, Attachments, and SOAP Messages Via Direct Internet Message Encapsulation, MSDN Magazine, [www] http://msdn.microsoft.com/msdnmag/issues/02/12/DIME /default.aspx

[14] Neil O'Toole, Nettool, [www] http://www.capescience.com/articles/using_nettool/

[15] Ethereal, [www] http://www.ethereal.com/

[16] Nayef Abu-Ghazaleh, Michael J. Lewis, Madhusudhan Govindaraju, Differential Serialization for Optimized SOAP Performance, In proceedings of 13th International Symposium on High Performance Distributed Computing (HPDC), Honolulu, Hawaii, pp: 55-64, June 2004.