# Chapter 9

# 9 CONCLUSIONS AND FUTURE WORK

## 9.1 Summary of the Observations Made

This study has provided reasons to come to the following conclusions:

The Benchmark represents a reasonable cross section of the real world because its ranking of the Web service frameworks tested is consistent with Industry view. [22]

The round trip time for a SOAP message is at least affected by the complexity of the SOAP message and the payload it is carrying. A framework that is good at handling complex SOAP messages may not deal with higher payloads equally well. The difference in the values of the coefficients of complexity and payload terms statistically substantiate this

A benchmark that utilizes a set of real world type data sets will give a more accurate picture about the performance of the frameworks rather than theoretical data sets. Again the values of the coefficients in the equations pertaining to the real world and hypothetical benchmarks provide evidence.

## 9.2 Future Work

This study has the potential to be extended in numerous ways. Firstly Web services will continue to evolve resulting in their penetration into the still untapped areas which will definitely use innovative data structures in the transactions. Therefore it will be necessary to continue to update the initial data set presented here in our real world type benchmark, in order to make the data set compatible with data structures used in contemporary real world Web services. Secondly, as far as this study is concerned frameworks have been tested with respective to their RTT only. However another major factor that governs the performance of a framework is its scalability. Therefore measuring the throughput of Web services under busy conditions will give a better understanding of their performances. Thirdly, another study, important from a scientific perspective, would be using the same benchmark to compare the performance of the other Web service frameworks (especially Microsoft's .Net framework) as our investigation is limited to Java based tools. Fourthly extending the investigation to use the real world type benchmark to compare SOAP based Web services with other technologies such as CORBA would provide better insight. Elfwing et el has done a similar comparison using theoretical datasets in [3]. Fifthly this study has implemented the scenario, where client and server run on the same machine but at many practical situations the case is almost entirely different. Introducing probable network delays, which are characteristic of various scenarios, will contribute toward understanding the real performance of the frameworks under various actual conditions. Our effort, which was basically aimed at developing a benchmark that can depict the real word scenarios in an acceptable manner, has not gone into these various implementation specific directions but future studies on them will produce useful results.

How the Benchmark can be enhanced to make it more compatible with the developments that have taken place with respect to the Web service stack after the implementation of the Benchmark has been discussed in the next chapter.

# Chapter 10

# 10 RECENT DEVELOPMENTS IN WEB SERVICE STACK

This chapter discusses the impact of some of the recent developments, which have taken place with respect to the Web services stack, on the Benchmark described in the preceding chapters.

## 10.1 From RPC to Document Style

The existing Benchmark uses RPC as its SOAP binding style because it was more popular then than the Document style which was basically for message oriented communication. However, document exchanges are becoming more popular due to interoperability issues, etc. and therefore the Benchmark should cater to such developments. Benchmarks should reflect contemporary industry practices if they want to be regarded as true representations of the real world and therefore benchmarks should evolve with the changing patterns of usage.

## 10.2 Message Exchange Patterns

With WSDL 2.0 [21], Message Exchange Patterns (MEP) have come to prominence and accordingly, a set of scenarios are described here where MEPs can be seen in operation. While WSDL 2.0 specification presents 8 predefined MEPs, it allows any organization to define new Message Exchange Patterns if it is able and willing to do so. The following scenarios, described with respect to a Supermarket Chain, explain how various predefined MEPs can be helpful

1. In-Only

   This represents a receipt of a message of non-critical nature. For example a message from a staff welfare society targeted at employees can take this form. Even if the message generates a fault it will not be propagated

2. Robust In-Only

   This represents receipt of a critical message. If a fault is generated it should be propagated in the opposite direction. For example a Good Receive Notice (GRN) from a branch outlet received by a warehouse can be of this pattern. If the GRN generates an error, the branch outlet will most probably receive an error message.

3. In-Out

   The Supermarket Chain can receive an offer from a supplier to purchase items at a discounted price. The Supermarket Chain should respond whether it is going to accept the offer or not. If a fault is generated the error message will replace the outgoing message.

4. In-Optional-Out

   A warehouse of the Supermarket Chain can receive information about possible increased demand for a certain item from a branch outlet. If that particular item is not available only the warehouse will inform the branch outlet about its inability to meet the demand. However if a fault message is generated the warehouse can send it to the branch outlet

5. Out-Only

   The Supermarket Chain can send messages related to its promotional campaigns to its partners. No fault messages are expected

6. Robust Out-Only

   This represents sending out a critical message. If a fault is generated it should be propagated back to the sender. For instance, a message involved in informing the head office regarding shortage of goods at a branch outlet can be of this pattern.

7. Out-In

   The Supermarket Chain can send a purchase order to a supplier and it can receive the corresponding invoice. If a fault is generated the supplier can send an error message instead of the invoice.

8. Out-Optional-In

   A branch outlet of the Supermarket Chain can inform the head office about a possible increase in labour requirement during a given season. If the head office is unable to provide additional employees only it will inform the branch outlet. If a fault message is generated the head office will send it to the branch outlet.

The Benchmark can be enhanced by implementing the above mentioned various scenarios because it can then represent novel usages of Web services in the real world in a closer manner.

## 10.3 Making a Web service available at different endpoints over diverse protocols

With WSDL 2.0, a service can be made available at different endpoints and this in turn helps making the service made available over different transport protocols. This provides significant benefit for benchmark development because it facilitates checking performance of Web services under varying conditions thus making it possible for the users to select most optimum implementation-al details.

The performance of a Web service does not solely depend on the strength of the Web service framework used to make the service available to the outside world. The network conditions, pros and cons of the protocols used, etc also play a significant role in the ultimate service provided to the customers. Therefore to test the performance under varying real world conditions, the WSDL 2.0 provides an enhanced support.

## 10.4 Testing for Non-Functional Requirements

WSDL 2.0 enables describing non-functional characteristics of a service and this feature can help in the situations where services with such characteristics needs to benchmarked. For instance, if a service needs to be exposed with "reliability", "security", "correlation" or "routing" `features` then WSDL 2.0 enables associating such non-functional characteristics with the WSDL description. If a service needs to be delivered over a secure channel, then that `feature` can be added to the `interface`.

It can be mentioned as an additional example that if a service needs to have ACID properties then `feature` component can be added indicating that transactional abilities are required. If a Banking Service is considered then more often than not, under real world situations, a service with the roll back capabilities is preferred to a service without such abilities and WSDL 2.0 enables developing test cases to benchmark services with such characteristics.

## 10.5 No Support for Operation Overloading

WSDL 2.0 has removed support for operation overloading but this change does not demand modifying the Benchmark developed because in the implemented Benchmark, operation overloading has been avoided as it was not in the spirit of interoperability, which is the primary rational for going for Web services.

## 10.6 Measuring Performance of Evolving Web Services

In WSDL 2.0, `interfaces` are consisting of `operations` and it is possible to extend `interfaces` by adding `operations`. The ability to create complex interfaces by appending additional operations is significant in testing performance of Web services while they are evolving. Everything should evolve with time if they want to be persistent and so should software and Web services.

Today one can expose a Web service which is having an `interface` consisting of limited number of `operations`, but when the business grows it is natural that new `operations` should come in to play in addition to the existing ones. This can be achieved by inheriting previous interfaces and the capability to inherit preceding `interfaces` provides a window of opportunity to observe the changes in performance during the life cycle of a Web service.

## 10.7 Support for Multiple Inheritance

WSDL 2.0 supports multiple inheritance and the diamond problem, which is due to the ambiguity that arises when two classes inherit from a third common class and when a fourth class inherit from the first two classes, is solved by merging equivalence operations.

# 11 REFERENCES

[1] Cohen, F., Discover SOAP *Encoding's impact on Web service performance*, http://www-106.ibm.com/developerworks/webservices/library/ws-soapenc/, last accessed, June 2004.

[2] Cohen, F., *Performance testing SOAP-based applications*, http://www-106.ibm.com/developerworks/webservices/library/ws-testsoap/, last accessed June 2004.

[3] Elfwing, R., Paulsson, U., Lundberg, L., *Performance of SOAP in Web Service Environment Compared to CORBA*, Proc. Ninth Asia-Pacific Software Engineering Conference (ASPEC'02)

[4] Chiu K., Govindaraju M., Bramley R., *Investigating the Limits of SOAP Performance for Scientific Computing*, Proc. 11th IEEE International Symposium on High Performance Distributed Computing

[5] Davis, D, Parashar M., *Latency Performance of SOAP Implementations*, Proc. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid

[6] Christopher Kohlhoff et al, *Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems*, Proc. The 12th International World Wide Web Conference

[7] Robert A. van Engelan, *Pushing the SOAP envelop with Web services for scientific computing*, Proc. 1st International Conference on Web Services

[8] Ferguson, D., Storey, T., Lovering, B., Shewchuk, J., *Secure, Reliable, Transacted Web services*, http://www-3.ibm.com/software/solutions/webservices/pdf/SecureReliableTransactedWSAction.pdf, last accessed June 2004.

[9] Tuecke, S., Foster, I., Frey, J., Graham, S., Kesselman, C., Maquire, T., Sandholm, T., Snelling, D., Vanderbilt, P., *Open Grid Services Infrastructure*, http://xml.coverpages.org/OGSI-SpecificationV110.pdf, last accessed June 2004.

[10] Bhutta, K. S. and Haq, F., *Benchmarking Best Practices: an Integrated approach*, Benchmarking: An International Journal, Vol. 6 No. 3, 1999, pp 254-268

[11] http://www.w3.org/TR/xml/, last accessed, June 2004

[12] http://www.w3.org/TR/soap/, last accessed, June 2004

[13] http://www.w3.org/TR/wsdl, last accessed, June 2004

[14] http://ws.apache.org/axis/, last accessed, June 2004

[15] http://www.systinet.com/, last accessed, June 2004

[16] http://www.webmethods.com, last accessed, June 2004

[17] http://jakarta.apache.org/tomcat/, last accessed, June 2004

[18] http:// www.xmlrpc.com, last accessed, June 2004

[19] http://www.java.sun.com/xml/jaxrpc/index.jsp, last accessed, June 2004

[20] http://www.spss.com, last accessed, June 2004

[21] http://www.w3.org/TR/wsdl20/

[22] Wickramage, N., Weerawarana, S., *A Benchmark for Web Service Frameworks*, Proc. 2005 IEEE International Conference on Services Computing, pp 233-242