Quality of Services Requirements for Web Services

Aastha Singh *, Arun Singh Yadav **

* Department of Computer Science and Engineering, Goel Institute of Technology and Management, India
** Department of Computer Science, Goel Institute of Higher Studies, India

Abstract- This document describes quality-of-service (QoS) requirements for web services. With the dispersal of web services as a business solution in enterprise application, the importance of QoS for web services is increasing rapidly to the service providers and the clients. Providers need to specify and guarantee the QoS in their web services to remain competitive and achieve the highest possible revenue from their business while the clients desire to have a good service performance in terms of very high availability, shorter response time, accuracy, etc. However, due to the dynamic and unpredictable characteristics of the web services, it is not an easy task to provide the desired QoS for web service users. Even, different web service applications with different QoS requirements will compete for network and system resources such as bandwidth and processing time. But, an enhanced QoS for a web service will bring competitive advantage for service provider. To provide such a better QoS, it is first necessary to identify all the possible QoS requirements for web services, which is the objective of this document.

Index Terms- Web Service, QoS requirements, e-business, Service-oriented Architecture.

I. Introduction

With the widespread proliferation of Web services, quality of service (QoS) will become a significant factor in distinguishing the success of service providers. QoS determines the service usability and utility, both of which influence the popularity of the service. In the emerging electronic environment, knowing how to measure Web service quality is of great importance. Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. With the integration of Web Services as a business solution in many enterprise applications, the QoS of Web Services is becoming the main concern of both service providers and clients. Providers need to specify and guarantee the QoS in their web services to remain competitive and achieve the highest possible revenue from their business while the clients desire to have a good service performance (e.g. high availability, short response time, etc.)[1].

Delivering QoS on the Internet is a critical and significant challenge because of its dynamic and unpredictable nature. Applications with very different characteristics and requirements compete for scarce network resources. Changes in traffic patterns, denial-of-service attacks and the effects of infrastructure failures, low performance of Web protocols, and security issues over the Web create a need for Internet QoS standards. Often, unresolved QoS issues cause critical transactional applications to suffer from unacceptable levels of performance degradation [2].

With standards like SOAP, UDDI, and WSDL being adopted by all major Web service players, a whole range of Web services covering the financial services, high-tech, and media and entertainment are being currently developed. As most of the Web services are going to need to establish and adhere to standards, QoS will become an important selling and differentiating point of these services[3][4].

Web services QoS can be described as a set of nonfunctional attributes that may impact the quality of the service offered by a Web service. Quality-of-Service(QoS), which is usually employed for describing these non-functional characteristics, has become an important differentiating point of different Web services [1].In the field of service computing, Web service QoS have been discussed in a number of research investigations for presenting the non-functional characteristics of the Web services ([1][5][6][7][8][9]. [10] employ five generic QoS properties (i.e. execution price, execution duration, reliability, availability, and reputation) for dynamic Web service composition. [11][15] use five QoS properties (i.e., execution time, availability, price, reputation, and data quality) when making adaptive service composition in flexible processes. [9] Propose an efficient service composition approach by considering both generic QoS properties and domain-specific QoS properties. A wide variety of QoS parameters for web services have been presented in previous work ([1][12][4][13][14]). This work presents an overall description of requirements for measuring the quality of service (QoS) of Web services.
.

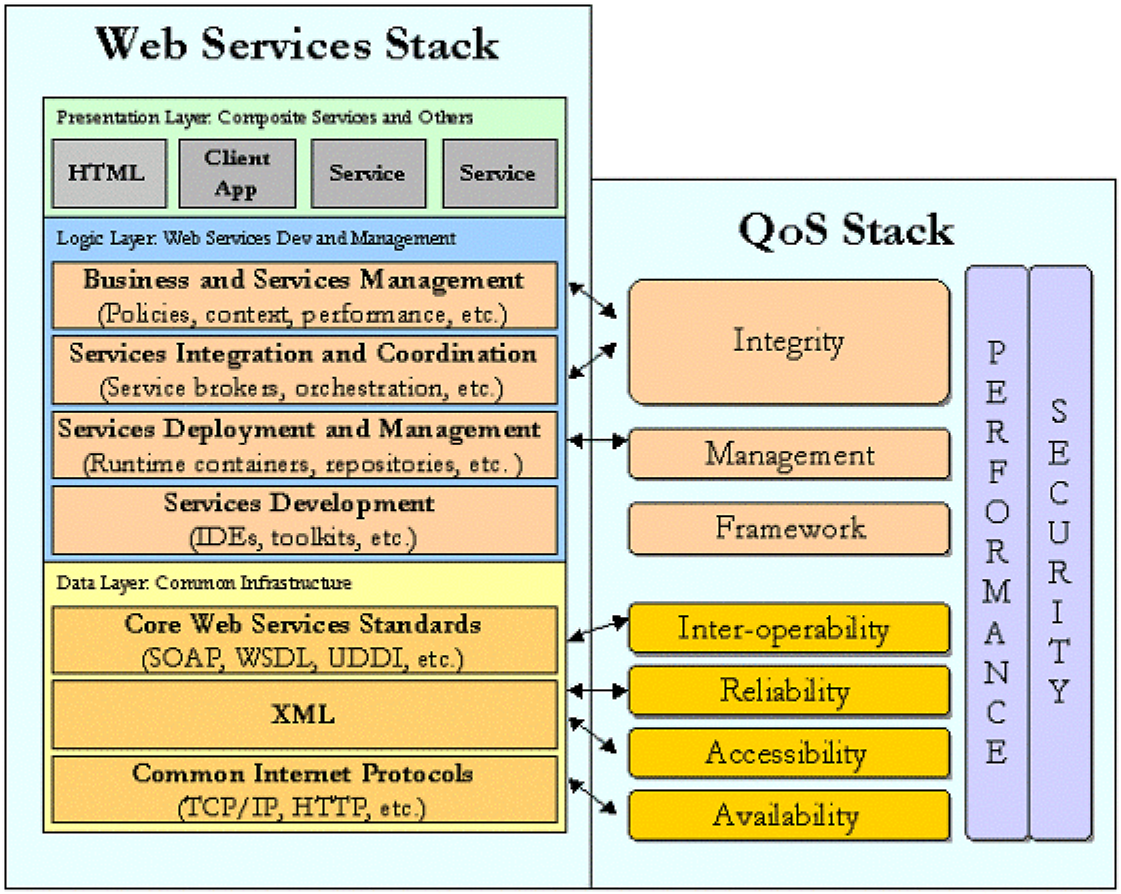
Web Services Stack
Presentation Layer: Composite Services and Others
HTML
Client App
Service
Service
Logic Layer: Web Services Dev and Management
Business and Services Management (Policies, context, performance, etc.)
Services Integration and Coordination (Service brokers, orchestration, etc.)
Services Deployment and Management (Runtime containers, repositories, etc.)
Services Development (IDEs, toolkits, etc.)
Data Layer: Common Infrastructure
Core Web Services Standards (SOAP, WSDL, UDDI, etc.)
XML
Common Internet Protocols (TCP/IP, HTTP, etc.)
QoS Stack
Integrity
Management
Framework
Inter-operability
Reliability
Accessibility
Availability
PERFORMANCE
SECURITY

Fig. 1 Web Services *QoS* "Stack"

II. QOS REQUIREMENT FOR WEB SERVICE

This section describes the major QoS requirements for good quality of web services:

A. *Performance*

In The performance of a Web service concerns how fast a Web service request can be processed and serviced. This requirement can be determined by measurements like throughput and latency. Throughput is the measure of the number of requests serviced in specific amount of time. Latency is the amount of delay that is experienced by a client starting from the time when the request is submitted to the time the response information starts to arrive. The request time is the time when the client submits a request to the Web server, and the response time is the time when the Server replies after processing the request. Obviously, the response time and the throughput depend on the workload that the Web server is experiencing at that time. Both latency and throughput can be measured by keeping track of the timestamps at the request time and response times. The latency of any request processing is the difference between the timestamps corresponding to the request time and the response time, and the throughput is given by the total number of requests divided by the elapsed time between the request time and the response time.

Limitations

The overall performance of Web Services depends on application logic, network, and most importantly on underlying messaging and transport protocols, such as SOAP and HTTP, that it uses. The SOAP protocol is still maturing and harbors a lot of performance and scalability problems. The SOAP protocol uses a multistep process to complete a communication cycle.
The SOAP request begins with the business logic of your application learning the method and parameter to call from a Web Services Description Language (WSDL) document. This whole process is a time consuming one, which requires various levels of XML parsing and XML validation and hence hits the performance of the Web Service.

Optimal Practices

Analyzing the above situation and looking in to SOAP and other related standards, as of now we cannot get away with the steps of XML processing. Instead, we can optimize the XML processing steps.

Some of the best practices to be followed to achieve the same are listed below:

- Use of efficient and lightweight parsers.
- Efficient use of XML validation in production mode.
- Use of compressed XML for sending the messages over network.
- Web Service Caching
- Use of simple data types in SOAP messages as far as possible.

B. Reliability

Reliability requirement measures the quality of a Web service in terms of how it performs, given a specific amount of time and the present network conditions, while maintaining the service quality. It is also determined by the number of failures per day and the medium of delivery. In fact, Web Service Reliability (or WS Reliability) is a latest specification for open, reliable Web service messaging. WS-Reliability ensures guaranteed delivery of messages, elimination and/or detection of duplicate messages and the right ordering of messages. The WS-Reliability can be embedded into SOAP as an additional extension rather than to a transport level protocol. This specification provides reliability in addition to interoperability, thus allowing communication in a platform and vendor-independent manner. The WS Reliability specification defines a set of SOAP headers and instructions in SOAP envelopes that manage the message acknowledgements, message ordering etc. WS- Reliability makes asynchronous messaging a healthy choice because of the extra features that it provides like message acknowledgements and message ordering which in turn allow the communicating parties to be more independent of one another [16]. Reliability determines the percentage of the times an event is completed with success. This numeral will provide an opportunity for the service consumers to expect the probability of a failure that can occur during a transaction. The count on failures is based on the number of dropped deliveries, duplicate deliveries, faulty message deliveries, and out-of-order deliveries. An event may either succeed or fail, and there is no middle ground in that issue. Therefore, total number of events will be the number of failures added to the number of successful events.

Limitations

The Web Services currently rely on transport protocols such as HTTP, which are inherently stateless and follow a best effort delivery mechanism. It does not guarantee whether the message will be delivered to the destination.

Optimal Practices

The above problem rising due to the use of unreliable protocols can be solved using the following techniques:

- Use of asynchronous message queues.
- Adoption of new reliable transport protocols (such as HTTPR, REST, and BEEP).

C. Scalability

Scalability requirement defines how expandable a Web service can be. Web services are being introduced to new interfaces and techniques everyday and this makes keeping a Web service up-to-date a very important necessity. If the situation demands for more computing capabilities and servicing more requests, the system should be capable of supporting additional systems and newer technologies. The Web service should be able to handle heavy load while making sure that the performance in terms of response time experienced by their clients is not objectionable. The Performance Non-Scalability Likelihood (PNL) metric is a relatively newer technique to predict whether the system is going to be able to withstand the higher loads of traffic without affecting the performance levels. This metric is used to calculate the intensity of the loads at which the system cannot perform without degrading the response time and throughput. The calculation of PNL involves generating potential workloads and studying the behavior of the system which will be similar to how the system would react given such varying workloads. If the system crashes, the engineers will know that it is not scalable enough to accommodate potential future workloads and they could eventually organize an upgrade to the server capability. There can be two states in terms of their behavior. The behavior state can either be acceptable (0) or unacceptable (1). We can also denote the states with relative values in between 0 and 1 (in the interval [0, 1]) instead of constant values to point out the degree at which the system fails to behave acceptably.

Limitations
Building scalable systems are expensive, and this may cause smaller companies to defer this requirement. Also, this becomes an infrastructure issue for companies that deploy Web Services within their enterprise.

Optimal Practices

- Service pooling
- Load balancing (Scalability)

D. Accuracy

Accuracy requirement is a measure of correctness delivered by a Web service. The number of errors generated by a Web service, the number of fatal errors, and the frequency of the situation determine the amount of accuracy for a Web service. Accuracy is defined as the level to which Web services give accurate results for the received requests. Accuracy refers to whether the measurement or metric really measures what we intend to measure. An experiment is conducted to measure the accuracy of the Web service by calculating the standard deviation of the reliability. As the average value of the standard deviation is equal to zero, the measurement is said to be accurate. If the average value of the standard deviation is very high, then the measurement is considered to be not accurate.

E. Integrity

Integrity requirement assures that any modifications to a Web service are performed in an authorized manner. Data integrity assures that the data is not corrupted during the transfer, and if it corrupted, it assures that there are enough mechanisms in the design that can detect such modifications. Data integrity is an important element to consider, because ignoring it may damage large software modules and create errors that are impossible to trace back. Data integrity is the measure of a Web service's accurate transactional and data delivery abilities. The data messages that are received are verified to see if they have not been modified in transit. This can be done with techniques like checksum calculation or digital signatures. There are a number of tools in the market like SIFT [17] that can collect and monitor the data being sent and received between the communicating parties. These tools can be used to monitor the number of faulty transactions that are unidentified and the data messages that are received but with the checksum or hash that cannot be tallied. Data integrity is a boolean value, meaning that data either has integrity or does not. There is no middle ground or a range that can specify how much integrity the data holds. Data integrity can therefore be calculated as the ratio of successful transactions to the total number of transactions.

$$Integrity = \frac{Number\ of\ successful\ transactions}{Total\ number\ of\ transactions}$$

Limitations
Data Integrity is important for any object's proper functioning and must be assured or it could corrupt a larger program and generate a very difficult to trace error. Web Services transactions tend to be asynchronous and long running in nature. Transaction integrity is just one of several QoS elements, including security and process orchestration, which are missing from the first incarnations of Web Services standards of SOAP, UDDI, and WSDL.

Optimal Practices
Emerging standards in the business process management and transactions will help to achieve the desired QoS.

- Adopting standards such as BPEL4WS, WS Coordination ,WS Transaction ,and BTP would benefit service providers.

F. Availability

Availability requirement is the probability that the Web service is up and in a readily usable state. High availability assures that there is the least amount of system failures or server failures even during the peak times when there is heavy traffic to and from the server and that the given service is available relentlessly at all times. Let us say the "down time" is when a system is not available. As the system is either available or unavailable, the remaining time after subtracting the down time can be termed as the "up time" that means the system is available. As checking upon down time is easier than up time (because down time is smaller than uptime), we can consider to calculate down time to measure the availability. Keeping a tab on the events failed during an operation can possibly give the down time [18].

Limitations

Building fault-tolerant systems for highly available Web Services is expensive. As companies roll out Web Services, the ability to manage this diverse, dynamic, distributed environment will become critical. Questions such as the following arise:

- Has one of my key servers become unavailable?
- Is a system being overly burdened?
- Why are requests taking so long?

Optimal Practices

- Web Service Management
- Web Service Clustering

G. Accessibility

A Web service's accessibility is the measure of the probability that the client's request to a Web service will be served. Accessibility is typically a measure of the success rate of a service instantiation at a given time. A Web service might not be accessible even though it is still available, because a system may be up and running but might not be able to process a request possibly because of the load it is experiencing. Accessibility in turn depends on how scalable the Web service system design is, because a highly scalable system constantly serves the request irrespective of the volume of the Web service requests. Accessibility is the ratio of the number of successful responses received from the server to the number of requests messages sent by the clients. It can be characterized as the degree of a system at which it is capable of responding to the user invocations of the service. Irrespective of type of acknowledgements received, (either negative/positive or correct/incorrect), accessibility can be calculated as a ratio of number of successful acknowledgements received to the total number of requests sent [18].

$$Accessibility = \frac{Number\ of\ acknowledgements\ received}{Total\ number\ of\ requests\ sent}$$

H. Interoperability

Web services are accessed by thousands of clients around the world using different system architectures and different operating systems. Interoperability means that a Web service can be used by any system, irrespective of operating system or system architecture and that accurate and identical result is rendered in any environment. Interoperability is a measure of how flexible the Web service has been created so that the clients do not have to worry about binding to a Web service that cannot be run in their environment. The developmental environment here includes operating system, programming language or hardware type. The inter operability can be calculated as the ratio of the total number of environments the Web service runs to the total number of possible environments that can be used.

$$Interoperability = \frac{Total\ number\ of\ enviorments\ the\ web\ service\ runs}{Total\ number\ of\ possible\ enviroments\ that\ can\ be\ used}$$

This interoperability value measures the successful execution of the Web service in different environments (such as different operating systems, programming languages, and hardware types). The value closer to 1 indicates higher interoperability which is desirable.

Limitations

Most of the Web Services specifications are defined under standards bodies. As these activities are under way, there seems to be a delay in the implementations. Vendors partly implement the specification in their products due to the competitive nature of this market. This results in poor interoperability.

Optimal Practices

Key to enabling seamless Web Services interoperability is the ability of one Web Services framework to consume the WSDL documents generated by other frameworks.

- Web Services-Interoperability (WS-I) Profiles.

The Basic Profile defines how a selected set of specified Web Services technologies, such as messaging and discovery, should be used together in an interoperable manner.

III. CONCLUSION

Quality of services is an important requirement of business to business transactions and thus a necessary element in Web services. The various QoS properties such as availability, accessibility, integrity, performance, reliability, etc, need to be addressed in the implementation of Web service applications. This paper mainly focused on the QoS requirements and its limitations and its possible best practices to get the desired services. As it is discussed that many protocols at different layers of web services architecture affected the performance and other characteristics that results the loss in e-business revenue. The Web services QoS requirements described here do not exhaust the list of yardsticks necessary to measure the quality of service in Web service.

ACKNOWLEDGMENT

We are heartily thankful to all the persons that help us to write and review this article ,the individuals that meet us online in discussion forum .Special thanks to Mr. Rohit Dwivedi sir for his valuable time and effort .We also want to present my gratitude to all the editorial members and the reviewer members of IJSRP journal.

REFERENCES

[1] D. A. Menascé, “QoS Issues in Web Services”, IEEE Internet Computing, pp. 72-75, Dec. 2002, http://csdl.computer.org/comp/mags/ic/2002/06/w6072abs.htm

[2] Kuyoro Shade O., Awodele O. Akinde Ronke O. and Okolie Samuel O, “ Quality of Service (Qos) Issues in Web Services” ,Babcock University, Ilishan-Remo, Ogun State, Nigeria, VOL.12 No.1, January 2012.

[3] Francisco Curbera , Matthew Duftler , Rania Khalaf , William Nagy , Nirmal Mukhi , Sanjiva Weerawarana, “Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI”, IEEE Internet Computing, v.6 n.2, p.86-93, March 2002.

[4] Anbazhagan Manim,Arun Nagarajan. “Understanding quality of service for Web services: Improving the performance of your Web services”. IBM developerWorks. Retrieved November 7, 2005.

[5] M. C. Jaeger, G. Rojec-Goldmann, and G. Muhl. Qos aggregation for web service composition using workflow patterns. In *Proc. 8th IEEE Int'l Conf. Enterprise Computing*, pp 149–159, 2004.

[6] J. O’Sullivan, D. Edmond, and A. H. M. ter Hofstede. What’s in a service? *Distributed and Parallel Databases*,12(2/3):117–133, 2002.

[7] M. Ouzzani and A. Bouguettaya. Efficient access to web services. *IEEE Internet Computing*, 8(2):34–44, 2004.

[8] S. Rosario, A. Benveniste, S. Haar, and C. Jard. Probabilistic qos and soft contracts for transaction-based web services orchestrations. *IEEE Trans. Services Computing*, 1(4):187–200, 2008.

[9] N. Thio and S. Karunasekera. Automatic measurement of a qos metric for web service recommendation. In *Proc.Australian Software Engineering Conference*, pp 202-211, 2005.

[10] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Software Engineering*, 30(5):311–327, 2004.

[11] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Software Engineering*, 33(6):369–384, 2007.

[12] M. Alrifai and T. Risse. Combining global optimization with local selection for efficient qos-aware service composition. In *Proc. 18th Int'l Conf. World Wide Web(WWW'09)*, pages 881–890, 2009.

[13] Liangzhao, “Quality Driven Web Services Composition”, *The Twelfth International World Wide Web Conference* ,Budapest, Hungary, May 2003.

[14] Ying Li, Xiaochen DING, Ying CHEN, Dong LIU, Thomas LI, “The Framework Supporting QoS-enabled Web Services”, *IEEE International Conference on Web Services*, Las Vegas, Nevada, USA, June 2003.

[15] Shuping Ran, “A Framework for discovering web services with Desired Quality of Services Attributes”, *IEEE International Conference on Web Services*, Las Vegas.

[16] Sun, “Web Services Reliability (WS-Reliability) Version 1.0:frequently asked questions (FAQ)”. Sun Developer Network(SDN), January 2003.

[17] B.Barbash, “Sift 1.5 by service integrity”. SOA Web Services Journal, September 2003.

[18] OASIS. “Summary of quality model forWeb services (OASIS Web Services Quality Model TC)”. OASIS, June 2002.

AUTHORS

First Author – Aastha Singh, Department of Computer Science and Engineering, Goel Institute of Technology and Management,U.P. Technical University, India, aasthasingh158@gmail.com
Second Author – Arun Singh Yadav, Department of Computer Science, Goel Institute of Higher Studies, India,arunsinghyadava@gmail.com
.