

DOCTORAL DISSERTATION

QoS integration in Web services with the WS-QoS framework

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

im Fachbereich Mathematik und Informatik

der Freien Universität Berlin

vorgelegt von

Dipl.–Ing. Min Tian

aus Peking

01. Dezember 2005

Gutachter:

Prof. Dr.–Ing. Jochen H. Schiller

Prof. Dr. Stefan Fischer

Tag der Disputation: 29. November 2005

Summary

Web Services are becoming more and more popular these days and more and more businesses are planning to build their future solutions on Web service technology. By now, SOAP and WSDL have become reliable standards in the field of Web service execution. While the concept of UDDI allows for automatic discovery of services implementing a common public tModel interface, there have been only few attempts to find a standardized form to describe the quality of service (QoS) with which services are executed.

The need for QoS specifications in Web services is driven by two demands. Clients aim to experience a good service performance, e.g. low waiting time, high reliability, and availability to successfully use the service whenever the need arises. On the other hand, when it comes to e-business, service providers need to formulate QoS-aware offers in order to gain the highest possible profit from their business. Examples are high throughput guarantees and low response time through dynamic capacity allocation, resource allocation, and load balancing in order to serve a high number of clients with assured QoS. Moreover, crucial transactions such as payment should experience prioritized execution realized by transaction differentiation. Service providers will strive to find an optimal relation between user satisfaction and system utilization.

There are sophisticated technologies to actively differentiate between various QoS levels both on the transport level (DiffServ, IntServ, classes of services in UMTS and ATM, etc.) and on the server level (load balancing, transaction differentiation, HTTP request differentiation, etc.), yet there are no standardized means to describe the desired QoS on the application level. In terms of the Internet model, the Web services can be placed between the application and network layer.

To close the gap between the Web Services layer and the underlying QoS-aware transport technologies, we have developed the Web services-QoS architecture (WS-QoS). The main motivations of our work are

- designing an architecture that allows both service clients and service providers to specify requests and offers with QoS properties and QoS classes,
- enabling an efficient service offer selection in order to accelerate the overall lookup process for the client,
- providing a flexible way for service providers to publish and update their service offers with different QoS aspects as well as
- mapping the QoS requirements regarding the underlying transport network from the higher (Web service and application) layer onto the actual underlying network technology at runtime in order to achieve an overall QoS support through the different layers in terms of the Internet model.

The main contributions of this work are the WS-QoS architecture that enables QoS-aware service specifications, the broker based Web service selection model that enables an efficient QoS-aware service selection as well as the QoS mapping guarantying the assured QoS.

Zusammenfassung

Mit der weiten Verbreitung von Web services (Webdienste) gewinnt diese Technologie mehr und mehr an Bedeutung sowohl in der Forschung als auch in der Industrie. Die Web service Technologie basiert auf offenen und standardisierten Internetprotokollen und erlaubt strukturierte Datenkommunikation über das Internet, unabhängig von der eingesetzten Netzinfrastruktur, Hardware, Betriebssystemen und Programmiersprachen.

Immer mehr konkurrierende Anbieter wie Börseninformationsdienste oder Suchmaschinen bieten ihre Dienste mit ähnlichen Funktionalitäten als Web services an. Durch die Fülle der verschiedenen Dienstangebote fällt es einem Dienstanutzer schwer, sich für einen Dienstanbieter zu entscheiden, da die Entscheidungsgrundlagen fehlen.

Die vorliegende Dissertationsarbeit führt die WS-QoS (Web service quality of service) Architektur ein. Zum einen kann mit ihrer Hilfe sich ein Dienstanutzer anhand bestimmter Entscheidungskriterien zur Laufzeit für ein Dienstangebot entscheiden. Zum anderen kann ein Dienstanbieter seine Dienstangebote anhand bestimmter Entscheidungskriterien anbieten. Die WS-QoS Architektur führt ein XML Schema ein, anhand dessen die Entscheidungskriterien, auch Dienstgüte genannt, einheitlich spezifiziert werden können. Die Spezifikation erlaubt eine schnelle Suche nach den passenden Dienstangeboten durch einen Dienstanutzer zur Laufzeit.

Die vorliegende Arbeit führt Mechanismen ein, die die Abbildung verschiedener Dienstgütekriterien, die oft auf einem hohen Abstraktionsniveau definiert werden, auf die technischen Systeme zur Laufzeit erlauben. Die dynamische Abbildung, was ein weiteres herausragendes Merkmal der Dissertationsarbeit im Vergleich zu anderen Arbeiten ist, garantiert eine durchgehende Unterstützung der Dienstgüte in allen Kommunikationsschichten und Komponenten entlang des Kommunikationspfades. Dies erlaubt eine echte Erfüllung der zugesicherten Dienstgüte.

Die Dissertationsarbeit belegt die Vorteile der eingeführten WS-QoS Architektur durch eine Implementierung und verschiedene Leistungsmessungen und -analysen. Die Arbeit zeigt ebenfalls die Vorteile der vorgeschlagenen WS-QoS Architektur für mobile Webdienste. Durch einfache Erweiterungen der Architektur können Kommunikationsgeschwindigkeiten für mobile Nutzer bei gleichzeitiger Kostensenkung erhöht und Server vor Überlastungen geschützt werden.

Lebenslauf

Vorname	Min
Nachname	Tian
Geburtsdatum	12. Juli 1973
Geburtsort	Peking, China
Staatsangehörigkeit	Deutsch
Familienstand	Verheiratet
Wohnanschrift	Am Großen Rohrpfuhl 27
PLZ	12355
Stadt	Berlin
Tel.	+ 49 30 97988688
Mobiltelefon	+49 179 7755432
E-Mail	sky@skyfield.de

09.1980 – 07.1986	Grundschule in Peking
09.1986 – 12.1987	Gymnasium in Peking
01.1988 – 06.1988	Deutschkurs in Berlin
08.1988 – 07.1992	Dreilindenoberschule (Gymnasium)
08.1992 – 06.1995	Oberstufenzentrum Energietechnik (Gymnasiale Oberstufe), Abitur
10.1995 – 05.2001	Studium an der TU Berlin, Fachbereich Technische Informatik, Diplom- Ingenieur
02.2002 – Dez. 2005	Promotion an der Freien Universität Berlin, Fachbereich Mathematik und Informatik

Table of content

1	<i>Introduction</i>	15
1.1	Targeting issues of this thesis	18
1.2	Contributions	18
1.3	Thesis overview	19
2	<i>Background</i>	21
2.1	Web services	21
2.1.1	Basic interaction model	22
2.1.2	Web service transaction specifications	23
2.1.3	Web service security	24
2.1.4	Web service trust	24
2.1.5	Web service addressing	24
2.1.6	Web services discovery	25
2.2	Quality of service	25
2.2.1	QoS classes	25
2.2.2	Service differentiation in web servers	27
2.2.3	Improvement of the availability of web servers	27
2.3	Mobile computing / Mobile devices	27
2.4	Conclusion	28
3	<i>Related work</i>	29
3.1	State of the art	29
3.2	Approaches towards QoS specification and management for Web services	31
3.2.1	Web service level agreement	31
3.2.2	Web service offering language (WSOL)	33
3.2.3	SLAng	35
3.2.4	UX	37
3.2.5	UDDIe	39
3.3	Conclusion	40

4	<i>The design of the WS-QoS architecture</i>	43
4.1	Requirements for QoS-aware Web services	43
4.2	WS-QoS XML schema for QoS specification	45
4.2.1	QoS Info	47
4.2.2	WS-QoS ontology	49
4.2.3	QoS definition	50
4.2.4	WS-QoS offer definition	51
4.3	QoS-aware service discovery and selection	52
4.4	QoS-aware service invocation	55
4.4.1	Mapping transport priorities to QoS-aware network technologies	57
4.4.2	Adaptive server performance	59
4.4.3	End-to-end QoS support for adaptive message load compression	62
4.5	Conclusion	64
5	<i>The implementation of the WS-QoS framework</i>	65
5.1	A scenario for QoS-aware service selection	66
5.2	WS-QoS XML schema	68
5.3	WS-QoS editor	70
5.4	WS-QoS requirement manager	71
5.5	WS-QoS monitor	71
5.6	Web service broker	72
5.7	QoS mapping	73
5.7.1	QoSProxy for the client side	73
5.7.2	QoSProxy for the server side	74
5.7.3	Mapping table	76
5.7.4	QoS proxies	78
5.8	Adaptive server performance	79
5.9	End-to-end QoS support for adaptive message load compression	80
5.10	Conclusion and discussion	81
6	<i>Applying the WS-QoS framework</i>	83
6.1	Implementing an abstract service description	83
6.2	Managing WS-QoS offers	84

6.2.1	Including WS-QoS files from WSDL extension elements_____	84
6.2.2	Updating WS-QoS offers with the WS-QoS offer manager _____	85
6.2.3	Using the WS-QoS extension to the .NET WSDL generator engine _____	86
6.3	Adaptation of client's QoS requirements _____	88
6.4	Setting up a WSB _____	90
6.5	Creating WS-QoS compliant client applications _____	90
6.5.1	Defining client side QoS requirements _____	91
6.5.2	Creating a proxy class _____	91
6.5.3	Declaring custom attributes on the proxy class _____	91
6.5.4	Adjusting WS-QoS files with the WS-QoS editor_____	96
6.5.5	The WS-QoS requirement manager _____	97
6.6	QoS-aware service selection with the WSB _____	98
6.6.1	Instantiation of the WSB _____	98
6.6.2	Updating the most appropriate WS-QoS offer _____	100
6.6.3	Comparing available offers with the WS-QoS monitor_____	101
6.7	Service invocation_____	102
6.7.1	Logging of WS-QoS SOAP headers _____	102
6.8	Conclusion _____	104
7	<i>Performance measurements _____</i>	<i>105</i>
7.1	Performance impact of Web service overhead _____	105
7.1.1	Web Service overhead _____	106
7.1.2	A dynamic approach for reduction of Web service responses _____	107
7.1.3	Experiments_____	108
7.1.4	Experimental results _____	110
7.2	Performance measurement of the Web service broker _____	118
7.2.1	Testbed and test application_____	118
7.2.2	WSB without database support_____	118
7.2.3	WSB with database support_____	120
7.3	Performance measurement of the QoSProxy _____	122
7.3.1	Testbed and test application_____	122
7.3.2	Test results_____	123
7.3.3	Performance of an ideal QoSProxy _____	126

7.4	Protecting servers from overload with adaptive WS-QoS offers	127
7.4.1	Test application	128
7.4.2	Testbed	129
7.4.3	Conclusion	133
8	<i>Conclusion and future work</i>	135
8.1	Comparison of WS-QoS with existing approaches	136
8.2	Future research	137
9	<i>Appendix</i>	139
10	<i>References</i>	147

List of Figures

Figure 1. A mobile Web service scenario _____	16
Figure 2. Web service protocol stack _____	22
Figure 3. Basic interaction model of Web services _____	23
Figure 4. The five stages of an SLA management lifecycle [32] _____	33
Figure 5. Components of a WSOL service offering _____	34
Figure 6. SLang's service provisioning reference model [30] _____	36
Figure 7. Vertical and horizontal SLAs [30] _____	37
Figure 8. UX architecture [31] _____	38
Figure 9. Client request with QoS requirements in UDDIe [24] _____	40
Figure 10. Participating domains _____	44
Figure 11. Structure of a wsqos element _____	45
Figure 12. Assignment of QoS classes to a service or its operations _____	47
Figure 13. Structure of a qosInfo element _____	47
Figure 14. Structure of a serverQoSMetrics element _____	48
Figure 15. Structure of a transportQoSPriorities element _____	48
Figure 16. Structure of a QoSOntology element _____	49
Figure 17. tMetricDefinition _____	50
Figure 18. Structure of a tQosDefinition element _____	51
Figure 19. Structure of a WSQoSOfferDefinition element _____	52
Figure 20. Further WS-QoS definition references in an include element _____	52
Figure 21. Interactions between the four participating roles _____	53
Figure 22. Reduced interactions between the four participating roles _____	54
Figure 23. QoS requirement in the SOAP header _____	56
Figure 24. QoS mapping in the Adaptation Layer _____	57
Figure 25. A WS-QoS transportQoSPriorities entry _____	58
Figure 26. A mobile Web service scenario _____	59
Figure 27. A WS-QoS serverQoSMetrics element. _____	60
Figure 28. Impact of server load on processing time [18] _____	61

<u>List of figures</u>	<u>X</u>
Figure 29. Impact of server load on throughput [18]	61
Figure 30. XML overhead for a simple request	62
Figure 31. securityAndTransaction entries for compression and decompression algorithms.	64
Figure 32. Implementation of the WS-QoS framework	66
Figure 33. Scenario for the initial request of offers for a specific service type	68
Figure 34. Classes of the WS-QoS API	69
Figure 35. Defining custom QoS properties	70
Figure 36. WS-QoS Monitor surveying the service selection of a sample client	72
Figure 37. QoSProxy on the client side	74
Figure 38. QoSProxy on the server side	76
Figure 39. Example showing the different BW values	78
Figure 40. Proxy applications on both sides	79
Figure 41. A WS-QoS serverQoSMetrics element	80
Figure 42. securityAndTransaction entries for compression and decompression algorithms	81
Figure 43. Classes of the WS-QoS API representing WS-QoS elements	93
Figure 44. Adding a custom server metric declaration with the WS-QoS Editor	97
Figure 45. An instance of the WS-QoS Monitor surveying the service selection for a WS-QoS compliant client	101
Figure 46. Testbed: measurement of Web service overhead	108
Figure 47. iPAQ service time over wireless LAN	111
Figure 48. iPAQ service time over Bluetooth	112
Figure 49. iPAQ service time over emulated GPRS network	113
Figure 50. iPAQ service time over emulated GPRS with poor connectivity	113
Figure 51. Comparison of response time with and without compressing the response	115
Figure 52. Comparison of throughput with and without compressing the response	115
Figure 53. Response time of the dynamic approach	117
Figure 54. Throughput of the dynamic approach	117
Figure 55. Testbed: measurements of WSB	118
Figure 56. Response time of the WSB at different request rate	119
Figure 57. Throughput of the WSB at different request rate	120

Figure 58. Response time of the WSB with database support at different request rate	120
Figure 59. Throughput of the WSB with database support at different request rate	121
Figure 60. The CPU usage of the server hosting the WSB	121
Figure 61. Testbed: performance measurement of QoSProxy	123
Figure 62. Performance impact of QoSProxy on the web server	124
Figure 63. Performance gains with QoSProxy, background traffic: 800 kilobyte/s	125
Figure 64. Performance gains with QoSProxy, background traffic: 1024 kilobyte/s	126
Figure 65. Performance gains with ideal QoSProxy, background traffic: 800 kbyte/s	127
Figure 66. Performance gains with ideal QoSProxy, background traffic: 1024 kbyte/s	127
Figure 67. Testbed: adaptive WS-QoS offers	129
Figure 68. Behaviour of the system in case of increasing CPU load	131
Figure 69. Server in a stable state	132
Figure 70. Behavior of the server in case of decreasing number of requests	133

List of tables

Table 1. Assessment of the introduced approaches _____	41
Table 2. Example of different QoS classes _____	46
Table 3. Mapping table for DiffServ _____	77
Table 4. Response size without and with compression and decompression _____	110
Table 5. Impact of the dynamic approach on the response time of the mobile client __	118
Table 6. Client configuration _____	130
Table 7. Comparison of WS-QoS with other approaches _____	137