CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Institutional Repository of the Freie Universität Berlin

QoS Integration in Web Services

M. Tian
Freie Universität Berlin, Institut für Informatik
Takustr. 9, D-14195 Berlin, Germany
tian @inf.fu-berlin.de

Abstract: With the growing popularity of Web services, a general QoS support for Web services will play an important role for the success of this emerging technology. Unfortunately, current Web service environments do not offer comprehensive QoS support. In this paper, we present an approach that does not only enable the QoS integration in Web services, but also the selection of appropriate services based on QoS requirements regarding server and network performance as well as the mapping of QoS requirements onto the underlying QoS aware network at runtime.

1 Introduction

Today, research activities in applications, Web services, and communication networks are running in many aspects widely independent from each other. In most cases, researchers of applications and Web service technologies assume that existing communication infrastructures provide reliable communication. Furthermore, researchers in middleware, Web services, and applications are not very considerate of the resources provided by the underlying networks. On the other hand, research activities in certain communication architectures and protocols are performed with less attention to requirements of actual applications. Therefore, most applications cannot actively consume the Quality of Service (QoS) that may be supported in the communication networks, and on the other hand common network technologies do not support application-dependent requirements.

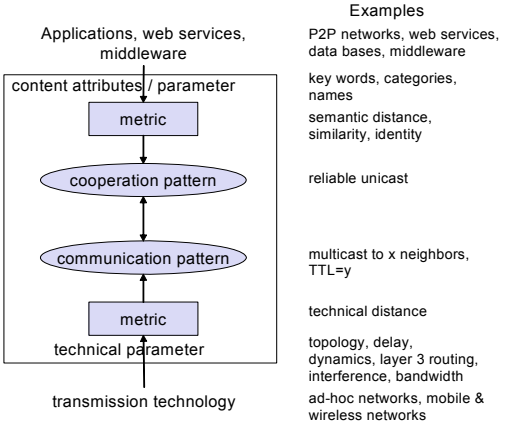

Figure 1 Mapping of applications and services onto communication technology

The demand on highly reliable and highly available Web services increases as more and more companies and customers rely on them to satisfy business and personal needs [MA02]. The growing variety of customers requires a diverse range of QoS support. The QoS a service provider delivers will become a decisive criterion when services with the same functionalities are available at customers' choice.

Nowadays, we have sophisticated technologies and research results regarding QoS support in different domains. They are for example

DiffServ and IntServ for the network layer QoS support, demand-based QoS support through an adaptive end system [Ri01]; QoS aware middleware [Na01], service differentiation in overloaded servers [Vo01]. Most recent efforts on QoS support in Web services are for example IBM's Web Services Level Agreement (WSLA) [Da02] and HP's Web Service Management Language (WSML) [Sa02]. These two languages have been developed to specify Service Level Agreements for Web Services. Electronic contracts are negotiated individually and then surveyed by a monitoring engine. Service offerings defined in the Web Service Offerings Language (WSOL) [To03] provide different predefined classes of service for clients to choose from.

However, most of these approaches neither support the mapping of QoS requirements from higher layers onto the underlying network layer in terms of the Internet model nor considerate the server performance. Figure 1 gives examples for parameters on different layers when mapping applications and services onto certain transmission technologies or when pushing performance parameters from transmission technologies up to applications, respectively. The communication and cooperation between different layers allows an efficient utilization of the underlying network resources as well as a better support of application-dependent requirements.

In this paper, we introduce our current effort tackling the gap between the Web service layer and network layer, as Figure 1 illustrates. We have been developing an architecture that allows the dynamic definition, publication, and matching of both Web service offers and requirements regarding server performance, network performance, security, transaction, pricing as well as customer defined issues at both implementation time and runtime. Our architecture supports the dynamic mapping of requirements regarding the network performance from higher layers onto the underlying network layer at runtime. Furthermore, our architecture allows users to obtain real-time information about server performance in order to prove the accomplishment of assured services. Our approach is extensible and based on Internet standards such as XML schema, SOAP, WSDL, and UDDI. This ensures the independence of any particular programming model and other implementation specific semantics.

The remainder of this paper is outlined as follows. In the next section, we will present the architecture of our QoS aware approach and discuss the specification issues. We conclude with an outlook of future work.

2 Web Service QoS Architecture

We propose QoS support in both the Web service layer and the network layer. By utilizing our system, service providers can augment their Web service offers with QoS aspects while clients can define their requirements related to QoS parameters. QoS parameters such as processing time, request rate, response time, availability, reliability, security protocols, transaction, price, and customer defined parameters are declared for the Web service layer QoS support by clients and servers. Standard and customer defined parameters such as delay, bandwidth, jitter, and packet loss are defined for the network layer QoS support by both parties.

We introduce a Web service broker (WSB) in order to accelerate the client lookup process for services. That means a Web service client will contact the WSB for looking up a service instead of doing this with a UDDI registry. The WSB has then the task of

testing the clients' requirements against the Web service providers' offers. Figure 2 depicts the participating roles service providers, clients, UDDI registries, and the WSB and their interactions.

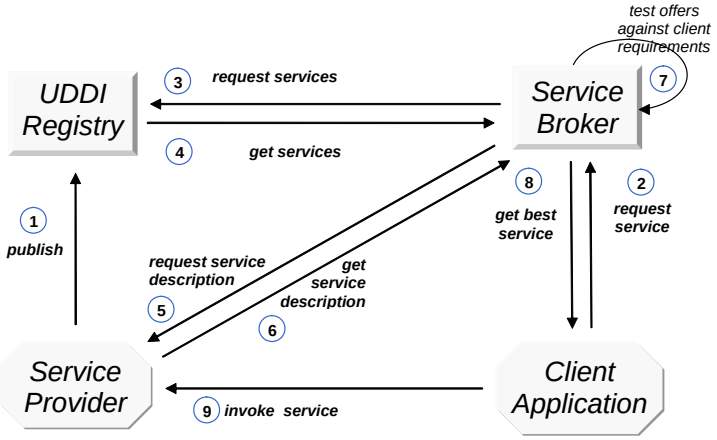

Figure 2 Interactions between the four participating roles

The interactions between the roles are as follows:

1. Service providers publish their Web services with QoS information to UDDI registries. Web services available in UDDI registries are identified uniquely by an interface key.
2. Clients ask the WSB for services that implement a certain interface and accomplish the required QoS requirements.
3. If the WSB does not already hold up-to-date information on offers that accomplish clients' requirements, the WSB will request Web services according to the interface key from one or more UDDI registries. Note that we would prefer the model in which the WSB prefetches information of offers that clients could be interested in. This would accelerate the lookup phase significantly.
4. The UDDI registries return a list of services that implement the interface key.
5. The WSB asks the service providers for service descriptions, e.g. WSDL files.
6. The service providers return their service descriptions with QoS offers.
7. The WSB tests the offers against the clients' requirements.
8. The WSB returns the most appropriate service to the client.
9. The client directly invokes the service with the original QoS requirements. At this time, the QoS requirements regarding the network performance are actively mapped onto the underlying transport technology.

Note that the WSB in step 7 tests the offers (step 6) against clients' QoS requirements sent in step 2. The definition of both the QoS requirements and offers is essential in our architecture. In the following subsections we will describe the QoS definition and components participating in this process, how offers and requirements are matched, the mapping of the QoS requirements onto the QoS aware network as well as how service providers deliver real-time information about the server performance to the user.

2.1 Web Service Layer QoS Support

For the QoS aware dynamic selection of Web services the QoS parameters defined by both service providers and clients must be compared by the WSB. The WSB selects the cheapest service fulfilling the requirements from all offers available for services that implement the specified interface. To standardize the QoS specification for efficient comparison, we have designed a Web service-QoS XML schema. Its core element is a *QoSInfo* node that defines a specific QoS level by assigning certain values to standard QoS parameters. *QoSInfo* elements are referenced in a *QoSDefinition* node either for the scope of an individual operation or as a default QoS level for the whole service. This *QoSDefinition*, which also relates the QoS level to a price, can either be a *WSQoSRequirementDefinition* element or a *QoSOffer* element. A *WSQoSRequirementDefinition* element specifies a client's minimal QoS requirements which must not be violated by underperformance. A *WSQoSOfferDefinition* element contains one or more *QoSOffer* elements that each declares a QoS level that a service provider is willing to deliver. Besides the standard parameters, further custom parameters can be declared, referring to a public WS-QoS ontology. Therefore a *WSQoSOntology* element holds definitions of QoS parameters and protocol references.

2.1.1 QoS Info

The most important of all elements are those of the type *tQoSInfo* as depicted in Figure 3. It holds information on the level of QoS regarding the server performance, transport QoS support and protocol required for providing security and transaction support. In a *serverQoSMetrics* element, values for the standard parameters processing time, requests per second, reliability, and availability can be declared as well as custom server QoS metrics.

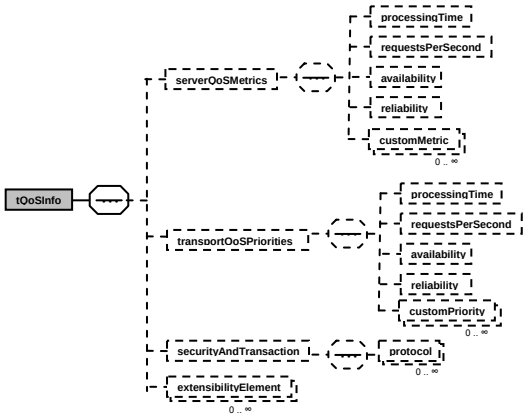

Figure 3 Structure of the type tQoSInfo

A *transportQoSPriorities* element specifies priorities for the four standard transport parameters delay, jitter, throughput, and packet loss rate and optional custom transport QoS priorities. Security and transaction management for Web Services is realized by a variety of protocols. Most of them already have sophisticated mechanisms of negotiating key and session information. Therefore, security and transaction support at this level will be restricted to listing protocols needed for a successful service execution.

2.1.2 WS-QoS Ontology

Custom metrics, custom priority and protocol support statements all have an attribute ontology, which references a file containing a *WS-QoS Ontology* where the referenced types are defined respectively. By using the combination of the ontology's URL and the

parameter name, a reference is unique. A custom transport *QoS priority* is defined by a distinct name and a human readable definition of what metric the priority refers to in a *priorityDefinition* element.

A *custom server QoS* metric defined in a *metricDefinition* element also has a name and a human readable description of what is measured, but it also declares a standard unit and the direction of how values are to be compared.

Accordingly, in a *protocolDefinition* element, a protocol is defined by its name, a human readable description of the reasons for using this protocol and the URL of an overview document of the protocol specification if available.

2.1.3 QoS Definition

Figure 4 shows the type *tQoSDefinition*. An element of this type holds one or more elements of the type *tQoSInfo*. These can be defined for the scope of an individual operation in an *operationQoSInfo* element or for the whole service in a *defaultQoSInfo* element. In its *contractAndMonitoring* node, a node of the type *tQoSInfo* provides references to protocols needed for service management and QoS monitoring as well as entries of third parties that one side would be willing to trust. Finally, the *price* element relates the specified QoS level to the cost of service usage per invocation.

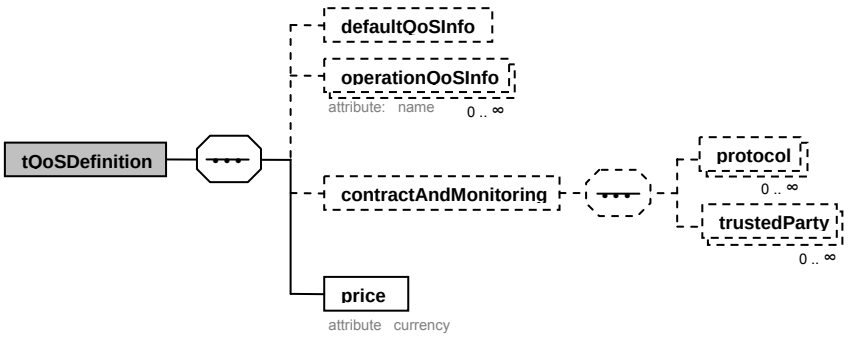

Figure 4 Structure of the type tQoSDefinition

Elements of the type *tQoSDefinition* are either instantiated as a *WSQoSRequirementDefinition* element expressing a client's QoS requirements or as a *qosOffer* representing a minimal QoS level a service provider guarantees to provide for all requests. The *qosOffer* element is extended by an attribute *expires* which denotes a point in time until which the offer will be valid.

2.1.4 WS-QoS Offer Definition

Offers for one service can be declared in a *WSQoSOfferDefinition* element which is introduced into the service's WSDL file as an extension element of the service description's service node. Apart from offers definitions within this node, offers in further WS-QoS files can be referenced in an include element. This allows for dynamically adjusting offers without changing the WSDL file. Furthermore, an offer could be referenced from multiple WSDL files and thus be reused for different services.

2.2 Network Layer QoS Support

In the previous section, we have introduced our approach that allows the definition, lookup, and matching of QoS statements declared by both Web service clients and providers. In order to control and set the requirements of the client application concerning the network performance, we have to deal with the network streams exchanged between the client application and the remote Web service provider. Note that we assume that the underlying transport technology supports QoS such as DiffServ, ATM, or UMTS. On the client side, a QoS proxy resides between the Web service client and the network interface. The proxy observes the traffic on a specific port, through which the Web service client sends its requests to the server. The QoS proxy maps the client's requirements onto the current QoS aware network after detecting QoS parameters set by the client application.

On the server side, a QoS proxy is located between the Web service and the network interface. It sets the QoS parameters according to the client requirements onto the underlying transport technology when the Web service provider sends responses to the service client.

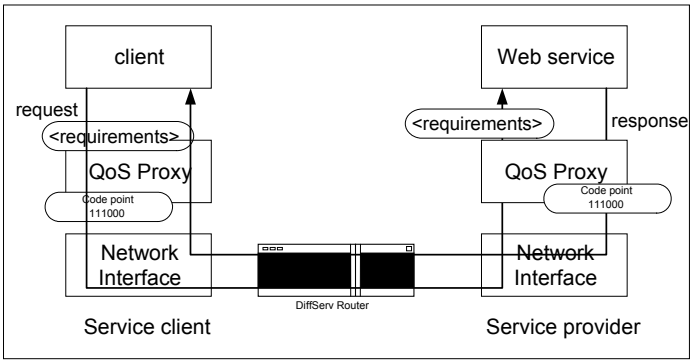

Figure 5 Proxies map client's requirements onto the underlying transport technology

Figure 5 depicts the participating components and the data flows during the interaction between a Web service client and the service provider at runtime. In this case, we assume that the QoS aware network is a DiffServ network. The QoS information regarding the network performance specified by the client is placed in the SOAP headers, which will be parsed by the QoS proxies on both client and server side. Based on the client's information, the proxies mark the DiffServ specific DiffServ code points (DSCP) in the IP packets. DiffServ routers in the network will treat the traffic between clients and server depending on the DSCP. For simplicity, we only show the interaction between the Web service client and provider, ignoring the UDDI registries and the WSB, which are also Web services.

2.3 Server and Network Performance Observation

Our architecture allows users to be informed with real-time information about the current server and network performance. We introduce a QoS channel between the server and the client. The QoS channel is realized by placing information into the SOAP headers. The user defines what QoS information regarding the server and network performance

she wants to know. The server delivers the required information to the client by applying the QoS channel. The client knows the service time, which is defined as the time interval between the moment the client requests the service and the moment the client receives the response. The server provides its performance data such as the processing time of the current request. The client can derive the network performance from this information.

A graphical user interface (GUI) on the client side shows the server and network performance. The usage of the GUI is fully flexible. The user can switch off the GUI completely; she can choose QoS parameters she is interested in from the GUI; she can request statistics about the server performance of to other classes of the same service. She can be alerted instantly in case of server and network underperformance. She can even get a feeling what would happen if she paid for a better or worse class of the same service as she does.

3 Conclusions and Future Work

In this paper, we have introduced our current effort on QoS support in Web services and the dynamic mapping of requirements from Web service layer onto the underlying QoS aware network layer. Our approach allows the dynamic selection of Web services depending on various QoS requirements. The QoS definition regarding network performance can be stated independently of the underlying network. Its mapping onto the current transmission technology takes place at runtime. Our approach allows the user to receive instant information about the server and network performance.

We have built a testbed in order to conduct performance measurements of our architecture. We are interested for example in the performance of the WSB for selecting the most appropriate service in comparison to the standard lookup model. Another interesting issue is to extend our architecture with support for mobile clients.

4 References

[Da02] A. Dan, A. R. Franck, A. Keller, R. King, H. Ludwig. (IBM) Web Service Level Agreement (WSLA) Language Specification 2002. http://dwdemos.alphaworks.ibm.com/wstk/common/wstkdoc/services/utilities/wslaauthoring/WebServiceLevelAgreementLanguage.html

[MA02] D. Menasce and V. Almeida, Capacity Planning for Web Services, Prentice Hall 2002

[Na01] K. Nahrstedt, et al., QoS-aware middleware for ubiquitous and heterogeneous environments IEEE Communications Magazine, Nov. 2001

[Ri01] H. Ritter, Bedarfsorientierte Dienstgüteunterstützung durch adaptive Endsysteme, VDI Reihe 10 Nr. 681, 2001

[Sa02] A. Sahai, V. Machiraju, M. Sayal, L. Jie Jin, F. Casati (HP) Automated SLA Monitoring for Web Services, http://www.hpl.hp.com/techreports/2002/HPL-2002-191.pdf

[To03] V. Tosic, B. Pagurek, K. Patel WSOL – A Language for the Formal Specification of Classes of Service for Web Services Research Report OCIECE Feb. 2003. http://www.sce.carleton.ca/netmanage/papers/TosicEtAlResRep03-03.pdf

[Vo01] T. Voigt, R. Tewari, D. Freimuth and A. Mehra. Kernel Mechanisms for Service Differentiation in Overloaded Web Servers. 2001 Usenix Annual Technical Conference, Boston, MA, USA, June 2001