A Framework for Using Web Services to Enhance QoS for Content Delivery

Francesco Buccafurri, Pasquale De Meo, Gianluca Lax, and Domenico Ursino
University Mediterranea of Reggio

Mariagrazia Fugini, Stefano Modafferi, and Barbara Pernici
Politecnico di Milano

Roberto Furnari and Anna Goy
University of Torino

Pasquale Lops, Domenico Redavid, and Giovanni Semeraro
University of Bari Aldo Moro

An integrated approach to quality of service for content delivery using Web services includes quality definitions, user contracts, and fault monitoring.

Market presence and low price drove the success of early Web-content delivery. Currently, however, quality-of-service (QoS) issues in content delivery are becoming prevalent. For example, consumers want to complete transactions safely, or want products (such as multimedia objects) delivered quickly and correctly. In the late 1990s, the development of so-called content-delivery networks that cooperate transparently to deliver content helped address some of the issues.[1] Even so, the wide variety of underlying network infrastructures, the multiplicity of end-user devices, the dynamic mix of available content, and the user demand for personalized services and content have created significant challenges for existing platforms, which don't yet offer enough flexibility to handle these challenges.

This article presents an approach and a framework designed to enable the QoS analysis of Web-service processes for real-time service provisioning (RTSP) based on service compositions. In the article, we demonstrate that it's possible to combine QoS parameters defined on various domains to provide differentiated services, and to dynamically allocate available resources among customers while delivering high-quality multimedia content. We also demonstrate that it's possible to customize multimedia streams to highly variable network conditions to provide acceptable quality in spite of factors possibly affecting QoS, such as network bandwidth or user frame rate when accessing the service. To achieve these objectives, we leverage our earlier work related to complex, adaptive Web-service processes to supply more information for determining the quality and size of the delivered object.[2]

Additionally, this article introduces an architecture that supports our approach. The architecture includes a module for predicting possible QoS faults through a machine-learning approach and a module for monitoring QoS parameters and supporting possible adaptation and recovery actions in case of failure. Our goal in presenting this work is to stimulate future research about quality of composite services in service-oriented architectures (see http://www.oasis-open.org/committees/tc_cat.php?cat=soa?). The ''Related Work'' sidebar presents other researchers' approaches.

Web services and streaming delivery

Our approach deals with processes interacting with different actors and offering value-added services that are able to satisfy user requests for complex objects, such as an e-learning object, a clinical health service, or an e-government service. The methods of quality analysis and the reference-tool architecture we outline here combine the worlds of Web services and streaming by focusing on jointly provisioning complex services and their quality.

We assume that the environment is composed of several nodes operating at two layers: Web services and their related protocols, and RTSP protocols. Figure 1 shows our reference scenario: a user requires, and eventually receives, a complex service obtained as a composition (possibly a choreography[3]) of different Web services; one of these (WS2 in the figure) provides streaming content. Our main concerns

1070-986X/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society

include addressing problems associated with the guarantee of QoS requirements in variable contexts and providing an active approach to solving or anticipating possible failures. Therefore, the focus of this article is not only on monitoring, but also on anticipating faults with prediction techniques.

QoS definition

Defining a general QoS model is essential. We adopt two QoS models, one for the Web-service layer and one for the RTSP layer. Accordingly, we use two ontologies to represent quality parameters, with the semantics conforming to methodologies and techniques used in the Semantic Web community. We used the OWL Web Ontology Language[4] to develop the QoS ontologies, and followed the conceptual structure proposed by Papaioannou.[5]

Web-service layer

Our Web-service QoS model relies on the following parameters to describe the QoS related to a single synchronous operation provided by the server:

- *Response time.* Time elapsed between the instant a request is sent from the client and the instant the server computes the response.
- *Price.* Amount a client pays to the server for operation provisioning.
- *Availability.* Probability that a given operation is accessible at the moment of the request.
- *Reputation.* Ratio of the number of invocations with the requested QoS and the total number of invocations.
- *Data quality timeliness.* Freshness (up-to-date degree) of data.
- *Data quality accuracy.* Correspondence between given data and reference data considered as correct.
- *Data quality completeness.* Coverage of exchanged data with regard to total data representing the managed information.

For each quality parameter, we define a correspondence among ranges of values $R_v \in \Re$ as disjoint sets, and define the QoS level L_{QoS} on a finite set of values (for example, high, medium, and low). With these considerations, the coverage of the possibility space is complete.

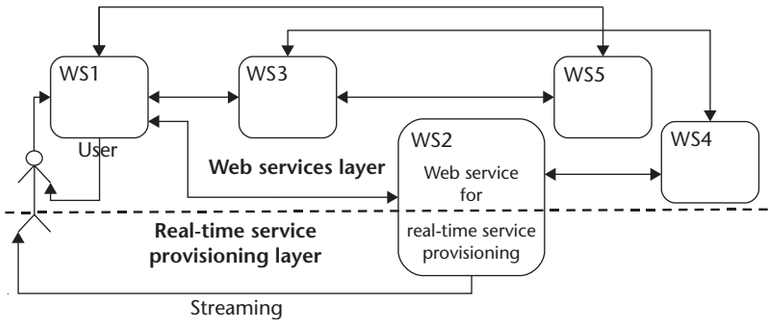

Figure 1. Reference scenario for our models.

To provide an example of associations between values, levels, and QoS contracts, we assume the following: $\forall$ *QParam Level*$_{QParam}$ = {high, medium, low}. Under this assumption, we could have the following response times (where "s" stands for seconds):

$T.Resp$ = low if $T \geq 15$ s
$T.Resp$ = medium if 5 s $\leq T <$ 15 s
$T.Resp$ = high if 5 s $< T$

Real-time service provisioning layer

The QoS of a multimedia stream is based on two classes of parameters, namely:

- *User related.* These express the user's requirements and preferences in accessing multimedia services, and allow the evaluation of relevance to the user of each component (video, audio, and data) of the delivered multimedia flow.
- *Network related.* These parameters support the assessment of the amount of available network resources (bandwidth, channel speed, and so on).

Table 1 (next page) provides the entire set of parameters defining the QoS at the RTSP layer, where "N" stands for *positive integer number* and indicates that no metric exists and "R" stands for *positive real number*.

QoS contract

Our method stipulates a QoS contract between a provider (server) and a consumer (user) regarding a set of parameters. Specifically, given that a user wants service *i* and a

Table 1. RTSP layer QoS parameters.

Parameter	QoS parameter	Metric/range
User related	Access count	—/N
	Video access count	—/N
	Audio access count	—/N
	Data access count	—/N
	Video degradation count	—/N
	Audio degradation count	—/N
	Data degradation count	—/N
	x-resolution	Pixel count/N
	y-resolution	Pixel count/N
	Chrominance	Bit count/N
	Luminance	Bit count/N
	Frame rate	Frames/second/N
	Audio channels	—/N
	Audio codex	—/N
	Audio frequency	Hertz/R
Network related	Video bandwidth	Kbps/R
	Audio bandwidth	Kbps/R
	Data bandwidth	Kbps/R

server provides service S, a user contract UC_{i-S} consists of two parts:

- a mandatory part consisting of the seven levels of QoS (see list in the "Web-service layer" section), one for each QoS parameter, and of a rule $R_{UC_{i-S}}$ used to determine whether a QoS violation occurs; and
- an optional part related to specific aspects, such as the QoS of real-time contents provided by the server.

Table 2. An example QoS contract with rules of a low/medium response time and price set at low.

QoS	Parameters	Level
Global Web service	Response time	Low
	Price	Low
	Availability	Medium
	Reputation	Medium
	Data quality timeliness	Medium
	Data quality accuracy	Medium
	Data quality completeness	Medium
Streaming	*x*-resolution	640
	y-resolution	480
	Chrominance	8
	Luminance	8
	Frame rate	25
	Audio channels	2
	Audio codex	1
	Audio frequency	128 kHz
	Data bandwidth	300 Kbytes per second

Each operation provided by the server to the user has to adhere to the corresponding contract, an example of which is shown in Table 2. We define the concept of global QoS as a tuple composed of the union of different QoS values or levels, due to the heterogeneity of the range domains. It follows that the global QoS value *GlobQoSV* of an operation *i* of a service *S* is the set composed of the union of each QoS value corresponding to a QoS parameter. It also follows that the global QoS level *GlobQoSL* of a given operation *i* of a service *S* is the tuple composed of each QoS level $L_{QoS_{QParam_j}}$ corresponding to a QoS parameter $QParam_j$.

Through its rule portion $R_{UC_{i-S}}$, the user contract UC_{i-S} splits the set of all possible global QoS levels $GlobQoSL_S$ into two subsets, $S_{UC_{i-S}}$ and $NS_{UC_{i-S}}$, the former satisfying UC_{i-S}, and the latter not satisfying UC_{i-S}. If the client or the server do not respect a contract, a contract violation arises and must be managed.

Prediction

To anticipate faults, we use a prediction model and a support framework based on monitoring and machine learning. Because the global QoS varies in the runtime environment, we determine the global QoS by observing a set of parameters (the prediction global QoS is the tuple containing only the Web-service quality parameters). Some regularity can emerge from observation of the global QoS, depending on the values of specific parameters in different situations. For example, the global QoS on the same sequence of operations can change, and this regularity can be useful in determining the range variability. By observing these regularities, we can define or predict the global QoS. In our scenario, a huge set of simulated data contained in the log files generated by the runtime environment is available, making it possible to analyze this data to define behavioral models. We use machine-learning techniques[6] to build a system capable of providing suggestions on possible variations of the global QoS.

Figure 2 shows a sample process useful in describing the learning problem. A process instance uses two services S_1 and S_2, and invokes several operations (O_1, O_2,..., O_n). We can formalize the learning problem as follows: given an answer to a process operation $S_i.O_i$ and given the current global QoS level $Curr_{QoSL_{Si.Oi}}$, we want to know, with a certain

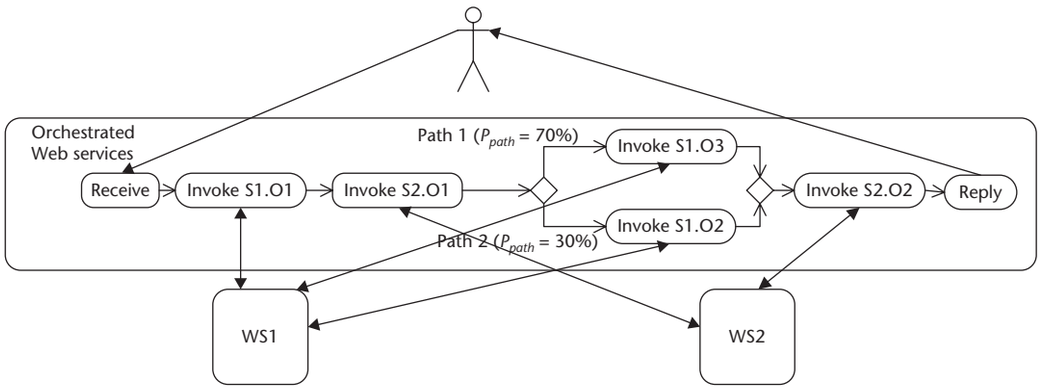

Figure 2. Sample Web-services process for QoS prediction.

probability $P_{globalQoS}$, the global QoS level $globalQoS_j$ corresponding to $Curr_{QoSS_{Si.Oi}}$, with j being an identifier of a future operation of S_i.

Because we know which object we want to analyze (that is, the global QoS level during a future operation), we can build a set of preclassified data (a training set) and apply a supervised machine-learning technique. We used the C4.5 classifier as the core of our machine-learning system.[6] Specifically, in a bootstrap phase where the set of training data is collected. Each occurrence of this set is formed by the global QoS observed for an input instance plus the input instance itself. The following features represent input instances for the classifier:

- Service name *S* identifying the server providing the current operation;
- Operation name *O* identifying the current operation;
- QoS parameters *T.Resp*, *P*, *Avail*, *Rep*, *DQ.Timel*, *DQ.Acc*, and *DQ.Compl* identifying the current values of each of the seven QoS parameters; and
- Target operation O_{Target} identifying the next operation upon which the QoS prediction will be performed.

Once the system completes the bootstrap phase, it can obtain a global QoS prediction on the target operation, giving the current values of the input instance for the current operation to the machine-learning system.

Overall QoS framework

We use the QoS models presented here in a framework that tests the approach through a software architecture whose components separately validate each QoS aspect. The overall architecture (presented in detail elsewhere[7]) is represented by a set of nodes that cooperate through the mechanisms of the Service-Oriented-Architecture that provide complex object delivery based on business processes. The cooperation among nodes at the Web-service layer operates as a choreography, guaranteeing a loosely-coupled composition framework suitable for the integration of heterogeneous services.

The architecture monitors, detects, and predicts QoS faults, which the architectural components detect and manage by providing a self-healing Web-service approach. The approach consists of making the services aware of possible faults and capable of repairing them. A complete description of the repair and self-healing features is outside the scope of this article; interested readers can refer to Sokol and K.P. Eckert for more details.[8]

The business scenario consists of several coordinated services that provide real-time content. Process management occurs on the communication layers for Web-services interaction and for RTSP due to the technological differences between the two layers. However, it's necessary to manage the two layers uniformly from the user perspective, and hence the streaming server exposes a management interface to the Web-service layer. The separation is total for communication protocols, while for QoS, information exchange is enabled between the two layers to react to QoS faults.

Web-service layer

In the Web-service layer, monitoring involves several purposes:

- checking if the execution of the complex service correctly follows the interaction protocol defined by the global choreography;

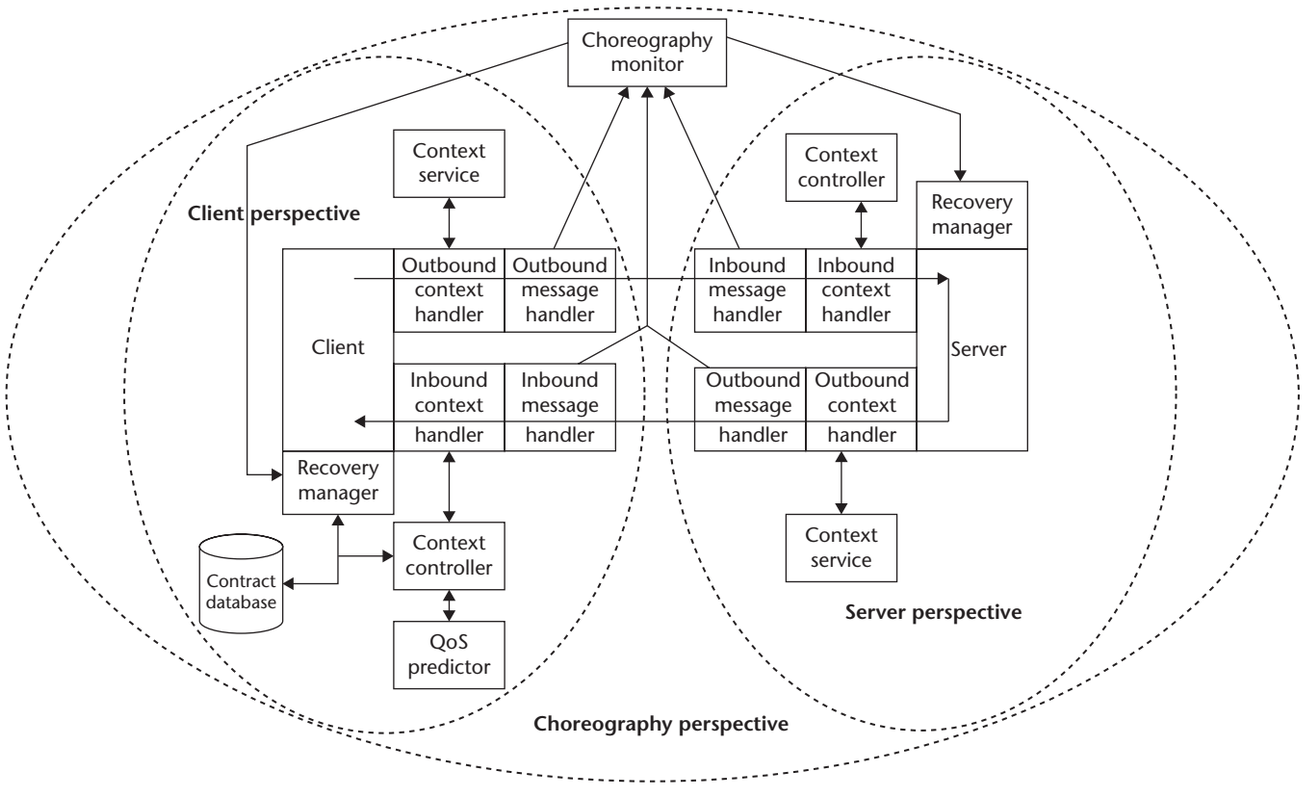

Figure 3. Architecture for QoS management in composite services. (Figure originally appeared in Buccafurri et al.[7])

- checking whether a QoS contract is respected; and
- estimating the QoS of following operations to prevent QoS faults.

The recovery function reacts to faults through repair actions, such as undoing, redoing, substituting, or compensating an operation.[9] Figure 3 shows how the framework works when considering a single client–server interaction.

In a Web-service choreography, none of the involved participants centrally executes the composite service. Each Web service is simply aware of its own status and doesn't have a global view encompassing all the cooperating services. However, the choreography definition[3] represents a global perspective on the composite service that a choreography-monitoring Web service can rely on to detect faults occurring at the choreography level. The choreography monitor, relying on the notification messages received by each Web service and on the global choreography description, can track the progress of the service execution. Thus, the choreography monitor can detect possible mismatches between the order of message exchanges occurring during service execution and the one prescribed by the choreography definition.[10,11]

As shown in Figure 3, two logical components implement the monitoring functionality:

- the choreography monitor uses the global choreography description to check the complex service execution; and
- the inbound and outbound message handlers, implemented by each Web service participating in the choreography, intercept messages and notify the choreography monitor.

If the system detects any mismatch in message exchanges, the monitor indicates a choreography fault to the recovery manager of the involved service.

QoS parameters define the context of each service operation and must be identified when a message is leaving an actor and controlled when a message reaches an actor. The context controller allows the identification of situations such as violations of a QoS contract. At the client side, the controller is in charge of interacting with the QoS predictor. In case of anomalies, the

controller reports to the application manager (for example, to the orchestration engine).

Each node in the architecture is associated with a (local) context service. This is a (local) Web service in charge of managing the local context—that is, of computing the values of the QoS parameters when requested. Context handlers attached to the application interact with the context service (see Figure 3). The controller reads the incoming QoS values and maps them onto the corresponding QoS levels. Then, it checks whether the global QoS level respects the contract stored in a specific database. If the global QoS level doesn't respect the contract, the controller indicates a QoS fault exception to the recovery manager.

In addition, a controller can ask the QoS predictor to perform a QoS prediction. The QoS prediction system operates as a Web service with a Web-Services Description Language (WSDL) interface through which the controller can require a QoS prediction. The input instance features together with the global QoS from the target operation form the training set.

After the training phase, the Web service can manage QoS prediction requests. In particular, it's possible to ask for more than one prediction at the same time, a characteristic that is needed when the process contains conditional points leading to alternate branches. In these cases, the process manager needs to know the prediction for each branch with a certain probability P_{Path} used to discriminate between alternatives. Furthermore, the process manager holds information about a probability for both alternative paths, and sends this information to the prediction services, which normalize it with the precision of the classifier.

The path probability is 1 when the request concerns a single future operation; otherwise, the path probability has to be specified on the process structure at design time. In Figure 2, let us assume now that O_3 is included in P_{Path1} and that $P_{Path1} = 75$ percent and O_4 is included in P_{Path2} and that $P_{Path2} = 25$ percent. It follows straightforwardly that the path probability is 1 when the request concerns a single future operation.

The output of the QoS prediction system will be the QoS level $globalQoS_j$ (associated with a probability $P_{globalQoS}$) that identifies the $Curr_{QoS_{Si.Oi}}$, with O_j being the operation of service S_i on which the prediction operates. Because the workflow contains a choice construct, the output provides the path probabilities computed as $P_{path}|P_{globalQoS}$. Tables 3 and 4 show examples of input and output produced by the classifier. In particular, it shows the QoS level associated with S_1, with the next operation being $globalQoS_4$ with a 60 percent probability and $globalQoS_1$ with a 23 percent probability. The controller decides, according to given thresholds, whether the probability is enough to indicate a QoS prediction fault to the recovery manager.

Table 3. Input example.

Input	Parameter	Value
Current service	—	S_1
Current operation	—	O_2
Current QoS	Response time	5 s
	Price	7 euros
	Availability	0.99
	Reputation	0.75
	Data quality timeliness	0.75
	Data quality accuracy	0.75
	Data quality completeness	0.75
Target operation	—	O_3(75%), O_4(25%)

Table 4. Output example.

Output	Parameter	Value
Current service	—	S_1
Current operation	—	O_2
Next operation 1	O_3	$globalQoS_4$ (60%)
Next operation 2	O_4	$globalQoS_1$ (23%)

Real-time service provisioning layer

Monitoring bandwidth available for RTSP plays a key role in the processes related to multimedia streaming. Specifically, knowledge of the bandwidth available on each network link enables the detection of bottleneck links. Monitoring bandwidth is therefore essential for regulating and improving the QoS associated with a streaming application.

Recently, researchers have proposed many approaches to estimate the bandwidth available for a service containing a multimedia component; some of these techniques rely on the definition of suitable mathematical models capable of describing how multimedia traffic is distributed over a computer network.[12] Other techniques are based on the use of

probing packets, namely packets that are transferred over the network at a fixed rate. The techniques based on the probing-packet methodology let us estimate available bandwidth starting from the packet-transmission rate.

To monitor bandwidth, we use a technique, related to probing packets, that makes use of one-way delays. Consider an information source delivering a multimedia flow directed to a particular user; this flow consists of many packets (probing packets), which are sent at a particular transmission rate (a probing rate). Let B be the bandwidth to estimate. If the probing rate is less than the bandwidth to be estimated, then the delay associated with the receipt of two subsequent packets is equal to 0; in other words, the rate at which the sender emits the packets is equal to the rate at which the receiver gathers them.[2] On the contrary, if the probing rate is greater than B, such a delay would be a positive number.

We use this technique to design an algorithm capable of effectively determining the bandwidth across a communication channel.[12] The algorithm relies on two support variables: R_{min} (initially set equal to 0) and R_{max} (initially associated with a large value greater than the largest bandwidth value offered by communication channel). The algorithm performs a loop. At the n-th iteration, the following steps occur:

1. The sender emits packets at a rate equal to R_n.
2. The receiver analyzes the packet delay. If the delay is positive, then the sender poses $R_{max} = R_n$ otherwise it sets $R_{min} = R_n$.
3. The sender computes the value $R_{n+1} = R_{max} + R_{min}/2$.

The algorithm ends if $|R_{max} - R_{min}| \leq \omega$, where ω is a predefined threshold; if B is constant, then the algorithm converges in logarithmic time against the initial value associated with R_{max}.

To recover RTSP when a contract violation occurs, we start by defining a linear optimization problem. The weights of the objective function reflect each multimedia component's relevance to the user. That is, they indicate whether the video component is more relevant than the audio one for a particular user. Our approach solves the proposed optimization problem and determines the new values (B_v^*, B_a^*, B_d^*) of video, audio, and data bandwidth.

Once we compute the optimal video, audio, and data bandwidth distribution, we must determine the combination of QoS streaming parameters that fit the bandwidth constraints and maximize the QoS perceived by the user. Various combinations of QoS parameters might require the same video bandwidth. However, the various parameters presumably have different importance for the user. Consequently, different QoS values could be obtained with the same video bandwidth. For this reason, our approach considers the average values of the various parameters required by the user in the past and derives some weights to use in a second optimization problem.

This technique is designed to maximize the value of an ad hoc defined index D_S measuring the satisfaction degree of the user. D_S takes into account how the values of the streaming QoS parameters (for example, horizontal resolution or luminance) are close to the corresponding, user-indicated maximum and minimum values. Interested readers can find the analytical details relevant to the definition of the two optimization problems (the first solved with the simplex method, and the second with the gradient projection method) as well as the analytical procedures belonging to the optimal resource distribution and to the maximization of the user satisfaction elsewhere.[13]

Quadrantis implementation

We implemented our framework in the Quadrantis Project (see http://home.dei.polimi.it/pernici/index.htm). We developed the Web-service layer components using WS-Business Process Execution Language (see http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel) as a composition model and the WS-Context specification[14] for the client–server interaction protocol. We developed the handler based on axis handlers (see http://ws.apache.org/axis/). The choreography-monitoring tool in Quadrantis relies on the WS-Coordination protocol (see http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-errata-os.pdf) by replacing the WS-Coordinator with the monitor Web service. We developed the QoS prediction system as a Web service and provided the WSDL interface with two methods (one for the training and one for the steady execution)

through which the controller can require a QoS prediction. With this setup, each component in this layer operates as a Web service.

The RTSP layer consists of two main components. One monitors the bandwidth available on a multimedia channel by executing the algorithm described in the ''Real-time service provisioning layer'' section. The other component solves two optimization problems. The first is a linear-optimization problem that maximizes customer satisfaction (measured on the basis of the user profile) under the constraints given by the available bandwidth. The solution is to distribute the bandwidth over the three multimedia components (audio, video, and data). We implement the solution in Java following the scheme from the simplex method.

The second optimization problem, which is nonlinear, is to maximize customer satisfaction, operating within each component of the bandwidth allocation (audio, video, and data) through a suitable setting of its high-level parameters. For example, for the video component, two high-level parameters are resolution and chrominance. To solve this problem, we designed a heuristic polynomial solution inspired by the gradient projection method. Because the implementation of our strategy needs some basic operations of numerical linear algebra, we used some facilities available in the Java Matrix package (see http://math.nist.gov/javanumerics/jama/).

Conclusion

We plan on extending our proposed QoS models and mechanisms to react to possible faults detected by our environment. We are developing and extending recovery actions and the associated support tools using adaptation and self-healing capabilities. For the RTSP aspects, specifically, we plan to implement ad hoc recovery policies to describe and manage the customer-satisfaction level. We also plan to extend the architecture with fine-grained mechanisms to have the two levels communicate in an interleaved way and to elaborate advanced contracts for QoS. **MM**

Acknowledgments

The Italian MIUR PRIN Project Quadrantis partially funded this work.

Related Work

To encourage repeat use of a Web site and build customer loyalty, organizations needed to shift their business focus to e-services. In general, integrated-service management supports efficient cooperation between various business entities to offer end-to-end, QoS-based services. But doing so is challenging, especially when using heterogeneous technologies. Several solutions propose content-distribution optimization through advanced server replication,[1] while others propose platforms based on open interfaces and standards, which can offer interoperable and configurable solutions.[2,3] The Web-services model is entering the picture as a possible strategy.

Sokol and Eckert propose a service-management architecture involving service, content, and network providers, and content consumers, with the objective being end-to-end, QoS-based services through the integrated management of content, networks, and devices.[4] Yu and Lin discuss QoS dimensions, namely latency, availability, timeliness, and reliability in Web applications that provide real-time information, multimedia content, and time-critical services.[5] These researchers study QoS control issues and a proposed QoS-aware, Web-service architecture where a QoS broker operates between users and providers. Zhang and Chung present a component model to express the requirements for a QoS-centered, device-independent, multimedia, Web-service environment to establish a three-tier, open framework that partially implements the requirements.[6]

References

1. G. Pallis and A. Vakali, ''Insight and Perspectives for Content Delivery Networks,'' *Comm. ACM,* vol. 49, no. 1, pp. 2006, pp. 101-106.
2. Z. Lu et al., ''MCSAMS: A Novel WSRF and Multi-Agent Based Distributed Multimedia Content Service Alliance and Management Scheme,'' *Proc. IEEE Congress on Services,* IEEE Press, 2007, pp. 348-355.
3. J. Sokol and K.P. Eckert, ''MCDN: Multimedia Content Discovery and Delivery,'' *Proc. Int'l Symp. Autonomous Decentralized Systems* (ISADS), IEEE Press, 2007, pp. 411-420.
4. E. Borcoci et al., ''Service Management for End-to-End QoS Multimedia Content Delivery in Heterogeneous Environments,'' *Telecommunications 2005: Advanced Industrial Conf. Telecommunications/Service Assurance with Partial and Intermittent Resources Conf./E-Learning on Telecommunications Workshop* (AICT/Sapir/Elete), IEEE CS Press, 2005, pp. 46-52.
5. T. Yu and K.J. Lin, ''QCWS: An Implementation of QoS-Capable Multimedia Web services,'' *Multimedia Tools and Applications,* vol. 30, no. 2, 2006, pp. 165-187.
6. J. Zhang and J.Y. Chung, ''An Open Framework Supporting Multimedia Web services,'' *Multimedia Tools and Applications,* vol. 30, no. 2, 2006, pp. 149-164.

References

1. M. Pathan, R. Buyya, and A. Vakali, eds., *Content Delivery Networks,* LNEE 9, Springer, 2008.
2. D. Ardagna et al., ''PAWS: A Framework for Processes with Adaptive Web Services,'' *IEEE Software,* vol. 24, no. 6, 2007, pp. 39-46.
3. C. Peltz, ''Web Services Orchestration and Choreography,'' *Innovative Technology for Computing Professionals,* vol. 36, no. 10, 2003, pp. 46-52.
4. D.L. McGuinness and F. van Harmelen, eds., OWL Web Ontology Language Overview, World Wide Web Consortium (W3C) recommendation, Feb. 2004; http://www.w3.org/TR/owl-features/.
5. I.V. Papaioannou et al., ''A QoS Ontology Language for Web-Services,'' *Proc. IEEE Int'l Conf. Advanced Information Networking and Applications,* (AINA), IEEE CS Press, 2006, pp. 101-106.
6. I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques,* 2nd ed., Morgan Kaufmann Publishers, 2005.
7. F. Buccafurri et al., ''Analysis of QoS in Cooperative Services for Real Time Applications,'' *Data and Knowledge Engineering,* vol. 67, no. 3, 2008, pp. 463-484.
8. J. Sokol and K.P. Eckert, ''MCDN: Multimedia Content Discovery and Delivery,'' *Proc. Int'l Symp. Autonomous Decentralized Systems,* (ISADS), IEEE Press, 2007, pp. 411-420.
9. WS-Diamond team, ''WS-Diamond: Web Services—DIAgnosability, MONitoring, and Diagnosis,'' to be published in *At Your Service: An Overview of Results of Projects in the Field of Service Engineering of the IST Program,* Series on Information Systems, MIT Press.
10. L. Ardissono et al., ''Monitoring Choreographed Services,'' *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering,* T. Sobth, ed., 2007, Springer Verlag, pp. 283-288.
11. W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, ''Business Process Management: A Survey,'' *Proc. Int'l Conf. Business Process Management,* (BPM), LNCS 2678, Springer, 2003, pp. 1-12.
12. M. Jain and C. Dovrolis, ''End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput,'' *IEEE/ACM Trans. Networking,* vol. 11, no. 4, 2003, pp. 537-549.
13. G. Araniti et al., ''Adaptively Controlling the QoS of Multimedia Wireless Applications through User Profiling Techniques,'' *J. Selected Areas in Comm.,* vol. 21, no. 10, 2003, pp. 1546-1556.
14. M. Little, E. Newcomer, and G. Pavlik, eds., *Web Services Context Specification (WS-Context),* OASIS committee draft, Nov. 2004; http://xml.coverpages.org/WS-ContextCD-9904.pdf.

Francesco Buccafurri is a full professor of computer science at the University Mediterranea of Reggio Calabria, Italy. His research interests include knowledge representation, model checking, information security, data compression, agents, and peer-to-peer systems. Buccafurri has a PhD in computer science from the University Mediterranea of Reggio Calabria. Contact him at bucca@unirc.it.

Pasquale De Meo is a postdoctoral researcher at the University Mediterranea of Reggio Calabria. His research interests include multiagent systems, user modeling, personalization, XML, cooperative information systems, and folksonomies. De Meo has a PhD in system engineering and computer science from the Department of Electronics, Computer Sciences and Systems, University of Calabria. Contact him at demeo@unirc.it.

Gianluca Lax is an assistant professor of computer science at the University Mediterranea of Reggio Calabria. His research interests include data reduction, peer-to-peer systems, data streams, e-commerce, and digital signatures. Lax has a PhD in computer science from the University Mediterranea of Reggio Calabria. Contact him at lax@unirc.it.

Domenico Ursino is an associate professor at the University Mediterranea of Reggio Calabria. His research interests include multiagent systems, personalized and device-adaptive e-services, knowledge extraction and representation, scheme integration, semistructured data and XML, cooperative information systems, and folksonomies. Ursino has a PhD in system engineering and computer science from the University Mediterranea of Reggio Calabria. Contact him at ursino@unirc.it.

Mariagrazia Fugini is an associate professor of computer engineering at Politecnico di Milano, Italy. Her research interests include information system security and development, software reuse, e-government, and cooperative applications for e-science. Fugini has a PhD in computer engineering from Politecnico di Milano. Contact her at fugini@elet.polimi.it.

Stefano Modafferi is a researcher in the Department of Electronics and Information, Politecnico di Milano. His research interests include the development of adaptive services. Modafferi has a PhD in information engineering from the Politecnico di Milano. Contact him at modafferi@elet.polimi.it.

Barbara Pernici is full professor of computer engineering at Politecnico di Milano. Her research interests include workflow design, cooperative information systems, Web-based information systems and virtual enterprises, adaptive information systems and Web services, data quality, temporal databases, and applications of database technology. Contact her at pernici@elet.polimi.it.

Roberto Furnari is a PhD student in computer science at the Doctoral School of Science and High Technology, University of Torino, where he works in the computer science department. His research interests include Web-service orchestrations and choreographies. Furnari has a laurea degree in computer science from the University of Torino. Contact him at furnari@di.unito.it.

Anna Goy is a researcher in the computer science department at the University of Torino. Her research interests include distributed Web-based applications, in particular adaptive hypermedia, context-aware systems, and Web 2.0 technologies. Goy has a PhD in cognitive science, focusing on lexical semantics, from the University of Torino. Contact her at goy@di.unito.it.

Pasquale Lops is assistant professor in the Department of Informatics, University of Bari Aldo Moro, Italy. His research interests include machine learning, recommender systems, digital libraries, user modeling, and universal access. Lops has a PhD in hybrid recommendation techniques based on user profiles from the University of Bari Aldo Moro. Contact him at lops@uniba.it.

Domenico Redavid is a postdoctoral researcher in the department of informatics, University of Bari Aldo Moro. His research interests include machine learning, Semantic Web, Semantic Web services, and semantic management of business processes. Redavid has PhD in informatics from the University of Bari Aldo Moro. Contact him at redavid@uniba.it.

Giovanni Semeraro is an associate professor in the computer science department at the University of Bari Aldo Moro, where he teaches programming languages, formal languages and compilers, enterprise knowledge management, and intelligent information access, and natural language processing. His research interests include intelligent information access and personalization, Web and content mining, user modeling, machine learning, Semantic Web, and digital libraries. Contact him at semeraro@di.uniba.it.