Semantic Modeling, Translation and Matching of QoS

Alexandra Moraru[1], Blaž Fortuna[2#], Carolina Fortuna[2]**
[1]*Faculty of Computer Science, Technical University of Cluj-Napoca,
George Baritiu 26-28, 400027 Cluj-Napoca, Romania
[2#]Department of Knowledge Technologies, Jožef Stefan Institute,
[2]*Department of Communication Systems, Jožef Stefan Institute,
Jamova 39, 1000 Ljubljana, Slovenia
e-mail: ale_moraru@yahoo.com

ABSTRACT

The variety of access and transport technologies available in modern computer networks pose significant challenges related to compatibility and quality of service (QoS) related issues. Applications and services can have many different and unique requirements towards the transportation services (TSs) they use to interconnect. Traditionally, applications are required to specify their QoS requirements in the language which the TSs understand. This results in reformulation of intuitive parameters (i.e. desired video resolution) to parameters understood by the TSs (i.e. required bandwidth).

This paper presents techniques for (a) automatic matchmaking of application requirements to the offers by TSs providers and (b) automatic translation of application requirements into the TSs QoS requirements. To this end semantic technologies, namely OpenCyc, are used for ontological modeling, translation and matchmaking. We present relevant examples on how semantic technologies can be used in the context of communication networks.

1 INTRODUCTION

The word "quality" is defined by [1] as the "totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs". "Service" is defined by the same standard as a "type of product […] always the result of an activity or interaction between a service supplier and a customer and can take many forms". The QoS framework assumes that a customer requests a service having a given QoS profile from a provider.

Throughout this paper, the customers will be applications using services over a network to which the access is possible over multiple different transport services (TSs). The differences between TSs can come from different technologies (i.e. WiFi vs. UMTS), different pricing (i.e. pay-as-you-go vs. flat rate), different availability, etc.

Selecting appropriate transport services for a specific application in a environment with many available TSs can pose significant technological challenges, especially with the emergence of software defined radios [2][3]. Providing multi-services in a multi-network environment has been previously investigated [4] for wired networks using multiple layers of abstraction to hide transport specific complexity. However, complexity management and the necessity for interoperability require more advanced approaches.

In this paper we present an approach for automatic matching of applications to the appropriate TSs based on application requirements and the QoS parameters offered by the TSs. In order to do the matching, we developed an approach for automatic translation of application level requirements into the QoS parameters understood by the TSs.

Both approaches are based on the Semantic web technologies [5] which were already successfully applied for the task of web service (WS) [6] composition, matching and monitoring as well as for modeling and mapping of WS QoS specifications [7]. However, research on semantic translation of application QoS requirements to the network QoS parameters is still open as existing attempts have been limited in scope [7][8][9].

Application QoS requirements differ from one application to another (e.g. a streaming service compared to a browsing service) and need to be properly recognized and translated to TS requirements, which are platform and technology dependent. In our approach, a reasoning engine uses a QoS model combined with a set of rules to map and match requirements. For instance, it must be able to infer that if an application requires streaming TS for a QCIF picture size with frame rate of 15 pictures per second, the network must meet the following requirements: 64 kbps bit rate, 300 ms latency, 20 ms jitter and 10^{-4} packet error rate.

This paper is structured as follows. Section 2 presents related work. Section 3 we discuss QoS representation and matchmaking using OpenCyc and Section 4 presents experiments relevant to using OpenCyc for translation and inference. Finally, we conclude the paper.

2 RELATED WORK

One recent trend in service oriented architecture (SOA) [10] related work is to develop QoS aware web services. In [6] and [11], the authors introduce the Web services-QoS architecture (WS-QoS) meant to close the gap between the WS layer and the underlying QoS-aware transport technologies. WS-QoS extends the Universal Description,

Discovery and Interoperability (UDDI) by introducing a broker. QoS requirements and offers are defined using XML schema, approach that makes the architecture highly interoperable. However, service discovery and matchmaking lack semantics with this approach.

In [7], the authors discuss the shortcomings of non-semantic specifications of QoS for WS and propose a semantic QoS (SQS) framework. They built a QoS hierarchy ontology model encoded in RDFS and test the overhead of the ontology design. As opposed to [11], this work does not go down to the transport network QoS, thus not considering this aspect of QoS for WSs. They actually use a middleware approach that passes application QoS specifications to the underlying technology.

Developments of semantic representations for QoS for WSs in the form of ontologies can be found in [8][9] and [12]. In the first two, the authors report on the development of QoSOnt using Web Ontology Language [13]. The latter reports on an ontological encoding for QoS developed in DAML-S and then ported to OWL. They also provide results regarding matchmaking and measurement using the ontology. A survey on other representations for QoS services can be found in [14].

In [15] the authors develop OWL-QoS ontology for the purpose of finding matches between offers from the TS providers, called adverts, and the consumer requests (called request). Example of an advert can be seen in Figure 1.

Figure 1 ProviderProfile Class

The OWL-QoS ontology uses a three layer representation of QoS: QoS Profile Layer, QoS Property Definition Layer and QoS Metrics Layer. Profile layer stands for matchmaking purpose, the property definition layer specifies the domain and range constraints of the properties and metrics layer contains metrics definition and measurement. The authors use reasoning engine Racer [16] for performing matching adverts to the requirements.

The work in this paper uses available QoS representations and investigates their usability for application to TSs QoS translation. We base our ontology on the one presented in [7] and we use OpenCyc [17] for ontology modeling, semantic matchmaking and QoS parameter translation.

3 QOS REPRESENTATION AND MATCHMAKING USING OPENCYC

In this section we first describe how the QoS domain and its parameters are modeled using OpenCyc and then show an example of how we can use the model to do the matchmaking between the TSs providers and consumers.

After trying and working with several different available semantic models of QoS [7][8][15] we decided for the approach presented in [7]. The main advantage was the hierarchy of applications which can be used to better determine the QoS characteristics. The dimensions of QoS are represented in the base-class layer of the ontology, on top of which the QoS domain ontology layer is built. We recreated a part of the base-class ontology in OpenCyc.

The Cyc ontology consists in a few numbers of collections, predicates and implication rules. For example, relevant characteristics for a video application can be represented, among others, by the frame rate of the video, the video codec used and the resolution or format of the screen. We modeled this data by creating the following collections: VideoCodec (as a subclass of Codec collection), VideoResolution and VideoFormat. A video format is characterized by video resolution and for this we created a predicate which links the VideoFormat with the correspondent resolution. Also video codecs are typically standardized so these relations can be stated. For instance, some resolutions are related to some frame rates based o the codec used; for this we created another predicate, but this time for linking three objects: one video codec, one video resolution and one numerical value for the frame rate. In the next section we will explain how we used this structure for the inferring the data rate.

In the previous section we described the OWL-QoS ontology and how it was used for matchmaking between the advert and the request: a match is a pair (*advert*, *request*) where the objectives requested by the *request* are satisfied by the *advert*. The matchmaking algorithm presented in [12] introduces degrees of matching, and Racer reasoning engine is used to infer the matches between requests and adverts and their degrees.

We implement the matchmaking similar to the one described in [12], without introducing the degrees of matching however. OpenCyc was used for both modeling and for reasoning. We created a QoSProfile collection which holds both adverts and requests. It has two predicates which define the cost for a service and the response time which a generic system takes to react to a given input. QoSRequest and QoSAdvert are subcollections of QoSProfile and they inherit the defined predicates for QoSProfile. An advert is a match for a request if the cost for the service provided by the advert is lower than the price the requester can pay and the response time advertised is also lower than the one requested.

The matchmaking conditions can be stated in CycL using rules as in Figure 2. The matching predicate takes two arguments: the first one is a request profile and the second one is an advert profile. The first rule from Figure 2 matches the response time between an advert and a request. X stands for a QoSRequest and Y for a QoSAdvert. It can

be noticed that there is one condition that verifies that X is a request (isa ?X QoSRequest). This is there only for making the rule more human readable, otherwise the condition is redundant as X is the first argument of the "matchTime" predicate.

For every new QoSProfile added, request or advert, the rule will automatically calculate every possible match, since these rules are designed to be forward rules. This enables fast retrieval of matching adverts. For example if there are two adverts in the knowledge base (KB), the first offering a response time of 3000 milliseconds at the cost of $1 per second and the second one a response time of 1000 milliseconds for $3 per second, when a new request stating that it needs a response time of 2000 millisecond for $5 per second is added, then the second advert will be automatically found as a match, without having to specifically ask for the match to be done. Another advantage of using OpenCyc is that there are some concepts for units of measure already defined and integrated in the KB.

```
(implies
    (and
      (isa ?X QoSRequest)
      (responseTime ?X (MillisecondsDuration ?T1))
      (isa ?Y QoSAdvert)
      (responseTime ?Y (MillisecondsDuration ?T2))
      (or(equals ?T1 ?T2) (lessThan ?T2 ?T1)))
    (matchTime ?X ?Y))

(implies
    (and
      (costPerSecond ?X (USDollarFn ?C1))
      (costPerSecond ?Y (USDollarFn ?C2))
      (or(equals ?C1 ?C2) (lessThan ?C2 ?C1)))
    (matchCost ?X ?Y))

(implies
    (and
      (matchCost ?X ?Y)
      (matchTime ?X ?Y))
    (match ?X ?Y))
```

Figure 2 Matching rules in CycL

4 USING OPENCYC FOR INFERRING NETWORK QOS

This section describes an approach for automatic translation of QoS requirements from the application point of view into the QoS parameters that the TS providers understand. Combining the translation with the matchmaking presented in the previous section results in a system where application requests, expressed in a language intuitive for their domain (i.e. video streaming) can be automatically matched to the appropriated TS according to their QoS specifications.

Based on the approach from [7] which introduces a QList as a support for specifying the requirements of one application we created a similar structure in OpenCyc, for the translation of application requirements to network requirements. For a video application, the list of requirements can specify, for instance, the video resolution and the codec; others may specify only the video format. Along with other specific requirements for a video application, like color depth or frame rate, we want to translate all of them into network requirements. One of the network requirements is the data rate needed by the application and this data rate can be inferred from the application requirements even if these are incomplete. The formula based on which the data rate is calculated is expressed in and the rules for OpenCyc to make the necessary inferring are shown in Equation 1.

$$DR = FrameHeight \cdot FrameWidth \cdot FrameRate \cdot Color\ Depth$$

Equation 1 Computing data rate

Table 1 specifies two lists with different requirements and the inferred data rates. The first list specifies the video codec, the maximum frame size (as video resolution), frame rate and color depth. From the first three requirements, using the rule on the left side of Figure 3 and the information from the KB, the compatible frame size will be inferred. In the KB, relations are specified between codec, frame rate and the related resolution for that combination; in this case for a frame rate of 20 fps and MPEG4 codec, the video resolution inferred is 320x240. The second list specifies only three parameters: video format, color depth and frame rate. Knowing the video format, OpenCyc will infer, based on the KB, that the frame size is 320x240. Then, by applying the rule on the right side of Figure 3 the data rate is computed. So, the system is resilient to different formats of specifications. The requirements can be different as long as there is enough knowledge in the KB.

	QoSList1	**QoSList2**
Video Codec	MPEG4	
Video Resolution	320 x 320	
Video Format		QVGA
Color Depth (bits)	8	8
Frame Rate (fps)	20	20
Inferred Data Rate (bit/sec)	**12288000**	**12288000**

Table 1 QoS requirements and inferred bits per second

Another issue which can appear when there are multiple users and each one "speaks" its own language is the way they can understand each other. For instance, in the above example, we used *frame size* and *video resolution* referring to the same thing and it is easy for humans to understand that. However, for a machine this equivalence must be explicitly specified. In OpenCyc, a fast way to say that *frame size* is the same with *video resolution* may look like this: "(isa FrameSize VideoResolution)". Having this rule, it will not matter which term one uses in the list of requirements. But as anything easy and fast to do, it is not really correct, because in this way FrameSize represents a subclass of VideoResolution, not an equivalent class. This happens due to the fact that OpenCyc applies Unique Name Assumption (UNA) for all the concepts that were set in the KB there must be different names only for different entities. A way to solve this problem is to associate one or more strings to a concept so that it will be possible for that concept to be found also by different names. In our

```
(implies
    (and
        (hasCodec ?Q ?C)
        (hasFrameRate ?Q ?FR)
        (hasMaxResolution ?Q ?MR)
        (hasColorDepth ?Q ?CD)
        (hasResAndFR ?C ?R ?FR)
        (frameWidth ?R (Pixel-UnitOfCount ?W))
        (frameHeight ?R (Pixel-UnitOfCount ?H))
        (frameWidth ?MR (Pixel-UnitOfCount ?MW))
        (frameHeight ?MR (Pixel-UnitOfCount ?MH))
        (or
            (lessThan ?W ?MW)
            (equals ?W ?MW))
        (or
            (lessThan ?H ?MH)
            (equals ?H ?MH))
        (evaluate ?BS (TimesFn ?W ?H ?FR ?CD)))
    (computeDR ?Q ?BS))
```

```
(implies
    (and
        (hasVideoFormat ?Q ?VF)
        (hasFrameRate ?Q ?FR)
        (hasColorDepth ?Q ?CD)
        (formatHasResolution ?VF ?R)
        (frameWidth ?R (Pixel-UnitOfCount ?W))
        (frameHeight ?R (Pixel-UnitOfCount ?H))
        (evaluate ?BS (TimesFn ?W ?H ?FR ?CD)))
    (computeDR ?Q ?BS))
```

Figure 3 Rules for inferring data rate

example, the concept is defined as VideoResolution, and to this concept a string is attached adding the following assertion: (nameString VideoResolution "FrameSize") to the English Micro Theory.

5 CONCLUSIONS

In this paper we studied the existing ontologies for QoS and performed several experiments to study the way these can be used to perform matchmaking between providers and consumers of transfer services and how to do automatic translation from application level requirements to the QoS parameters which TS can understand. To this end, we modeled in OpenCyc several QoS profiles to simulate matching between adverts and requests. We also constructed several rules in OpenCyc which were used to inferre network QoS parameters from application parameters.

Because of the large number of applications and network technologies existing nowadays, semantic technologies seem suitable for QoS modeling. However, larger taxonomies and more complex experiments are required to assess the full potential of this approach. In the future we plan to extend the ontology for the translation of QoS application characteristics to network characteristics and do a tighter integration between the matchmaking and the parameter translation. Furthermore we want to integrate this into OpenCyc because of the good inference it provides.

6 ACKNOWLEDGEMENTS

This work was supported by the Slovenian Research Agency and the IST Programme of the EC under NeOn (IST-4-027595-IP) and PASCAL2 (IST-NoE-216886).

References

[1] Quality management systems – Fundamentals and vocabulary, ISO 9000:2000

[2] B. Xie, A. Kumar, D.P. Agrawal, Enabling multi-service on 3G and beyond: challenges and future directions, IEEE Wireless Communications Magazine, Jun. 2008, vol. 13, p. 66-72

[3] C. Fortuna, M. Mohorcic, B. Filipič, Multiobjective Optimization of Service Delivery Over a Heterogeneous Wireless Access System, ISWCS 2007, Reykjavik, Iceland

[4] F. Steegmans, N. las Mercouroff, B. Ceccaldi, Mu3S - A Middleware Platform for Telecommunications Information Networking, In Proc. Telecommunications Information Networking Architecture Conference, 1999, p. 131-133

[5] D.J. Weitzner, J. Hendler, T. Berners-Lee, D. Connolly, Creating a policy aware web: discretionary, rule-based access for the World-Wide Web, In Web and information security, E. Ferrari and B. Thuraisingham (Eds), IRM Press, 2005

[6] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schiller, A Concept for QoS Integration in Web Services, In Proc. Web Information Systems Engineering Workshop, 2003, p. 149-155, http://page.mi.fu-berlin.de/tian/pdf/tian_wqw2003.pdf

[7] L. Zhou, H. K. Pung, L. H. Ngoh, Towards Semantic Modeling for QoS Specification, In Proc. Conference of Local Computer Networks, Nov. 2006, p. 361-368

[8] G. Dobson, R. Lock, I. Sommerville, QoSOnt: a QoS ontology for service-centric systems, 31st EUROMICRO, Porto, Portugal, Aug.-Sept. 2005

[9] G. Dobson, A. Sanchez-Macian, Towards unified QoS/SLA ontologies, IEEE Service Computing Workshop, Sept. 18-22, 2006

[10] P. Bianco, R. Kotermanski, P. Merson, Evaluating a Service-Oriented Architecture, Technical Report, September 2007, CMU/SEI-2007-TR-015, http://www.sei.cmu.edu/pub/documents/07.reports/07tr015.pdf

[11] M. Tian, QoS integration in Web services with the WS-QoS framework, Doctoral Dissertation, Freien Universität Berlin, Nov. 29th, 2005

[12] C. Zhou, L.-T. Chia, B.-S. Lee, DAML-QoS Ontology for Web Services, In Proc. IEEE International Conference od Web Services, 2004, p. 472, http://www3.ntu.edu.sg/home5/pg04878518/Articles/icws04_235_Chen_Z.pdf

[13] OWL Web Ontology Language overview, W3C, http://www.w3.org/TR/owl-features/

[14] V. X. Tran H. Tsuji, Semantics in QoS for Web Services: A Survey, http://sigswo.org/papers/SIG-SWO-A801/SIG-SWO-A801-02.pdf

[15] C. Zhou, OWL-QoS ontology, http://www3.ntu.edu.sg/home5/PG04878518/OWLQoSOntology.html

[16] RacerPro, http://www.racer-systems.com/

[17] OpenCyc, http://www.opencyc.org/