Supplying legacy applications with QoS: a description syntax at application, end-user and network level [*]

Anne Thomas
Software engineering group, Department of computer sciences,
Technische Universität Dresden
01062 Dresden, Germany
Anne.Thomas@inf.tu-dresden.de

ABSTRACT
In this article we consider end-to-end interdomain Quality of Service support for non QoS-aware applications. We present a description of QoS requests resp. offers at two levels: 1- at network level: the so called *network service profile*, and 2- at application resp. end-user level: the so called *application profile*. These profiles support end-to-end and interdomain interworking for legacy applications wether they are QoS-aware or not. The network is QoS enabled and offers guaranteed network services.
With the proposed mechanisms (combination of converters and description syntax at network and application level) it is possible on the one hand to offer user-friendly QoS, and on the other hand to provide a negotiation and description syntax for QoS.
Thereby we address the topic of reverse-engineering Internet applications in order to extract the behavior and parameterizations information necessary for the creation of the application profiles.

KEY WORDS
QoS, legacy application, end-user, application profile, reverse engineering

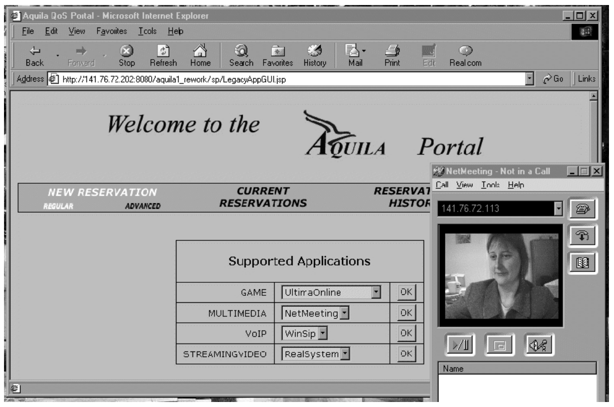
1 Introduction

The variety of applications and services using the Internet is constantly increasing. The network which was originally planned to carry mainly data traffic is nowadays being used for various traffic profiles, ranging from real time audio and video to web traffic. However, the best effort nature of the current Internet is not sufficient to cope with the requirements of these different kinds of traffic, in terms of throughput, delay, jitter and packet loss (network parameters affecting the proper execution of most Internet applications).
Motivated by the rapid changes of the Quality of Service *QoS* requirements of these new network applications the Internet research has been evolving towards providing a range of architectures like the Integrated Services (IntServ) architecture [1], the first one introducing QoS in the Internet, or the more scalable and manageable Differentiated Services (DiffServ) architecture [2].
At present time no public QoS-enabled network exists, and QoS aware applications are rare since few QoS APIs (like the RSVP API RAPI and Winsock2 from Microsoft) are available for application developers. This fact implies that most of applications susceptible of being used in a future public QoS enabled network will be non-QoS-aware legacy applications. In this article we address the topic of QoS provisioning for those legacy applications and focus on the QoS request-offer scenario between the end-user and a QoS enabled network. To enable this we propose a method based on a description syntax specifying QoS at end-user, network and application level. We discuss the question wether a generic or implementation dependent specification is appropriate. Thereby we address the problem of reverse engineering of Internet applications in order to extract behavior information. At the end of the paper we present ongoing related works.

2 Scenario

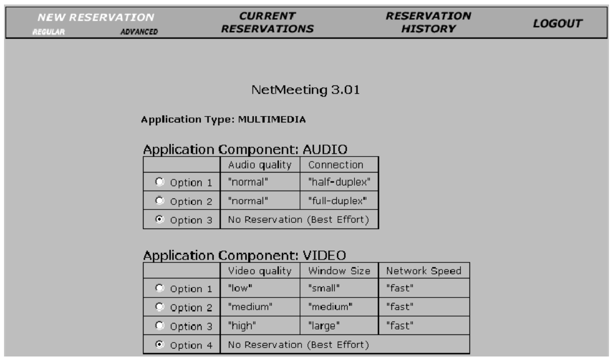
In the following, we consider a QoS enabled network where the different domains offer guaranteed network services. In the presented case study the network relies on the AQUILA [4] architecture which has been realized in a prototypical implementation. The end-user uses legacy applications (like RealPlayer from RealSystems, NetMeeting from Microsoft, a SIP agent etc.) as usual. Theses applications can neither request nor offer QoS. The concern of this article is to present a method to supply such applications with QoS without touching their code. One solution consists in offering QoS at end-user level in parallel Fig. 1 to application execution via a so called "QoS portal".
The QoS portal (Fig. 1 and Fig. 2) enables the identification of the application, the presentation of the QoS related to the application in use and the selection of

[*]This work was performed in the framework of IST Project AQUILA (Adaptive Re-source Control of QoS Using an IP-based Layered Architecture - IST-1999-10077) funded in part by the EU. The author wish to express her gratitude to the other members of the AQUILA Consortium for valuable discussions.


Aquila QoS Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Real.com
Address http://141.76.72.202:8080/aquila1_rework/sp/LegacyAppGUI.jsp Go Links
Welcome to the AQUILA Portal
NEW RESERVATION REGULAR ADVANCED
CURRENT RESERVATIONS
RESERVAT HISTOR
NetMeeting - Not in a Call
Call View Tools Help
141.76.72.113
Name

Supported Applications		
GAME	UltimaOnline	OK
MULTIMEDIA	NetMeeting	OK
VoIP	WinSip	OK
STREAMINGVIDEO	RealSystem	OK

Figure 1. Application selection in the portal. The QoS portal as well as the application are running



NetMeeting 3.01

Application Type: MULTIMEDIA

Application Component: AUDIO

	Audio quality	Connection
○ Option 1	"normal"	"half-duplex"
○ Option 2	"normal"	"full-duplex"
● Option 3	No Reservation (Best Effort)	

Application Component: VIDEO

	Video quality	Window Size	Network Speed
○ Option 1	"low"	"small"	"fast"
○ Option 2	"medium"	"medium"	"fast"
○ Option 3	"high"	"large"	"fast"
● Option 4	No Reservation (Best Effort)		

Figure 2. Quality selection for the application.

a quality level. The aim of the portal is to provide information about the application for the underlying network and vice-versa. The end-user has the possibility to request specific QoS (see Fig. 2) in a simple way using metaphors for QoS. Beside using the QoS portal, a QoS request can be triggered from a complex Internet service (value added web platform binding basic Internet applications together) or directly via a QoS aware application. The QoS request, after a mapping into a network request, is transmitted to an admission control agent accepting or rejecting it. This activity is enabled using the so called *application profile* Fig. 3.
The application profile is a syntax that allows the description of applications in order to present the QoS offer to the end-user via metaphors and to request for QoS at a network independently from a network implementation. The counterpart at network level is a part of the application profile description syntax: the *network service profile*. It enables the description of the network services offered by a domain and serves the inter-domain negotiation. The application profile (and indirectly the network service profile) description

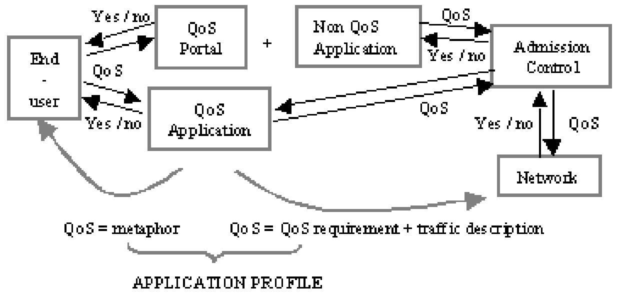


Figure 3. Utilization of application profile - scenario.

syntax is based on the hierarchical application structure depicted in Fig. 4. There are multiple views and it is possible to describe an application in many ways. The description of the application and QoS at the different levels corresponds to the description of the different following artefacts. At network level: description of the QoS expectations and requirements of the application, description of the produced traffic, and description of the implementation dependent QoS request. At application level (control plane level): description of the protocol used, port used. At application level (data plane level): description of the implementation issues of the different service components, and description of the different configuration options (e.g. audio, video). At end-user level: building the metaphors, presenting the possible QoS levels (SessionCharacteristic in Fig. 4). This description syntax is described in detail in the following chapter.

3 Description syntax

In this chapter we present a description of QoS requests (resp. offers) at network level: the so called *network service profile*, and at application resp. end-user level: the so called *application profile* by using the eXtensible Markup Language (*XML*) [7] enhanced by the Data Type Definition (*DTD*) [6] language.
The description syntax is based on the application structure depicted in Fig. 4. A DTD-file rules the description of the different following artifacts at network, application (control plane and data plane levels), and end-user levels.
Toward the network the *application profile* syntax (based on the *network service profile*) offers the possibility to either implement QoS aware applications or complement non QoS aware (but QoS sensitive) legacy applications in order to make them QoS aware. In this special case an intermediate entity serving as a mediator (the so called *converter*) between the network and the application is required.

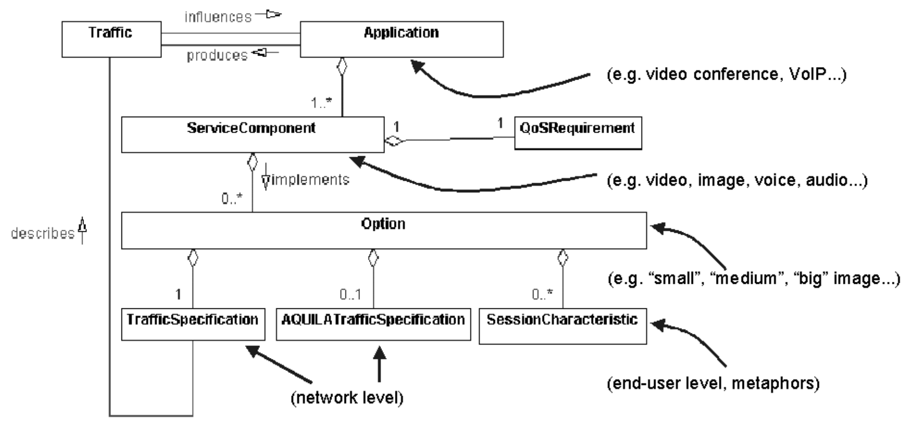


Figure 4. UML diagram analyzing an application. *Application*: legacy QoS non aware Internet application produces *traffic* at the data plane level; is composed of *Service Components* that have general *QoS requirements*; they offer many quality *options*; Each option has a behavior.

3.1 At Network level

The description syntax at network level enables on the one hand the description of the application QoS request toward the network and on the other hand the QoS offer of the network toward the application or the network.

3.1.1 Generic description

At network level, with the description language *network service profile* it is possible to specify and abstract the underlying technical network services of a network independently from the concrete implementation (e.g. DiffServ, etc.). This general description is in a first step the basis for the QoS agreement between two domains at border router level. At the access point of the network, this specification is part of the QoS requirement/request of the application resp. end-user and the Internet Service Provider QoS offer. A weighting system and a subjective (with wordings) rating enable a better definition of the requirements. The DTD syntax of the traffic specification Fig. 4 is as follows:

```
<!ELEMENT TrafficSpecification  (type+,
duration, adaptivity, burstiness,
packetSize, bitRate, flow)>
 <!ELEMENT type EMPTY>
 <!ATTLIST type
  type (realTime | nonRealTime | stream
  | elastic) "stream">...
```

The parameters "type" , "duration" , "adaptivity", "burstiness", "packetSize", "bitRate", and the "flow" describe the traffic the application produces.
The DTD syntax (extract) of the QoS requirements Fig. 4 of a service component is as follow: where the parameters "maxDelay", "maxJitter", "maxLoss", the "bwGuarantee", and "ordering" correspond to the QoS requirements of the application.

```
<!ELEMENT QoSRequirement (maxDelay,
maxJitter, maxLoss, bw, ordering)>
 <!ELEMENT maxDelay (#PCDATA)>
 <!ATTLIST maxDelay
  unit CDATA #FIXED "ms"
  requirement (Low |... High) "medium"
  weight (0 | 1 | 2...| 9 | 10) "5">..
```

Below a detail of a concrete application profile of NetMeeting for the video service component:

```
<QoSRequirement>
 <maxDelay requirement="high" weight="1"
          unit="ms">1200</maxDelay>
 <maxJitter weight="3" unit="ms"
          requirement="low">120</maxJitter>
 <maxLoss weight="5" unit="percent"
          requirement="medium">10</maxLoss>
 <bw weight="8" unit="percent"
          requirement="high">-1</bw>
 <ordering weight="8 requirement="true"/>
</QoSRequirement>
```

3.1.2 Implementation dependent description

The following DTD syntax relies on the specification of the in described [3] QoS request. It is implementation dependent.

```
<!ELEMENT AQUILASpecification (serviceID,
BSP, BSS, minPU, maxPS, PR, SR)>
```

The ”serviceID” parameter corresponds to the name of the AQUILA network service, ”BSP”, ”BSS”, ”minPU”, ”maxPS”, ”PR”, and ”SR” correspond to the parameters of the QoS request format used in [4]. Detail for the video-conference tool NetMeeting corresponding to a ”video very low quality” scenario:

```
<AQUILASpecification>
 <serviceID value="PVBR" />
 <BSP unit="bytes">2000</BSP>
 <BSS unit="bytes">5120</BSS>
 <minPU unit="bytes">60</minPU>
 <maxPS unit="bytes">1500</maxPS>
 <PR unit="bit/s">160000</PR>
 <SR unit="bit/s">75000</SR>
</AQUILASpecification>
```

3.1.3 Discussion

At interdomain level the generic *network service profile* enables the negotiation between two domains and the selection of the next domain providing most appropriate network service. For example a ”Domain a” selects the most appropriate domain fulfilling the commitments to the consumer e.g. ”Domain f”. The basis for selection is the general description of the request and the general description of the offers This implies that each domain possesses a mapping entity enabling the translation of the offer resp. the request specified with the *network service profile* into the proprietary technical representation of the offer resp. request. If we consider a network constituted of n different QoS domains, having a generic description implies having $2n$ mapping possibilities. Not having a generic description implies n^2 mapping possibilities.
A mapping with a generic description makes in this manner only sense when the amount of domains offering different QoS is greater than two. As a matter of fact, it still has to be proven that the generic description syntax works with other QoS implementations than [4]. Moreover the IETF [5] activities of the working groups show that network operators etc. are dealing with much more concrete and basic problems and are not so advanced to discuss cross-implementation issues.
As a conclusion it can be pointed out that on a small scale it does not make a big difference having a generic description or not, since the amount of possible mappings is relatively restricted. But when generic descriptions are not used and new QoS technologies emerge, the number of mappings will grow quadratically.

3.2 At Application level

At application level our idea consists in concretely describing applications both in the user space and the network space as well as at control and data-plane levels.

3.2.1 Data plane level

The information described at data-plane level corresponds to the identification of the different flows produced by the application and the corresponding transport protocol. The following DTD syntax enables the listing of the different service components with their corresponding transtport protocol.

```
<!ELEMENT Implementation
(ServiceComponent,TransportProtocol)>
 <!ELEMENT ServiceComponent (name, option*)>
 <!ATTLIST ServiceComponent
    file CDATA #REQUIRED >
  <!ELEMENT option (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
 <!ELEMENT TransportProtocol EMPTY>
 <!ATTLIST TransportProtocol
        name (TCP | UDP) "TCP" >
```

Example for the NetMeeting tool:

```
<Implementation>
 <ServiceComponent file="Video.xml">
  <name>Video</name>
 </ServiceComponent>
 <TransportProtocol name="TCP"/>
</Implementation>
<Implementation>
 <ServiceComponent file="Speech.xml">
  <name>Speech</name>...
```

3.2.2 Control plane level

The information described at control-plane level corresponds to the characterization of the control protocols, the port numbers, etc. as showed in the following DTD extract.

```
<!ELEMENT protocol
(lowerPortNo?,upperPortNo?,isControlPort?)>
<!ATTLIST protocol
 name (RTP | ... | H324) "H323">
 <!ELEMENT lowerPortNo (#PCDATA)>
 <!ATTLIST lowerPortNo
  value (fixed | configurable) "fixed">
 <!ELEMENT upperPortNo (#PCDATA)>
```

Example for the NetMeeting video-conferencing tool.

```
<protocol name="H323">
 <isControlPort value="true">1720
 </isControlPort>
</protocol>
<protocol name="RTP">
 <isControlPort value="false"></is...
```

3.3 At End-user level

Toward the end-user the description is based on the assumption that an end-user cannot express QoS in terms of high complex technical parameters like e.g. RSVP, UMTS, etc. ones. The end-user can express QoS via metaphors corresponding to the human senses: sight, hearing, and on the perception of time-related behaviour. User-friendly descriptions for QoS correspond to a universal apprehension of applications and ideally make reference to well-known similar services from everyday life like: TV, video recorder, hi-fi, or phone etc. The *application profile* provides a syntax enabling the user friendly description of the QoS sensitive artifacts of applications the so called *session characteristic*. With this syntax it is possible to describe the application artifacts for the end-user.
The *application profile* defines a syntax that enables applications to present the QoS offer to the end-user via metaphors and to request for QoS independently from a network implementation.

```
<!ELEMENT SessionCharacteristic
 (name, semanticalGroup*)>
 <!ELEMENT name (#PCDATA)>
 <!ELEMENT semanticalGroup
 (description, qualifier*)>
  <!ATTLIST semanticalGroup
    type
    (Technical|UserFriendly) "UserFriendly"
    language
    (en |...| gr) "en">
    <!ELEMENT qualifier (#PCDATA)>
```

The name of the service component is e.g. picture size, a semantical group for the service component could be a user friendly description, in english qualified with qualifiers: adjective, terms describing the service component e.g. medium, big ... A detail for the NetMeeting video service component:

```
<SessionCharacteristic>
 <name>Picture size</name>
 <semanticalGroup  type="UserFriendly"
   language="en">
  <description>picture size</description>
  <qualifier>very small</qualifier>
 </semanticalGroup>
 <semanticalGroup type="UserFriendly"
   language="fr>
  <description>taille de limage</description>
  <qualifier>tres petite</qualifier>...
</SessionCharacteristic>
```

3.4 Realization

The presented approach has been implemented within the AQUILA project [4] and successfully deployed. In order to create the application profiles for concrete applications like NetMeeeting from Microsoft, RealSystems from RealNetworks, etc. it is necessary to apply reverse engineering in order to discover behavior artifacts. Each application is analyzed. In order to extract the values the reverse engineer studies carefully the documentation (if existing) and with a test bed measures under varying configurations (e.g. type of the network access: modem, LAN etc.) the different values of the parameters. After the values are experimentally collected it is possible to create the corresponding profile.
Considering the overall scenario it is obvious that the QoS request resp. offer has to be translated at the different levels: between the end-user representation and the network representation, and between the generic representation and the concrete network implementation. These mappings are the tasks of converters situated at the different levels. In the special case of [4] the mapping results are integral part of the concrete application profiles. Due to the fact that [4] provides a single QoS implementation the mapping is one to one.

4 Related work

In the following we give a brief review of other QoS description or specification languages. These languages enable the specification of QoS for new applications but do not focus on the support of legacy applications and apart from [9] do not address end-user QoS. The CQML language [8] appears to be the most developed language, nevertheless it does not support a standardized description language like XML and its implementation tools (parser, etc.).
The ODP-based QuO [10] (Quality of Service for CORBA Objects) framework provides quality of service (QoS) at the CORBA layer and extends the CORBA functional Interface Description Language (IDL) with the QoS Description Language (QDL). QDL is an aspect oriented programming language for describing the QoS aspects such as QoS contracts, the adaptive behaviour of objects and delegates, and the configuration of QuO applications.
The MAQS [11] (Management for Adaptive QoS-enabled Services) project includes QIDL, an aspect oriented programming language extension of the IDL that supports the specification of QoS interfaces.
The Quality Assurance Language (QuAL) [14] in QoSME [13] enables the specification of how to allocate, monitor, analyse, and adapt to delivered QoS. Applications can express in QuAL their QoS needs and how to handle potential violations.
The QML (QoS Modeling Language) [12] is a general-purpose language for defining QoS properties. QML has three main abstraction mechanisms for QoS specification: contract type, contract and profile.
The Component Quality Modelling Language

(CQML) [8] is a language for specifying QoS. The QoS a component provides can be specified independently of how the support is to be implemented and without affecting the specification of its functional properties.
HQML [9] is an XML-based Hierarchical QoS Markup Language, to enhance distributed multimedia applications on the World Wide Web (WWW) with Quality of Service (QoS) capability. HQML allows distributed multimedia applications to specify all kinds of application-specific QoS policies and requirements.

5 Conclusion

With the concept of network services profiles we demonstrated that at network level an implementation independent interdomain interworking scenario is conceivable even recommendable, if more than two different QoS domains interconnect. The working scheme is as follows: at interdomain level the "Domain a" maps its request to a generic request that can be compared to the different generic offers provided by the other "Domains i". This mapping is as well possible at application level between an application and the QoS offers.
With the concept of application profiles we showed that at application level it is possible to supply at least legacy non QoS aware applications with QoS. The solution is based on the assumption that we have a QoS enable network offering network services and non QoS aware Internet applications running on a host. The working scheme is as follows: applications run stand alone at the host in parallel to a so called QoS web portal and protocol gateways. The task of the web portal is to enable the identification of the running application (via manual selection by the end-user), to present the QoS offer in appropriateness with the running application (using the application profiles), and to request for QoS on behalf of the application toward the network. The task of the protocol gateways is to identify control plane information of the running application in order to know which flows are to be supported with QoS. The task of the application profiles is to describe information about application's QoS profile (what requirement does an application have), and so constitute a repository of concrete application profiles.
As a conclusion we want to point out the high efficiency of this solution. With the proposed mechanisms (combination of converters and (generic) description syntax at network and application level) it is possible in an interdomain interworking scenario on the one hand to offer user-friendly QoS, and on the other hand to provide a uniform negotiation and generic description syntax of QoS to support legacy applications.

6 References

[1] R. Braden, D. Clark and S. Shenker, Integrated Services in the Internet Architecture: an Overview, RFC 1633.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, An Architecture for Differentiated Services, RFC 2475.

[3] Winter et al., Final system specification, AQUILA deliverable D1203, 2002. See [5].

[4] AQUILA Adaptive Resource Control for QoS Using an IP-based Layered Architecture, http://www-st.inf.tu-dresden.de/aquila/, 2000-2002.

[5] IETF Internet Engineering Task Force, http://www.ietf.org.

[6] DTD Document Type Definition http://www.w3.org/XML/1998/06/xmlspec-report-v20.htm

[7] XML Extensible Markup Language http://www.w3.org/XML/

[8] Aagedal, Quality of Service Support in Development of Distributed Systems, PhD. thesis, Department of Informatics, University of Oslo, March 2001. http://www.ifi.uio.no/ janoa/papers/thesis.pdf

[9] Gu, Nahrstedt, Yuan, Wichadakul and Xu, An XML-based Quality of Service Enabling Language for the Web, Department of Computer Science University of Illinois at Urbana-Champaign, Urbana, UIUCDCS-R-2001-2212, April 2001.

[10] QuO homepage http://quo.bbn.com/

[11] Becker, Geihs, Generic QoS-Support for CORBA, in Proceedings of 5th IEEE Symposium on Computers and Communications (ISCC'2000) Antibes/France 2000.

[12] Frølund, Koistinen, QML: A Language for Quality of Service Specification, Software Technology Laboratory, Hewlett-Packard Company, Report: HPL-98-10, 1998. http://www.hpl.hp.com/techreports/98/HPL-98-159.pdf

[13]Soares Florissi, QoSME: QoS Management Environment, PhD thesis, Columbia University, 1996. www.cs.columbia.edu/dcc/publications/thesis/pgsf/

[14] QuAL homepage
http://www.cs.columbia.edu/ pgsf/qual.html.

[15] Salsano et al., Specification of traffic handling for the second trial, AQUILA deliverable D1302, 2001.