THE QOS PROVISIONING SERVICE (QPS)

G. Fábián, A.T. van Halteren

KPN Research, PO Box 96, 7500 AB Enschede, The Netherlands.
E-mail: {a.t.vanhalteren,g.fabian}@kpn.com

Situation

Middleware is a software infrastructure for distributed object systems that shields the heterogeneity and distributed nature of underlying software and hardware systems. Traditionally, middleware was used as a communication medium for client/server applications. However, the marketplace for middleware has changed significantly. Today it includes a growing number of QoS sensitive applications, such as distributed multimedia applications and distributed telecom management applications that typically have performance and availability demands. In order to support these QoS sensitive applications, system-specific QoS mechanisms such as OS scheduling and network bandwidth reservation need to be controlled. The middleware layer is a natural place for negotiation between QoS requirements of applications and the QoS capabilities of the underlying system, and to control system resources. However, this has to be done in a transparent manner in order to preserve the portability and interoperability of applications.

Requirements

The requirements on future QoS-aware middleware products can be summarized as *a)* to provide application writers a high level programming paradigm to be able to specify QoS requirements, *b)* to provide applications with transparent QoS negotiation and QoS control mechanisms, *c)* to be able to operate in an open dynamic environment, and *d)* to be generic and flexible enough to accommodate many (future) system-specific QoS mechanisms and to support a wide variety of QoS dimensions.

Solution

In this abstract we present the QoS Provisioning Service (QPS) framework that has been designed to meet the above requirements. QPS is being developed at KPN Research, The Netherlands; it is an instantiation of the QoS-control Architecture for Object Middleware as specified in the AMIDST project [1].

QPS provides QoS provisioning for a single binding between a client application and a server object. QPS realizes QoS provisioning in several steps. At instantiation time, client applications specify their QoS requirements whereas server objects specify the QoS level at which they offer their services. QPS uses XML documents, and a language that is based on the QoS Markup Language (QML) [2] for high level QoS specifications. At run-time, QPS initiates a three-party negotiation between the client, the server and the Distributed Resource Platform (DRP) to establish a QoS agreement. A successful negotiation results in a QoS contract that is associated with the binding. Following these steps, QPS reserves the resources for the binding, in order to ensure the agreed QoS level. The resources can be communication, storage and/or processing resources. The QPS framework allows that QoS modules managing these resources are plugged dynamically into the framework. In the operational phase, the task of QPS is to maintain the agreed QoS for the lifetime of a binding, and provide adequate feedback to applications if the agreed QoS cannot be sustained. This allows applications to adapt and request a QoS re-negotiation. For the QoS maintenance, QPS supports a feedback control loop for resource management as specified in the AMIDST QoS–control architecture.

The CORBA implementation of the QPS framework is shown below in Figure 1. On the client side, QPS is registered as a CORBA service, and it provides a QoS Repository interface for managing QoS requirements of clients. On the server side, the POA is extended with a dedicated QoS Negotiator object to perform the negotiation, and it is also extended with the QoS Repository interface for managing the offered QoS of servants. The current QPS

implementation includes one QoS module that manages network resources. This is QIOP that uses RSVP for network resource reservation, and it implements the Open Communication Interface (OCI) to register and interact with the ORB. Our experiments have shown the performance benefits of communication via QIOP compared to the communication via IIOP.

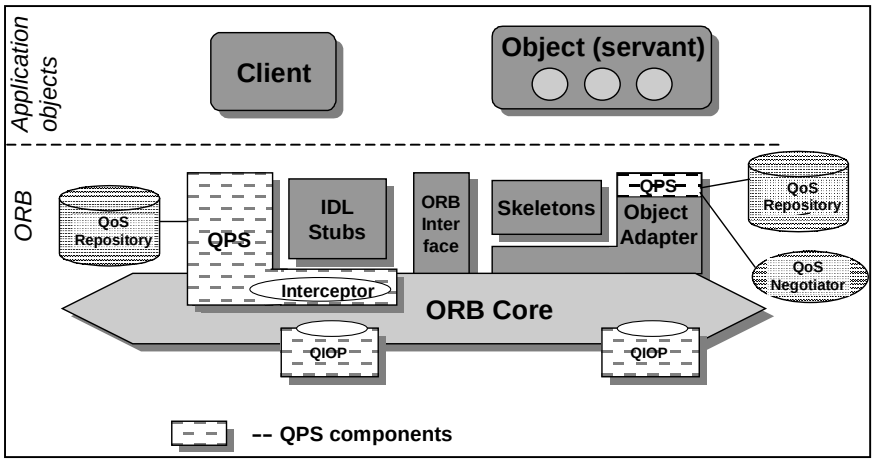

Figure 1: The QPS architecture

The QoS Provisioning Service (QPS) is built on standard extension hooks such as the Portable Object Adapter, Portable Interceptor, and Open Communication Interface, most of which are standardized in CORBA from version 2.4 on. Since QPS is designed as a CORBA ORB service and it uses standardised ORB extension hooks, it can be run on any ORB implementation that provides these extension hooks. QPS is available at http://quamj.sourceforge.net/.

Future directions

Current work on QPS includes the application of the QPS QoS provisioning cycle to multimedia streams. To achieve this, we are extending the ORB with an object infrastructure that can establish a binding between a producer and a consumer of multimedia data. The QoS properties of these bindings objects are then controlled by QPS.

Another direction we're taking QPS to is to use the Meta-Object Facility (MOF) to specify a meta-model for describing QoS capabilities. This allows the object middleware to inspect and adapt QoS capabilities of QoS modules and gives QoS modules a standard way to express their QoS capabilities. Applications can obtain and configure the QoS properties of components, by interacting with a QoS meta-object. Furthermore, we're developing new QoS modules in the area of load-balancing and security.

REFERENCES

[1] L. Bergmans, A. van Halteren, L. Ferreira Pires, M. van Sinderen and M. Aksit, *A QoS-Control Architecture for Object Middleware*, Proceedings of the 7th IDMS Workshop, October 17-20, 2000, Enschede, The Netherlands.

[2] Svend Frolund and Jari Koistinen, *Quality of Service Specification in Distributed Object Systems Design*, Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS), Santa Fe, New Mexico, April 27-30, 1998.