QUALITY OF SERVICE PROVISIONING IN MIDDLEWARE

Cosmina Ivan
Department of Computer Science
Faculty of Automation and Computer Science
Technical University of Cluj, Romania
26, Baritiu str.
Tel: +40-264- 402478
e-mail : cosmina.ivan@cs.utcluj.ro

Abstract: . Commercial off-the-shelf (COTS) distribution middleware is gaining acceptance in the distributed real-time and multimedia systems. Existing COTS specifications, however, do not effectively separate quality of service (QoS) policy configurations and adaptations from application functionality. Application developers therefore often intersperse code that provisions resources for QoS guarantees and program adaptation mechanisms throughout applications, making it hard to configure, validate, modify, and evolve. Middleware is gaining wide acceptance as a generic software infrastructure for distributed applications, a growing number of applications are designed and implemented as a set of collaborating objects using object middleware, such as CORBA, EJB and (D)COM(+). However, quality aspects of interactions between objects cannot be specified nor enforced by current object middleware, resulting only in a best-effort QoS support in middleware. In order to support QoS sensitive applications, system-specific QoS mechanisms such as OS scheduling and network reservation mechanisms need to be controlled, but allowing applications to directly access and control these mechanisms would negatively impact the distribution transparencies offered by the middleware layer. To avoid this, next generation object middleware should offer abstractions for management and control of the system level QoS mechanisms, while maintaining the advantages of the distribution transparencies. The paper discusses the design and implementation of a QoS-enabled middleware service for content delivery with QoS guarantees. The integration of the QoS support result in a QoS-enabled middleware which is representation, location and QoS transparent.

Key words: QoS control, a service for QoS- enabled middleware, adaptive middleware

1. INTRODUCTION

Middleware provides distributed objects with a software infrastructure that offers a set of well-known distribution transparencies. These transparencies enable the rapid introduction of applications for heterogeneous, distributed systems. However, to support guaranteed Quality of Service (QoS) system-specific QoS mechanisms need to be controlled. Allowing applications to directly access and control these mechanisms would negatively impact the distribution transparencies offered by the middleware layer and would reduce the portability and interoperability of distributed object applications.

To avoid this, next generation object middleware should offer abstractions for management and control of the system level QoS mechanisms, while maintaining the

advantages of the distribution tranparencies.The challenge for next-generation middleware is to support application-level QoS requirements, coherent mapping on low level mechanisms, while maintaining the advantages of the distribution transparencies.

This paper is organised as follows. Section 2 describes the QoS concept in open distributed systems. Section 3 gives an overview of the requirements for a middleware-based software infrastructure that offers QoS support to distributed objects and a survey of existing frameworks Section 4 presents our solution in the form of a service for QoS provisioning, with a short evaluation of this framework based on the implementation and section 5 completes with conclusions and further developments.

2. A CONCEPTUAL FRAMEWORK FOR QoS ENABLED MIDDLEWARE

Provisioning of QoS usually involves a common understanding between the two or more parties about the quality characteristics of the service, these parties can be end-users or software components. The generic concepts are based on the ISO/IEC QoS Framework, and the provisioning model defines as main entities in the system the service provider and the service user. Often the user requirements are expressed as subjective requirements whereas the service provider needs objective requirements in order to handle them, therefore the user requirements must be translated into specific QoS parameters expressed in terms of QoS characteristics of the Service provider.

The QoS provisioning model should enable entities to express their quality requirements. The relevant concepts for a QoS provisioning model [2] are defined as follows:QoS characteristic, QoS requirement ,QoS management function and QoS mechanism to manage and control QoS. QoS management architectures provide a coherent set of abstractions and components in order to enable applications for QoS handling.A QoS framework combines interfaces and mechanisms that support the development, implementation and operation of QoS enabled applications and also infrastructure services such as for the negotiation of QoS agreements and monitoring them. Thinking of QoS provision as a client-server relationship means the interaction between them must be augmented with QoS specific behaviour. The integration of QoS provision in middleware must address the following points: *QoS specification, QoS mechanism integration and binding andQoS adaptation.*

Functions that realise QoS support in a distributed processing environment and their positioning in an open distributed system are designed mainly based on the following principles: the *separation principle* which states that transfer , control and management of data are distinct activities and the *integration principle* states that QoS must be configuable, predictable and maintainable over all architectural layers to meet end to end QoS, both principles derived from multimedia and broadband networks.

There are various requirements on QoS design concepts but the most important ones are : extensibility, composability, and a verifable and suitable run time representation. Following those requierements , we propose a layered architecture to structure the problem space and position the functions that provide QoS support in an open distributed system. In this architecture, three functional *layers* are distinguished, each with distinct responsibilities, offering services to adjacent layers on top and using services of adjacent layers below. Orthogonal to the layers, three *planes* are identified: data transfer, control and management.

The architectural framework presented in this paper has been developed to be flexible and reusable and the main benefit that it allows us to combine and balance solutions for the control of multiple QoS characteristics.

3. QoS ENABLED MIDDLEWARE

The original motivation for introducing middleware platforms has been to facilitate the development of distributed applications, by providing a collection of general-purpose facilities to the application designers. Currently, commercially available middleware platforms(COTS) , such as those based on CORBA, are still limited to the support of best-effort Quality of Service (QoS) to applications. This constitutes an obstacle to the use of midlleware systems in QoS critical applications, limitation for the available middleware technology has inspired much of the research that is currently being done on QoS-aware middleware platforms. Ideally, a middleware platform should be capable of supporting a multitude of different types of applications with different QoS requirements, making use of different types of communication and computing resources, and adapting to changes, e.g., in the application environment and in the available resources[1].

Middleware provides distributed objects with a software infrastructure that offers a set of well-known distribution transparencies. These transparencies enable the rapid introduction of applications for heterogeneous, distributed systems. However, to support guaranteed Quality of Service (QoS) system-specific QoS mechanisms need to be controlled. Accessing the low-level mechanisms directly by applications crosscuts the transparency offered by the middleware and limits portability and interoperability.

The middleware layer is a natural place for brokering between QoS requirements of applications and the QoS capabilities of operating systems and networks. The following requirements have been identified and are used as constraints on the design of our QoS provisioning service:

- applications should be able to specify their QoS requirements using high-level QoS concepts.
- the software infrastructure should be modular and easily extensible with new interfaces to system level QoS mechanisms
- the software infrastructure should allow adaptive QoS support.

QoS-enabled middleware is being developed in several projects, with different focuses. We describe here only those systems that enable applications to specify their QoS requirements using high-level language concepts and realise resource adaptation. Three main QoS-aware middleware groups can be identified: general purpose middleware, real-time middleware and multimedia middleware.QuO is a CORBA based framework for configuring distributed applications with QoS requirements[8]. It comes with a suite of description languages that allow applications to specify the interdependencies between QoS properties and system objects, thereby configuring the adaptive behaviour of the underlying system, but allows QoS mechanisms to be added at design time. MULTE-ORB is another QoS-aware middleware that supports configurable multiple bindings [7]. A QoS requirement is specified per binding, together with policies for negotiating QoS and for performing connection management, but the QoS configuration and management system is however ChorusOS and SunOS specific. OMG's Real-Time CORBA (RT-CORBA) specification is targeted at real-time distributed systems. Applications specify policies that guide selection and configuration of protocols and RT-

CORBA supports explicit binding in order to validate the QoS properties of bindings. But after binding time, however, protocols may not be reconfigured.

4. DESIGN AND IMPLEMENTATION

4.1. Design elements

The QoS service is a control plane service, because its actions are limited to a single association between a client and a server object, acting as a broker between the application level QoS requirements and the available QoS mechanisms of the distributed resource platform. The service is a controller for QoS agreements and the framework can be implemented as a CORBA service designed to make use o standard CORBA extension hooks, it is based on specific CORBA mechanisms the Portable Object Adapter (POA), the Portable Interceptor and the Open Communications Interface [6],[7].

A generic control system consists of a controlled system in combination with a controller. The interactions between the controlled system and the controller consist of monitoring and manipulation performed by the controller on the controlled system.

In QoS-enabled middleware, the 'controlled system' is the middleware functionality responsible for the support of interactions between application objects, while the 'controller' provides QoS control.The environment represents the operational context of the middleware, which consists of application objects with QoS requirements and QoS offers. The middleware platform encapsulates the computing and communication resources at each individual processing node, which may be manipulated in order to maintain the agreed QoS. Since the service uses standardized ORB extension hooks (interceptors), it can work with any standard ORB implementation that implements these extension hooks. On the server side, the POA is extended with a dedicated ServantLocator and a Negotiator object for managing servants with an offered QoS, and on the client side , the service provides a QoSRepository (QR) interface for managing QoS requirements of clients.

The lifecycle of bindings controlled by the service revolves around the QoS level offered ($Q_{offered}$) by a server object, the QoS level required ($Q_{required}$) by a client object and the agreed QoS level (Q_{agreed}). The purpose of the service is to control the resources in such a way that some Q_{agreed} is negotiated and maintained for the lifetime of the binding. This agreed QoS is the result of a matchmaking process between the offered QoS of the server object and the required QoS of the client.

The lifecycle phases are inform, negotiate, establish, operate and release . During the establish phase, the service assigns the resources that have been reserved during the previous phase, so other bindings cannot claim these resources. Once sufficient resources have been assigned to the binding, Q_{agreed} must be maintained, this means correcting drifting quality levels, for example, by re-allocating system resources or, in case it is not possible, by informing applications to take appropriate actions. Applications can subsequently decide to lower their $Q_{required}$ and request a re-negotiation, or end their binding, this is the operate phase and finally, when the client does not further need the binding (this is indicated explicitly by the client) system resources are released.

4.2.The service implementation

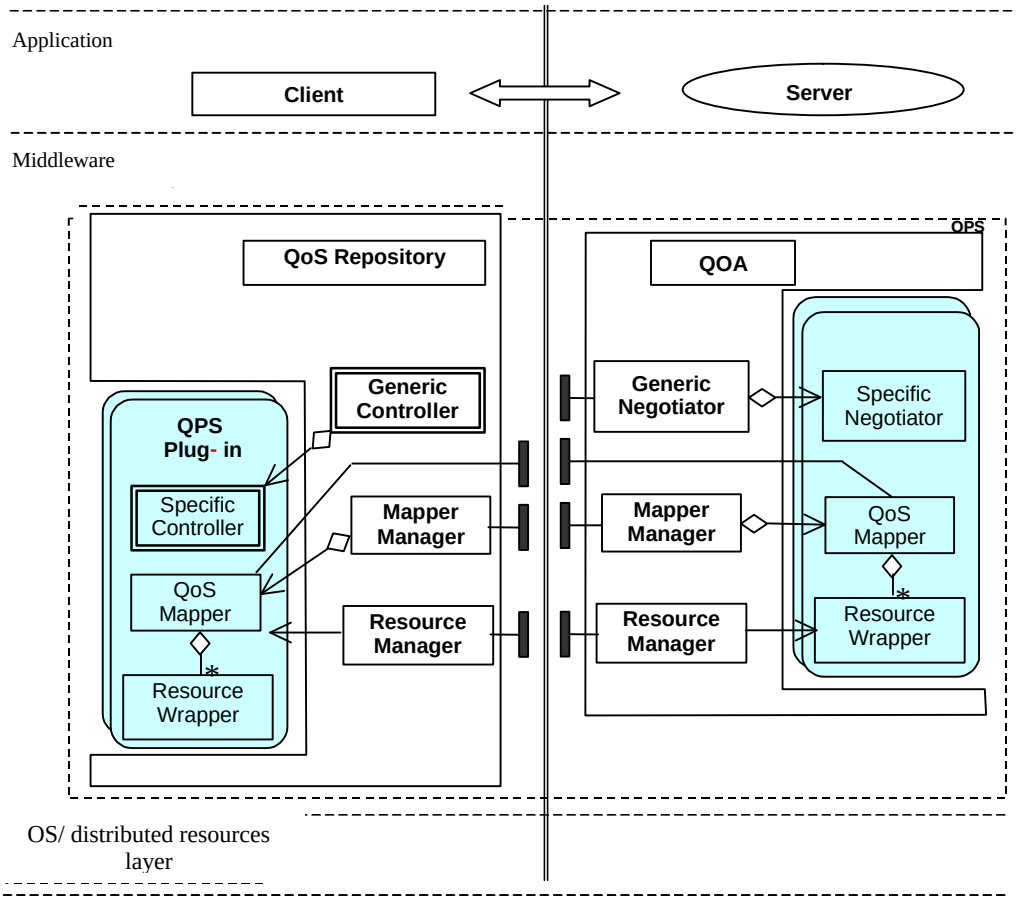

Fig. 1. The QoS framework implementation

We identify three main concerns addressed by the service: QoS negotiation, QoS mapping, and concrete resource manipulation. These concerns are addressed by generic components and plug-in components. The generic components provide generic functionality to manage the plug-in components.

Three generic components are exposed to the application layer: the *QoSRepository* – offers an interface available to clients for registering required QoS with CORBA object references, QOA (QoS-aware Object Adaptor) – offers an interface available to servers for registering offered QoS on a CORBA object and its servant and *GenericNegotiator* which encapsulates the general protocol for performing client-initiated explicit negotiation. The negotiation is performed by the client using an object reference that has been registered with the QoSRepository before.

5.2. Conclusions

Next generation middleware must meet the challenge of evolutionary changes and run-time changes for heterogeneous distributed computing environment in order to provide distributed objects with support for QoS. This paper presents an architecture for QoS-enabled middleware that separates the QoS support functions from 'traditional' data transfer functions. The framework represent a QoS Adaptive Service that enables control plane functions to be added to off-the-shelf object middleware for controlling the QoS of individual client-server associations.

The service follows a lifecycle model to establish and control a QoS agreement between a client and a server.The implementation presented here uses standard CORBA extension hooks, which makes the service a portable CORBA service. Future work includes the study of the applicability of the provisioning service to manage the QoS of multimedia streams and implementing a more advanced interface between the service and system level QoS control mechanisms, including other QoS networking mechanisms.

6.REFERENCES

[1] S. Frolund and J. Koistinen (1999) Quality of Service Specification in Distributed Object Systems Design, Proceedings of the 4 th USENIX Conference on Object-Oriented Technologies and Systems (COOTS), Santa Fe, New Mexico, April 27-30, 1998. Applications (DOA'99)

[2] M. Karsten, J. Schmitt and R. Steinmetz (2001) Implementation and Evaluation of the KOM RSVP Engine, IEEE InfoCom 2001

[3] T. Kristensen and T. Plagemann (2000) Enabling Flexible QoS Support in the Object Request Broker COOL, 20th International Conference on Distributed Computing Systems (ICDCS'00), Taipei, Taiwan

[4]C Ivan. K. Pusztai, An adaptable QoS provisioning service for distributed objects, CSCS14 – 2-5 July 2003, Bucharest , ISBN973-8449-17-0, pag 412-421, Editura Politehnica press

[5] C Ivan, Concepts, models and services for QoS-aware middleware, EMES 2003, Oradea , ISSN12223-2106, section C - Computer Science and Reliability - pg. 71 -81

[6] Object Management Group (2000) The Common Object Request Broker: Architecture and Specification OMG document formal/00-10-33.

[7] Siqueira and V. Cahill (2000) A QoS Architecture for Open Systems, 20 th International Conference on Distributed Computing Systems (ICDCS'00), Taipei, Taiwan.

[8] J. Zinky, R. Schantz, J. Loyall, K. Anderson and J. Megquier (2001) The Quality Objects (QuO) Middleware Framework. Workshop on Reflective Middleware (RM 2001), New York, USA.