

A reflective QoS provisioning service for object middleware

Aart van Halteren
KPN Research
P.O. Box 15000, 9700 CD Groningen
The Netherlands
A.T.vanHalteren@research.kpn.com

Situation

State-of-the-art object communication middleware, such as CORBA, DCOM and Java RMI, is in essence a software infrastructure that enables application components to interact with each other. This software infrastructure provides application components with a rich set of transparencies. These transparencies include location transparency (components can be either local or remote), access transparency (components can be accessed through various transport protocols and deployed on various hardware architectures), failure transparency (components can be replicated or put under transaction management) and management transparency (components and the resources they consume are configured at deployment time).

Communication middleware defines a clear distinction between the responsibilities of the application and a generic software infrastructure. Middleware concerns are separated from application concerns. Application designers can concentrate on the design of software components by designing interfaces, behaviour and the composition of components.

QoS provisioning cross-cuts layers

The separation of concerns imposed by communication middleware can be restrictive. In particular when an application has Quality of Service (QoS) requirements, such as availability, safety and

responsiveness, the degree to which these requirements can be met depends on the support of the middleware. For example: the responsiveness of CORBA interactions cannot be guaranteed for the simple reason that today's ORB implementations use TCP/IP as a transport protocol.

Most object communication middleware infrastructures provide no means to application designers for influencing the QoS they expect for an application. From the perspective of the middleware, QoS capabilities that are provided are usually not published. Even if the middleware QoS can be influenced, this is through proprietary interfaces thus compromising the portability of applications.

Concerning QoS support in object middleware, we conclude that this crosscuts the responsibilities of the application and the middleware layer.

Solution

We propose a solution whereby the object middleware allows application components to inspect and adapt QoS capabilities. Applications can obtain and configure the QoS properties of components, by interacting with a

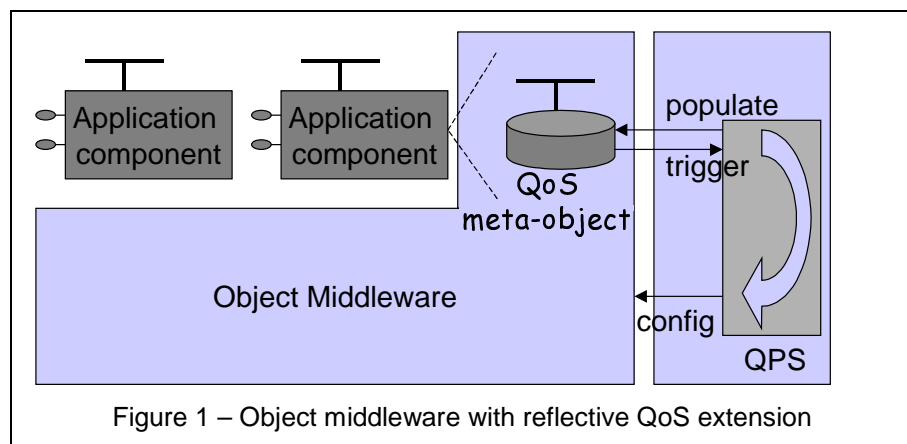


Figure 1 – Object middleware with reflective QoS extension

QoS meta-object. The middleware is responsible for creating and populating such a QoS meta-object with the appropriate QoS properties. In addition, changes to the QoS properties of the QoS meta-object triggers the middleware to deal with the changed QoS requirements. This trigger could lead to the reconfiguration of the internals of the middleware.

We've started to validate our approach by extending standard middleware (in our case an open source CORBA implementation) with a QoS Provisioning Service (QPS). Figure 1 depicts how QPS extends the object middleware. QPS is responsible for creating the QoS meta-objects and populating these with the QoS capabilities that the middleware offers. Our initial prototype has a very simple Meta-Object Protocol (MOP). QoS capabilities and QoS requirements are communicated as a stringified QML specification between the application and the QoS meta-object. QML is the Quality of service Modeling Language designed by HP laboratories [1,2].

To meet its responsibilities, QPS recursively applies reflection, through *inspection* of the QoS characteristics of the middleware and *adaptation* of the middleware internals.

QPS has been built as an extension to ORBacus [3], and makes use of the Open Communication Interface (OCI) to dynamically switch transport protocols if this is demanded to meet the QoS requirements of the

application [4]. The transport protocol components are internal components to the middleware and have an interface that QPS uses to obtain information about the QoS capabilities of each component.

Conclusion

The strict separation of application designer responsibilities from the middleware designer responsibilities can be restrictive when it comes to QoS provisioning. Application level QoS provisioning requires an end-to-end approach that crosscuts the middleware and application layers. We apply reflective principles for exchanging QoS capabilities and QoS requirements between the application and middleware layer.

To meet application QoS requirements, we extend the object middleware with QPS. This service uses reflection on the middleware internals to construct the compound QoS capabilities of the middleware and configures it according to the application requirements. This requires reflective interfaces on the internal components of the middleware.

Our initial prototype demonstrates the feasibility of the approach, but also raises some issues for further research. This includes the definition of a more sophisticated MOP for inspection and adaptation of the QoS properties both at the application level and at the level of internal middleware components.

References

- [1] S.Frolund and J.Koistinen, *QML: A Language for Quality of Service Specification*, 1998. <http://www.hpl.hp.com/techreports/98/HPL-98-10.html>
- [2] S.Frolund and J.Koistinen, *Quality of Service Aware Distributed Object Systems*, 1999. <http://www.hpl.hp.com/techreports/98/HPL-98-142.pdf>
- [3] Object Oriented Concepts, ORBacus <http://www.ooc.com/ob/>
- [4] A.T.van Halteren, A.Noutash, L.J.M.Nieuwenhuis, and M.Wegdam, *Extending CORBA with specialised protocols for QoS provisioning*, Proceedings of DOA'99.