QoS Provisioning in CORBA by Introducing a Reflective Aspect-Oriented Transport layer

Mehmet Aksit, Ali Noutash, Marten van Sinderen and Lodewijk Bergmans
AMIDST project, CTIT, University of Twente
P.O. Box 217, 7500 AE, Enschede, The Netherlands.
Email: {aksit | anoutash | sinderen | bergmans}@cs.utwente.nl

Commercially available middleware systems today offer *best-effort* Quality-of-Service (QoS) to the application programs. Due to the natural limitation of resources and the differences between the priorities and demands of applications, middleware systems must have the capability to offer varying degrees of QoS. The QoS requirements of middleware applications can be monitored and fulfilled by configuring the middleware [1]. This could be implemented, for example, by encapsulating the specific QoS concerns of middleware within the components and by installing the most suitable component on a particular QoS demand. Unfortunately, not all the QoS concerns of a distributed system [2] can be defined and encapsulated by the interfaces of components. So-called *crosscutting aspects* [3] hinder the adaptation of middleware systems since the implementations of QoS support techniques cannot be restricted to the implementations of components.

Consider, for example, a client object that sends a message to a set of server objects. The TCP/IP protocol can be used for this, in case only a few server objects have to receive the message. If, however, the client object wants to send the same message to a large set of server objects, a multicast protocol would be considerably more efficient than the TCP/IP protocol [5]. Transparent QoS management requires the system to automatically switch from TCP/IP to a multicast protocol in order to meet the QoS demands.

We have been experimenting with this scenario using the ORBacus CORBA implementation. ORBacus supports the CORBA/OCI architecture [5], which implements the functionality of ORB middleware with three layers, viz. the ORB layer, Messaging (GIOP) layer and Transport layer. These layers are separated from each other by means of an IDL interface. Consequently multiple implementations of each layer can be supported in a CORBA/OCI implementation [4][5]. However supporting multiple implementation of a layer can be still problematic. For example in the ORBacus implementation, the GIOP (messaging) layer first issues a call on the Transport layer. The Transport layer returns a buffer object to be filled in by the GIOP layer. When the buffer is full, the GIOP layer signals the Transport layer. The Transport layer then retrieves the data and provides it to the TCP/IP layer. Consequently in the implementation of ORBacus, static substitution of a Transport layer requires re-compilation of the GIOP layer [5]. Dynamic substitution of the Transport layers [5] isn't possible in the ORBacus implementation, although this is necessary to fulfil Transport-related QoS needs of QoS critical applications.

The origin of the problem is twofold. Firstly, in a design as exemplified by ORBacus, the buffering aspect is non-separable from the transport protocol aspect. Each Transport layer may make different assumptions about the buffering strategy and therefore swapping the protocol may be impossible without affecting other parts of the system. Secondly, the GIOP layer and Transport layers [5] are (necessarily) directly connected to each other in run time. This hinders implementation of a transparent QoS management system.

To overcome these problems, we propose a reflective and aspect-oriented technique based on the principle of Composition-Filters. Our proposal consists of the implementation of three aspects. Firstly, we think that the *buffering aspect* must be separated from the Transport layer. Secondly, we propose a *message reflection mechanism* [6] between the GIOP layer and a Transport layer. The reflected message is inspected and controlled by the QoS management system. Thirdly, we propose a *conditional delegation mechanism* [7] to dynamically dispatch the requests of the GIOP layer to the desired Transport layer. The Composition-Filters model is capable of expressing synchronisation [8], message reflection [6] and dynamic delegation [7] in a uniform manner. In our implementation, synchronisation, reflection and dynamic delegation aspects

are implemented by Wait, Meta and Dispatch filters, respectively. These filters are placed between the GIOP and Transport layers. Each filter provides extensibility within its aspect domain, such as reusable synchronisation specification. In addition, each aspect expressed by a filter can be composed easily with other aspects. This makes the system more extensible than the hard-wired solutions.

References

[1] D.L. Levine, S. Flores-Gaitan and D.C. Schmidt, An Empirical Evaluation of OS Support for Real-time CORBA Object Request Brokers. June 1999. Vancouver, British Columbia, Canada.

[2] Information Technology - Quality of Service- Guide to Methods and Mechanism- Technical Report Type III. 1997. Notes: ISO/IEC. TR 13243 (editor's draft 1.0) JTC1

[3] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes Aspect Oriented Programming ECOOP'97 Conference Proceeding, LNCS 1241, Springer-Verlag, 1997, pp. 220-242

[4] OMG, Revised version of the AT&T/TelTec/GMD Fokus IN/CORBA submission. 1998 OMG document number: telecom/98-06-03, http://www.ooc.com/ob

[5] A. van Halteren, A. Noutash, L. Nieuwenhuis, M. Wegdam, Extending CORBA with specialized protocols for QoS Provisioning, DOA99, pp318-327

[6] M. Aksit, K. Wakita, J. Bosch, L. Bergmans and A. Yonezawa, Abstracting Object-Interactions Using Composition-Filters, In object-based distributed processing, R. Guerraoui, O. Nierstrasz and M. Riveill (eds), LNCS, Springer-Verlag, pp. 152-184, 1993.

[7] M. Aksit, L. Bergmans and S. Vural, An Object-Oriented Language-Database Integration Model: The Composition-Filters Approach, ECOOP '92, LNCS 615, Springer-Verlag, pp. 372-395, 1992.

[8] Bergmans and M. Aksit, Composing Synchronisation and Real-Time Constraints, Journal of Parallel and Distributed Computing 36, pp. 32-52, 1996.