Proc. of the 23rd Int'l Computer Software & Applications Conf. (COMPSAC'99), Oct. 1999

Dynamic Client-Side Scheduling in a Real-Time CORBA System [*]

W. Feng
feng@lanl.gov

School of Elec. & Comp. Engg.
1285 Electrical Engg. Bldg.
Purdue University
W. Lafayette, IN 47907

Network Engineering, CIC-5
P.O. Box 1663, M.S. B255
Los Alamos National Laboratory
Los Alamos, NM 87545

Abstract

CORBA allows objects to communicate, independent of the specific techniques, languages, and platforms used to implement the objects. However, due to the multilevel software layering needed to provide this independence, CORBA cannot support real-time applications since it lacks essential quality-of-service (QoS) features. Recent work on real-time CORBA includes an off-line scheduled, hard, real-time system based on rate-monotonic scheduling and an on-line scheduled, best-effort, real-time system based on the earliest-deadline-first algorithm. The former provides QoS guarantees at the expense of run-time scheduling flexibility while the latter provides the complement. In this paper, we propose an approach which provides the advantages of both, that is, QoS guarantees and run-time scheduling flexibility.

1 Introduction

CORBA is a specification of an architecture and interface that allows an application to operate on objects (servers) in a transparent, independent manner, regardless of platform, operating system or locale considerations. This middleware simplifies and reduces the cost of application software development by providing a uniform view of a heterogeneous environment. The capability of simplifying and reducing the cost of software development is critical as software cycle times are getting shorter and shorter for increasingly complex software. A recent IEEE survey found that 30% of all software development projects are cancelled, 50% are more than 150% over budget, and only 60% of desired functionality on average is achieved [3].

While CORBA is well-suited for conventional non-real-time applications, the same cannot be said for real-time applications. Thus, over the past few years, there has been a movement to standardize middleware for real-time technologies.

[*]This research was performed while the author was a faculty member at the University of Illinois at Urbana-Champaign.

2 Related Work

CORBA provides an environment for programmers to automate as much of the development process as possible with maximum code re-use. However, to provide real-time services, CORBA must be enhanced to provide specifications for QoS, facilities to enforce QoS and real-time execution, and better performance.

The work in real-time CORBA has taken two major directions: the design of ORBs which can support hard and soft real-time applications using static scheduling and the design of ORBs which support dynamic scheduling but only with best-effort enforcement of timing constraints in applications. The former is implemented in a real-time middleware framework called TAO (The ACE ORB) [4], and the latter is part of the NRaD/URI real-time CORBA system [1]. While TAO works well for applications like avionics, it is not as well-suited for real-time applications which require dynamic admission of clients at run-time or efficient handling of changing resource requirements. On the other hand, while the NRaD/URI CORBA system allows clients to be admitted at run-time, it makes *no* guarantees that any of the clients' deadlines will be met.

3 Approach

Based on the current state-of-the-art, there exists no single ORB system that guarantees real-time performance *and* supports the dynamic admission of clients without having to reconfigure the system off-line. Our approach addresses this void by extending the TAO system to allow clients to be added (and removed) dynamically via an admissions test at run-time and would ensure scheduling feasibility.

3.1 Dynamic Clients

To allow new client requests to be handled on-line, we propose a modification to the real-time scheduling scheme provided in TAO. In addition to using TAO's Scheduling Service (SS) in a deterministic, static, and off-line manner for hard real-time applications, we also instantiate an SS object at run-time for each server object. This run-time SS object can be viewed as a *scheduling broker* [2] for the server object because it performs schedulability analysis on behalf of the server object. That is, as with the off-line SS object, the run-time SS object, or scheduling broker, performs schedulability analysis on all IDL operations that register with it in order to produce a schedule for the run-time scheduler. If a new request for an IDL operation on a server object fails the schedulability test, the client's request for that IDL operation is rejected. In short, this extension of TAO's SS allows clients' real-time tasks to be added to a server object's run-time schedule while maintaining real-time performance, i.e., meeting deadlines and guaranteeing QoS.

For example, Figure 1 shows a scenario with three clients, a server, and the server's scheduling broker. Before run-time, the off-line SS creates a real-time schedule for the server based on the QoS parameters of Clients 1 and 2, as in [4]. At run-time, this schedule is then used by the server's run-time scheduler to schedule Client 1 and Client 2 IDL operations (or tasks). When Client 3 makes a new request for an IDL operation on the server object, the request must first go through the scheduling broker for schedulability analysis in order to see if all the existing IDL operations from Clients 1 and 2 as well as the new IDL operations from Client 3 can be scheduled by their deadlines. Next, the scheduling broker indicates to Client 3 whether or not it has been admitted to the system with its current set of QoS parameters. If not, then Client 3 must wait for some of the server's resources to free up or modify its QoS parameters and try again. If, on the other hand, Client 3 is admitted, then the scheduling broker forwards a new schedule (which accounts for Client 3's IDL operations) to the server object for the run-time scheduler to execute. Once this is done, the server indicates readiness to Client 3, and at this point, Client 3 communicates its IDL operations directly to the server just as Clients 1 and 2 do.

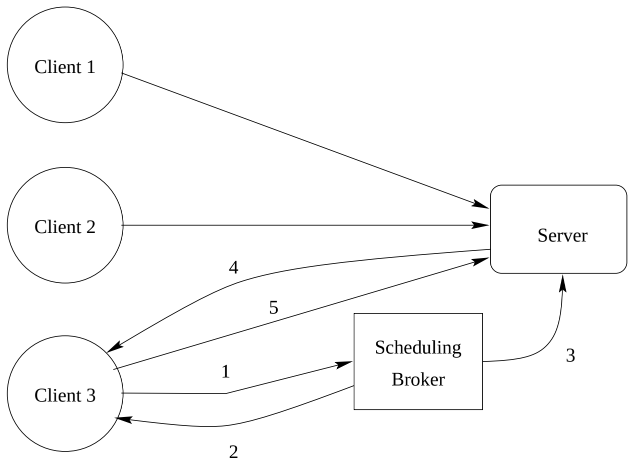

Figure 1: A Run-Time View of the Scheduling Broker

4 Summary

In this short paper, we briefly discussed two end-to-end quality-of-service frameworks: TAO by Schmidt et al. and NRaD/URI by Wolfe et al. Both TAO and NRaD/URI use CORBA to provide a uniform view of a heterogeneous environment, but they take complementary approaches toward realizing a real-time CORBA. First, TAO uses static real-time scheduling (RM) to meet the hard and soft deadlines of an avionics system while NRaD/URI uses dynamic real-time scheduling (EDF) which amounts to a best-effort approach towards enforcing timing constraints. Second, in TAO, the scheduling is done off-line and is not amenable to unplanned changes during run-time whereas the scheduling in NRaD/URI is done on-line and allows tasks to enter freely without admission control. Our proposal leverages the advantages of both approaches and extends the TAO system to handle the dynamic scheduling and re-configuration of client requests at run time, consequently providing for a more dynamic system.

References

[1] R. Ginis, V.F. Wolfe, and J. J. Prichard. The design of an open system with distributed real-time requirements. In *IEEE Real-Time Systems Symposium*, 1996.

[2] K. Nahrstedt. Experiences with qos brokerage and enforcement. In *Proceedings of 2nd International Conference on Multimedia Information Systems*, April 1997.

[3] ObjecTime. Overcoming the crisis in real-time software. Technical report, White Paper, 1997.

[4] D. C. Schmidt, D. L. Levine, and S. Mungee. The design of the tao real-time object request broker. *Computer Communications Journal*, Summer 1997.