Dynamic Real-Time Scheduling for Multi-Dimensional QoS-enabled Applications

Boeing Fellowship Research Grant Proposal

Candidate: Christopher D. Gill ⟨cdgill@cs.wustl.edu⟩
Advisor: Douglas C. Schmidt ⟨schmidt@cs.wustl.edu⟩

Department of Computer Science
Washington University
St. Louis, MO 63130, USA

1 Introduction and Motivation

Next-generation avionics [1] applications pose important challenges for managing resources, such as CPU cycles, under stringent, multi-dimensional quality of service (QoS) requirements, such as CPU share and operation latency. For example, as the number and diversity of features hosted on a Common Operational Flight Program (COFP) increases, the COFP must simultaneously meet end-to-end latency requirements for critical operations while avoiding starvation of non-critical operations. It is increasingly the responsibility of higher-level middleware, rather than lower-level operating systems, to schedule and control the competing, and often conflicting, QoS requirements in next-generation avionics applications.

Middleware resides between client and server applications and services in distributed systems. The goal of middleware is to integrate reusable software components to increase quality and to decrease the cycle-time and effort required to develop distributed applications. In particular, middleware simplifies application development by providing a uniform view of heterogeneous networks, protocols, and OS features.

At the heart of middleware are Object Request Brokers (ORBs), including the CORBA standard defined by the Object Management Group (OMG). CORBA middleware eliminates many tedious, error-prone, and non-portable aspects of developing and maintaining distributed applications using low-level network programming mechanisms like sockets. For example, ORBs reify core communication patterns to automate common programming tasks, such as object location, object activation, parameter marshaling/demarshaling, and connection and request demultiplexing.

To meet the QoS requirements of next-generation avionics applications, the scheduling strategies used for CORBA middleware must be flexible, efficient, predictable, and convenient to program. In addition, these strategies must be able simultaneously to resolve requirements in multiple QoS dimensions. For instance, a middleware scheduling strategy that is ideal to maximize throughput of a display image loading feature may be poorly suited for a route computation feature, which requires tighter latency and jitter bounds. When both image loading and route computation features are present, the scheduling strategy must balance their competing requirements.

These diverse requirements motivate the need for an open scheduling service framework, based on design patterns [2] and common distributed programming frameworks [3, 4], that can allow avionics applications sufficient flexibility to vary their scheduling strategies selectively.

2 Overview of Proposed Research Project

The scope of this project consists of three main objectives. First, we propose to extend our existing middleware infrastructure developed at Washington University by adding key new scheduling features to support avionics applications with multiple hard and soft QoS requirements. Second, in consultation with Boeing, we propose to develop a set of representative scheduling strategies to illustrate key tradeoffs for multi-dimensional QoS management. Third, we propose to conduct experiments to examine the effects of varying platform, application, and middleware factors on scheduling behavior and end-to-end enforcement of multi-dimensional QoS.

This section outlines the method by which we will leverage and enhance our earlier work to produce, validate, and demonstrate integrated middleware services that support a wide range of use-cases that are applicable to next-generation avionics applications. Our project deliverables will include tested and useable scheduling service middleware, example and test programs, technical papers, and system documentation in HTML.

2.1 Leveraged Work

To ensure the success of the proposed project, we will leverage our existing expertise and ongoing research results in communication middleware [5, 6, 7, 8, 1, 3, 4] to build upon the following components developed at Washington University:

The TAO ORB: TAO is a CORBA 2.2-compliant Object Request Broker (ORB) developed at Washington University's Center for Distributed Object Computing (DOC) [4]. TAO is an open-source, standards-based, high-performance, real-time ORB endsystem that supports applications with hard and soft QoS requirements, as well as "best-effort" requirements, and is the first ORB to support end-to-end QoS guarantees over ATM/IP networks [9]. TAO's features and optimizations include an ORB Core that minimizes context switching, synchronization, dynamic memory allocation, and data movement; a highly-scalable Object Adapter that demultiplexes requests in constant-time; an optimizing IDL compiler; a real-time I/O subsystem, and a global resource allocation and scheduling framework. For more information on TAO please see `www.cs.wustl.edu/~schmidt/TAO.html`.

TAO's scheduling service: TAO's scheduling framework currently supports QoS enforcement based on static fixed priority-based scheduling [1], dynamic scheduling, and hybrid static/dynamic scheduling [10]. While our existing scheduling framework is very useful for certain classes of operations, such as those with QoS requirements entirely based on latency and criticality, it does not yet meet the more diverse and multi-dimensional QoS requirements needed to effectively manage QoS for heterogeneous combinations of application features in a COFP environment. For more information on TAO's existing scheduling framework please see `www.cs.wustl.edu/~schmidt/corba-research-realtime.html`.

2.2 Proposed Effort

The primary research issues we will address in this project include the following:

Multi-dimensional QoS policies: defining families of policies that can appropriately balance requirements in multiple QoS dimensions such as priority, resource share, and timing constraints, for avionics applications. Rajkumar, et al. [**?**] have explored practical solutions to computationally complex problems of resource allocation in multiple QoS dimensions , in the context of the RT-Mach operating system. We propose to build on these results in defining appropriate and efficient policies for managing multiple QoS dimensions in open distributed middleware running on heterogeneous OS platforms.

Multi-dimensional QoS enforcement: designing QoS enforcement frameworks that can flexibly implement the appropriate mechanisms to enforce multi-dimensional QoS policies. Wang, et al. [11] have described a general framework for enforcing multi-dimensional QoS requirements in the RED-Linux kernel. We propose to explore the applicability of their model to TAO's strategized scheduling framework. In addition, we plan to explore the use-cases for distributed avionics applications, to design specific mechanisms and optimizations for effective multi-dimensional QoS enforcement within that more general model.

Combining multi-dimensional QoS requirements: investigating the impact of different combinations of application QoS requirements on end-to-end performance. We plan to conduct empirical measurements of the effects of varying application QoS requirements in multiple dimensions, for combinations of diverse features such as image load and route computations.

To support such diverse QoS requirements as latency, CPU share, and throughput (*e.g.*, for non-critical operations), we propose to extend TAO's strategized scheduling framework to address fairness via share-based scheduling capabilities. We plan to design and implement scheduling strategies that represent canonical resolutions of priority- and fairness-based design issues. Further, we propose to examine the use cases for distributed avionics applications with both hard and soft QoS requirements to define and document the patterns and programming guidelines needed to configure these strategies in real-world avionics applications.

Increasing the flexibility of TAO's strategized scheduling service architecture in this way is motivated by the following two design goals:

- **Hiding complexity:** shielding application developers from unnecessary implementation details of alternative scheduling strategies, so as to improve the system's reliability and maintainability.

- **Increasing flexability:** decoupling the strategy for priority and share assignment from the resource allocation model so the two can be varied independently, so as to increase the system's flexibility to adapt to varying application requirements and platform features.

Using the existing capabilities of TAO's strategized scheduling framework, along with the proposed extensions described above, we plan to investigate the effects of various priority and share based scheduling strategies, along with varying application requirements and platform characteristics, on end-to-end scheduling behavior.

Our research hypothesis is that varying each factor and observing the resulting behavior will expose relationships between these factors and the behavior of distributed QoS-enabled applications. This, in turn, will suggest techniques for configuring these factors in real-world avionics software systems.

2.3 Method of Accomplishment and Expected Results

In consultation with Boeing, we propose to perform the following tasks in order to develop the applications and conduct the experiments outlined below:

1. Extend our previous work on classifying QoS requirements of representative COFP and avionics application features to produce a complete taxonomy of known QoS dimensions in this domain.

2. Identify combinations of features that produce canonical systems of design forces within the QoS taxonomy, and identify, document, and validate the *patterns* that resolve these systems of forces to configure successful scheduling strategies in real-world avionics applications.

3. Examine the effects of varying application defined characteristics, such as criticality and CPU share, over a range of quantifiable characteristics, such as execution time and invocation frequency, for various combinations of features.

2.4 Benefits and Value to Boeing

For end-users and applications running in Boeing operating environments, the benefits of this research project will include: (1) advanced QoS management and scheduling capabilities over open source, standards-based, high-performance, real-time middleware that is (2) optimized to run on a range of operating systems, and that can (3) interoperate with and enhance existing CORBA-based services and toools; and (4) new application-oriented capabilities for adaptive enforcement of diverse resource scheduling QoS requirements.

For Boeing application developers and maintainers, the benefits of our research project will include (1) an environment for developing portable applications that take full advantage of the available resources, hardware improvements, and COTS OS/middleware capabilities, (2) easier maintenance and upgrading of these applications to add new capabilities and to support new hardware and OS/middleware, (3) ease in developing single applications that can run on many different configurations and in the presence of transient system overloads, (4) higher level, component-based and OO applications that are efficient, scalable, and predictable, yet are easier to maintain and are reusable, (5) separation of the functional processing capabilities of applications from the QoS scheduling aspects.

3 Project Objectives and Goals

The goals of this project include the following:

1. To expand the capacity of CORBA middleware to support application QoS in the face of varying requirements of distributed applications and varying features of available OS platforms.

2. To determine empirically the features required to build middleware endsystems that can enforce hard and soft end-to-end application QoS guarantees in multiple dimensions.

3. To identify, document, and validate design patterns and programming guidelines to configure successful scheduling strategies in real-world QoS-enabled distributed applications.

In order to meet these goals, the project will achieve the following objectives:

1. Developing new strategies for managing both hard and soft QoS requirements and integrating them into our open-source CORBA scheduling framework.

2. Conducting a systematic set of experiments to demonstrate the effects of varying application and middleware factors on scheduling behavior and end-to-end QoS enforcement.

3. Identifying, documenting and validating patterns for the effective use of QoS enforcement in middleware for combinations of application features with different QoS needs.

4 Long Term Impact to the Avionics Industry

The increasing demand to migrate new information management capabilities into COFP systems demonstrates an important technological role played by operational flight software. Such software acts as an intermediary between hardware components, such as sensors and actuators, that produce information and respond to commands and human human users, such as the pilot and navigator, who receive information and issue commands.

The next generation of avionics mission computing systems must provide capabilities previously seen only in resource-rich workstation environments, while still satisfying stringent real-time and resource constraints inherent to the avionics domain. Such software must have a whole new set of characteristics and powerful features for managing these competing requirements. Most importantly, it should be based on standard, completely open middleware. Middleware is software that users do not see, but which brokers communication and computing between distributed application components.

Efficient, predictable, scalable, and flexible middleware is a key enabling technology to realize the vision of powerful information management capabilities in a highly constrained environment. Unfortunately, first generation open standard middleware, such as CORBA, does not provide adequate QoS enforcement for performance-sensitive avionics applications.

There are several primary measures of success, some concrete and some abstract:

Technology transfer impact on Boeing R&D: *i.e.*, the degree to which the QoS-enabled software components and scheduling strategies developed in the project are adopted by Boeing Research and Development groups. Our group at Washington University has a long history of successfully transferring our middleware research into tools and products that are used widely used by our sponsors for commercial and research projects, such as telecommunications, network management, avionics mission computing, satellite control systems, electronic imaging systems, and financial services. Examples of our many successful technology transfer experiences are documented at www.cs.wustl.edu/~schmidt/users.html.

For this project, we will leverage our extensive user-base and augment it with a development and deployment process that emphasizes rapid prototyping, periodic demonstrations and course correction evaluations, and coordination with Boeing collaborators. The specific metrics for measuring the success of our technology transfer will involve determining the impact of the QoS-enabled TAO and application components in Boeing Research and Development organizations.

Intellectual contributions: *i.e.*, the proposed research will expand the available body of knowledge and experience related to QoS specification and enforcement in middleware for avionics applications. The long-term impact of our research will yield advances in the state-of-the-art for open standard middleware, so that it can address application QoS in the face of heterogeneous application requirements and platform features, as well as dynamically changing network and endsystem conditions. By documenting key design patterns and network programming guidelines, this research will also advance the state-of-the-practice in developing open standard middleware, and in turn will promote the realization of powerful information management capabilities in highly constrained avionics mission computing environments.

The specific metrics for measuring the success of our intellectual contribution will involve joint publications with Boeing collaborators in top journals and conferences.

Standardization: *i.e.*, the ability to transfer the scheduling strategies and QoS-enabled middleware components developed as part of this research into industrial standardization efforts, such as the OMG's CORBA. The Center for Distributed Object Computing at Washington University has been very active in the CORBA standardization effort, particularly the Real-time CORBA Special Interest Group, which is standardizing both fixed-priority and dynamically scheduled CORBA middleware.

The specific metrics for measuring the success of our standardization efforts will involve working jointly with Boeing collaborators to get the scheduling strategies and QoS-enabled middleware components adopted in industry standards.

References

[1] T. H. Harrison, D. L. Levine, and D. C. Schmidt, "The Design and Performance of a Real-time CORBA Event Service," in *Proceedings of OOPSLA '97*, (Atlanta, GA), ACM, October 1997.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.

[3] D. C. Schmidt, "ACE: an Object-Oriented Framework for Developing Distributed Applications," in *Proceedings of the* 6^{th} *USENIX C++ Technical Conference*, (Cambridge, Massachusetts), USENIX Association, April 1994.

[4] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design and Performance of Real-Time Object Request Brokers," *Computer Communications*, vol. 21, pp. 294–324, Apr. 1998.

[5] J. Hu, S. Mungee, and D. C. Schmidt, "Principles for Developing and Measuring High-performance Web Servers over ATM," in *Proceeedings of INFOCOM '98*, March/April 1998.

[6] I. Pyarali, T. H. Harrison, and D. C. Schmidt, "Design and Performance of an Object-Oriented Framework for High-Performance Electronic Medical Imaging," in *Proceedings of the* 2^{nd} *Conference on Object-Oriented Technologies and Systems*, (Toronto, Canada), USENIX, June 1996.

[7] P. Jain, S. Widoff, and D. C. Schmidt, "The Design and Performance of MedJava – Experience Developing Performance-Sensitive Distributed Applications with Java," *IEE/BCS Distributed Systems Engineering Journal*, 1998.

[8] D. C. Schmidt, "A Family of Design Patterns for Application-level Gateways," *The Theory and Practice of Object Systems (Special Issue on Patterns and Pattern Languages)*, vol. 2, no. 1, 1996.

[9] G. Parulkar, D. C. Schmidt, and J. S. Turner, "$\mathrm{a^It^Pm}$: a Strategy for Integrating IP with ATM," in *Proceedings of the Symposium on Communications Architectures and Protocols (SIGCOMM)*, ACM, September 1995.

[10] C. D. Gill, D. L. Levine, and D. C. Schmidt, "Evaluating Strategies for Real-Time CORBA Dynamic Scheduling," *The International Journal of Time-Critical Computing Systems, special issue on Real-Time Middleware*, 1999, to appear.

[11] S. Wang, Y.-C. Wang, and K.-J. Lin, "A General Scheduling Framework for Real-Time Systems," in *IEEE Real-Time Technology and Applications Symposium*, IEEE, June 1999.