Towards Real-Time Adaptive QoS Management in Middleware for Embedded Computing Systems

Christopher D. Gill and David L. Levine  
{cdgill,levine}@cs.wustl.edu  
Department of Computer Science  
Washington University  
St. Louis, MO 63130, USA

Douglas C. Schmidt  
schmidt@uci.edu  
Department of Electrical and Computer Engineering  
University of California, Irvine  
Irvine, CA 92697, USA

1 Introduction

Meeting the quality of service (QoS) requirements of distributed real-time mission-critical embedded systems is hard [1]. These systems impose timing constraints on both critical and non-critical operations, across distributed endsystems and networks. Likewise, these systems often require embedded processor and network resources, so that both time and space utilization are constrained. Many of these systems also must respond to rapidly changing environmental conditions, e.g., by adapting their timing constraints at run-time.

Supporting the needs of these types of systems requires quality of service (QoS) management that (1) is adaptive to changing constraints, (2) performs such adaptation in real-time, (3) integrates distinct canonical QoS management functions, and (4) operates at an appropriate architectural level for end-to-end QoS management in distributed real-time mission-critical embedded systems. Moreover, solutions that apply across commercial-off-the-shelf (COTS) and proprietary heterogeneous networks and endsystems are likely to offer greater applicability in practice.

2 Solution Approach

This research offers a middleware-based solution in which lower-level QoS management services are leveraged where possible, or are provided in middleware when necessary. This solution also complements and provides services to higher-level COTS and custom middleware QoS management techniques from the broader research community [2, 3, 4].

The primary contributions of our research are focused on Kokyu1, which is an open-source integrated middleware framework that supports adaptive distributed admission control strategies and mechanisms for end-to-end QoS management in real-time embedded middleware. The Kokyu project also provides a foundation for ongoing efforts to identify and document design patterns [5] for integrated real-time adaptive QoS management in the context of distributed real-time mission-critical embedded system middleware and applications.

2.1 End-to-End Admission Control

Distributed real-time mission-critical embedded system requirements pose new challenges for resource allocation. For adaptive rate reconfiguration, remote dependencies require an end-to-end admission control protocol to ensure that (1) appropriate adaptation is performed on each endsystem to maintain the end-to-end timing constraints and (2) sufficient resources are feasibly reserved on each endsystem. Thus, it is essential to identify and develop policies and mechanisms for end-to-end real-time admission control that address these challenges.

For example, consider a sensor processing application with sampling operations and processing operations on different endsystems [6]. Depending on the environment and application state, sensor sampling operations may run at any one of a set of rates. Whenever the sampling portion of the application is ready to adapt by changing the rate at which it samples the sensor input, the following two activities must occur:

1Kokyu is a Japanese word meaning literally “breath”, but also implying timing and coordination.
Integrating mechanisms within the Kokyu framework offers improved adaptive real-time performance. For example, consider two mechanisms: (1) classifying operations for dispatch priority assignment, and (2) rate selection for adaptive admission control (described in Section 2.1). If the possible rates of all operations are known at admission control time, priority assignment can be performed at the same time as rate selection, as illustrated in the following figure:

We integrate classification and rate selection in Kokyu as follows:

**Operation Characteristics:** We store information about operation characteristics, e.g., period, criticality, and worst-case, average, and best-case execution times, in a RT_Info descriptor. The application, or a higher level management layer, stores the values for the characteristics of each operation in its RT_Info descriptor.

**Denormalized Tuples:** In adaptive systems, multiple possible values may be specified for some operation characteristics. The Kokyu framework must select a value for each such characteristic. Priority assignment may need to be performed as well because priority assignment may depend on value selection (e.g., RMS [7] depends on selected rates).

To reduce the computational complexity of admission control, it is useful to perform both selection of values for operation characteristics and priority assignment in the same pass. This can be done by (1) de-normalizing the RTInfos and the sets of possible values into a sequence of tuples, (2) sorting the tuples according to both the priority assignment and admission control policies, and (3) performing value selection and priority assignment on the sorted sequence.

**Composing Strategies:** By using a stable sorting technique, or by composing admission control and priority assignment comparisons, the constraints of both policies can be met, assuming they are not contradictory. Moreover, it may be possible to apply increasingly efficient sorting algorithms depending on the information known about the operations at admission control time. For example, if all the rates are known in advance, it may be possible to apply an $O(n)$ algorithm, e.g., radix sort. Otherwise, an $O(n \log n)$ comparison sort, e.g., heap sort, is needed.

3 Concluding Remarks

Our end-to-end adaptive QoS management approach is being incorporated into the Kokyu framework. Kokyu is being developed to extend and enhance the context of the ACE [8] and TAO [4] open-source COTS middleware frameworks. Kokyu offers flexibility for extension, configuration, and integration of either COTS or custom policies and mechanisms, to meet stringent distributed real-time mission-critical embedded system requirements. This in turn will provide application developers with a high-level foundation for rapid system development across heterogeneous networks and endsystems.

References


