Feedback Control Real-Time Scheduling:
Framework, Modeling, and Algorithms*

CHENYANG LU  chenyang@cs.virginia.edu
JOHN A. STANKOVIC  stankovic@cs.virginia.edu
SANG H. SON  son@cs.virginia.edu
Department of Computer Science, University of Virginia, Charlottesville, VA 22903, USA

GANG TAO  g9s@ee.virginia.edu
Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22903, USA

Abstract. This paper presents a feedback control real-time scheduling (FCS) framework for adaptive real-time systems. An advantage of the FCS framework is its use of feedback control theory (rather than ad hoc solutions) as a scientific underpinning. We apply a control theory based methodology to systematically design FCS algorithms to satisfy the transient and steady state performance specifications of real-time systems. In particular, we establish dynamic models of real-time systems and develop performance analyses of FCS algorithms, which are major challenges and key steps for the design of control theory based adaptive real-time systems. We also present a FCS architecture that allows plug-ins of different real-time scheduling policies and QoS optimization algorithms. Based on our framework, we identify different categories of real-time applications where different FCS algorithms should be applied. Performance evaluation results demonstrate that our analytically tuned FCS algorithms provide robust transient and steady state performance guarantees for periodic and aperiodic tasks even when the task execution times vary by as much as 100% from the initial estimate.

Keywords: real-time scheduling, feedback control, Quality of Service, modeling, unpredictable environment

1. Motivation and Introduction

Real-time scheduling algorithms fall into two categories: static and dynamic scheduling. In static scheduling, the scheduling algorithm has complete knowledge of the task set and its constraints, such as deadlines, computation times, precedence constraints, and future release times. The rate monotonic (RM) algorithm and its extensions (Klein et al., 1993; Liu and Layland, 1973) are static scheduling algorithms and represent one major paradigm of real-time scheduling. In dynamic scheduling, however, the scheduling algorithm does not have complete knowledge of the task set or its timing constraints. For example, new task activations, not known to the algorithm when it is scheduling the current task set, may arrive at a future unknown time. Dynamic scheduling can be further divided into two categories: scheduling algorithms that work in resource sufficient environments and those that work in resource insufficient environments. Resource

* Supported in part by NSF grants CCR-9901706, CCR-0098269, and EIA-9900895, and DARPA grants F33615-01-C-1905 and N00014-01-1-0576. This paper is an extension to previous papers published in the Proceedings of IEEE Real-Time Systems Symposium (Lu et al., 1999, 2000).
sufficient environments are systems where the system resources are sufficient to \textit{a priori} guarantee that, even though tasks arrive dynamically, at any given time all the tasks are schedulable. Under certain conditions, earliest deadline first (EDF) (Liu and Layland, 1973; Stankovic et al., 1998) is an optimal dynamic scheduling algorithm in resource sufficient environments. EDF is a second major paradigm for real-time scheduling. While real-time system designers try to design the system with sufficient resources, because of cost and unpredictable environments, it is sometimes impossible to guarantee that the system resources are sufficient. In this case, EDF’s performance degrades rapidly in overload situations. The Spring scheduling algorithm (Zhao et al., 1987) can dynamically guarantee incoming tasks via on-line admission control and planning and thus is applicable in resource insufficient environments. Many other algorithms (Stankovic et al., 1998) have also been developed to operate in this way. These admission-control-based algorithms represent the third major paradigm for real-time scheduling. However, despite the significant body of results in these three paradigms of real-time scheduling, many real world problems are not easily supported. While algorithms such as EDF, RM and the Spring scheduling algorithm can support sophisticated task set characteristics, they are all ‘‘open loop’’ scheduling algorithms. Open loop refers to the fact that once schedules are created they are not ‘‘adjusted’’ based on continuous feedback. While open-loop scheduling algorithms can perform well in predictable environments in which the workloads can be accurately modeled (e.g., traditional process control systems), they can perform poorly in \textit{unpredictable} environments, i.e., systems whose workloads cannot be accurately modeled. For example, systems with open-loop schedulers such as the Spring scheduling algorithm are usually designed based on \textit{worst-case} workload parameters. When accurate system workload models are not available, such an approach can result in a highly underutilized system based on an extremely pessimistic estimation of workload.

In recent years, a new category of soft real-time applications executing in open and unpredictable environments has been rapidly growing (Stankovic et al., 1999). Examples include open systems on the Internet such as online trading and e-business servers, and data-driven systems such as smart spaces, agile manufacturing, and defense applications such as C4I. For example, in an e-business server, neither the resource requirements nor the arrival rate of service requests are known \textit{a priori}. However, performance guarantees are required in these applications. Failure to meet performance guarantees may result in loss of customers, financial damage, liability violations, or even mission failures. For these applications, a system design based on open loop scheduling can result in an extremely expensive and underutilized system.

As a cost-effective approach to achieve performance guarantees in unpredictable environments, adaptive scheduling algorithms have been recently developed. While early research on real-time scheduling was concerned with guaranteeing complete avoidance of undesirable effects such as overload and deadline misses, adaptive real-time systems are designed to handle such effects dynamically. There remain many open research questions in adaptive real-time scheduling. In particular, how can a system designer specify the performance requirement of an adaptive real-time system? And how can a designer systematically design a scheduling algorithm to satisfy system performance specifications? The design methodology for automatic adaptive systems has been developed in feedback control theory (Franklin et al., 1998). However, feedback control theory has