Real-Time Systems, 22, 5–7, 2002

Introduction to the Special Issue on Flexible Scheduling

Real-time computing systems were originally developed to support safety critical control applications characterized by stringent timing constraints. In these applications, missing a single deadline can jeopardize the entire system behavior and even cause catastrophic consequences. Hence, real-time software has to be designed under worst-case assumptions and executed with predictable kernel mechanisms to meet the required performance in all anticipated scenarios. Often, systems are dedicated to a specific purpose, involving special hardware. Tasks and their properties are identified through a static design before the system is deployed, thus ensuring a correct behavior, but preventing changes during system operation.

In recent years, the increasing power of personal computers and network technologies extended the domain of real-time systems to novel application fields, including multimedia computing, graphic animations, and virtual reality.

With respect to traditional control environments, such new application areas are characterized by less stringent timing constraints. In these systems, missing a deadline is acceptable and can be tolerated up to a certain degree. Nevertheless, some control on a system's performance is still required, although with more relaxed constraints.

Often, modern applications demand various types of tasks and constraints within the same system. The requirements on tasks may also change dynamically. While off-line guarantees are still essential for meeting a minimum performance, different types of requirements and runtime changes have to be included in the system analysis, such as demands on quality of service or acceptance probabilities.

Using hard real-time systems to support soft real-time applications is possible but often inefficient. In fact, when application tasks are characterized by computation times with a large variance (as in video processing), adopting worst-case assumptions in the system design causes a waste of resources.

On the other hand, using non real-time kernels is also inadequate, since no individual constraints can be enforced on a system's activities.

Moreover, in classical real-time and non real-time operating systems, coexistence of schedulers is typically not allowed, thus designers have to select a single paradigm, even when a combination of different policies would be desirable.

Modern real-time applications can be properly supported if the operating system is designed to accommodate changes in the scheduler, without time consuming redesign cycles.

The challenge of next generation real-time systems is now clear. They should enforce timing constraints with a certain degree of flexibility, aimed at achieving a trade off between predictability in the performance and efficiency in the resource utilization.

Flexibility also means a system's capability of on-line adaptation. Most traditional real-time kernels are inflexible because they were designed for systems with static characteristics. For this reason, most of the real-time theory proposed in the recent years was developed assuming simple task models with fixed parameters.

Modern real-time applications are more dynamic in nature: tasks are activated by event arrivals, periods of cyclic computations can vary based on the current system's status, computation times may have large variations, and so on. Often they demand the coexistence and cooperation of diverse scheduling algorithms to combine properties, e.g., to integrate deterministic and flexible activities in the same system. Algorithms might even change during system's runtime to better adapt to environment variations.

In such a new scenario, the basic assumptions made on the classical scheduling theory are not valid any more, and new approaches are needed to handle these situations in a predictable fashion.

The papers collected in this special issue provide an important contribution in solving the problems outlined above. They cover different crucial aspects involved in a real-time system, such as aperiodic task management (Bernat and Burns), disk scheduling, (Shenoy and Vin), quality-of-service control (Brant and Nutt), network scheduling (Wang, Wang, and Lin) and imprecise computation models (Terrasa, Garcia-Fornes, and Botti).

The first paper, ''Cello: A Disk Scheduling Framework for Next Generation Operating Systems'', by Prashant Shenoy and Harrick Vin, proposes a framework which enables the coexistence of diverse application specific schedulers via two-level hierarchy: the lower level uses a common scheduler for allocating bandwidth to application classes; the higher level provides for various schedulers within these bandwidths.

The second paper, ''Multiple Servers and Capacity Sharing for Implementing Flexible Scheduling'', by Guillem Bernat and Alan Burns, proposes a new scheduling mechanism which allows multiple aperiodic servers to share their capacity to exploit the unused computation time and increase a system's efficiency. In the presence of tasks with different criticality and timing characteristics (such as hard periodic tasks and soft aperiodic requests) the proposed approach is effective for improving aperiodic responsiveness while guaranteeing the critical tasks.

The third paper, ''Flexible Soft Real-Time Processing in Middleware'' by Scott Brandt and Gary Nutt, describes a framework for implementing a Quality-of-Service resource manager as middleware in general-purpose operating systems, such as Solaris or Linux. The proposed approach provides a useful environment for developing soft real-time applications on top of existing operating systems.

The fourth paper, ''Integrating Priority with Share in the Priority-Based Weighted Fair Queueing Scheduler for Real-Time Networks'', by Song Wang, Yu-Chung Wang, and Kwei-Jay Lin, presents a new queuing algorithm obtained by combination of scheduling paradigms whose performance can be tuned through appropriate scheduling parameters.

The fifth paper, ''Flexible Real-Time Linux: A Flexible Hard Real-Time Environment'', by Andreas Terrasa, Ana García Fornes, and Vincent Botti, presents an implementation of the imprecise computation model in the RT-Linux kernel. The paper also describes how to adapt the implementation of two resource sharing protocols to the proposed framework.

Acknowledgments

We would like to express our gratitude to the reviewers for their excellent work under rigid time constraints. We would like to acknowledge the valuable contributions and good cooperation with the authors of the papers in this issue, as well as with the Editor in Chief Jack Stankovic and with Melissa Fearon of Kluwer for their help and advice in managing the process.

Gerhard Fohler is a senior lecturer and leader of the predictably flexible real-time systems group at the Department of Computer Science at Malardalen University, Sweden. He received his Ph.D. from Vienna University of Technology in 1994 for research towards flexibility for offline scheduling in the MARS system. He then worked at the University of Massachusetts at Amherst as postdoctoral researcher within the SPRING project. During 1996–1997, he was a researcher at Humboldt University Berlin, investigating issues of adaptive reliability and real-time. Gerhard Fohler is currently chairman of the Technical Committee on Real-Time Systems of EUROMICRO.

Giorgio C. Buttazzo is an Associate Professor of Computer Engineering at the University of Pavia, Italy. He graduated in Electronic Engineering at the University of Pisa in 1985, received a Master in Computer Science at the University of Pennsylvania in 1987, and received a Ph.D. in Computer Engineering at the Scuola Superiore Sant'Anna of Pisa in 1991. During 1987, he worked on active perception and real-time control at the G.R.A.S.P. Laboratory of the University of Pennsylvania, in Philadelphia. From 1991 to 1998, he held a position of Assistant Professor at the Scuola Superiore Sant'Anna of Pisa, doing research on robot control systems and real-time scheduling. His main research interests include real-time operating systems, dynamic scheduling algorithms, quality of service control, multimedia systems, advanced robotics applications, and neural networks.