To appear in the Proceedings of the Workshop on the Role of Real-Time in Multimedia/Interactive Computing Systems, Raleigh-Durham, NC, November 1993.

# Applying Hard Real-Time Technology to Multimedia Systems

Clifford W. Mercer[†], Ragunathan Rajkumar[‡], and Hideyuki Tokuda[†]

School of Computer Science[†]
Software Engineering Institute[‡]
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
{cwm,raj,hxt}@cs.cmu.edu

**Abstract**

The real-time community tends to contend that multimedia systems *are* indeed real-time systems, while the multimedia community points to problems such as video databases on disks and dynamic environments as issues unique to multimedia systems. Also, historically, real-time and multimedia systems have had different roots, and therefore embody different schools of thought. In this paper, we consider the similarities and differences between hard real-time systems and multimedia systems. We argue that multimedia systems do exhibit new requirements but that many of the insights and techniques developed for real-time systems can be fruitfully applied to the design and implementation of multimedia systems. We then propose augmenting the hard real-time system framework with mechanisms to support the dynamic nature of multimedia systems. As a matter of fact, one of the goals of the Real-Time Mach operating system being developed at Carnegie Mellon University is to be able to support both hard real-time systems as well as multimedia systems. As a specific example of hard real-time technology being applicable to multimedia systems, we present our model of processor reservation which uses hard real-time scheduling theory as a basis for admission control and scheduling.

## 1. Introduction

Hard real-time systems and multimedia systems share many characteristics. A well-accepted definition of hard real-time systems is that in such a system the time at which computations are performed directly affect the correct operation of the system. Similarly, a multimedia system also needs to support video, audio and/or animation, which are data types with time as a fundamental component. Many common issues arise from this fundamental dependence on time. The programmer/application must have a way to specify the timing constraints of programs to the system, preferably using sophisticated abstractions that support high-level reasoning about timing constraints. A scheduler must take this timing information into account and ideally provide some assurances about the degree to which timing constraints will be honored. And other parts of the system must participate in the integrated framework so that assurances given to an application will be satisfied. For example, unbounded priority inversion can cause unacceptable latencies in both kinds of systems and must therefore be avoided.

This work was supported in part by a National Science Foundation Graduate Fellowship, and by the U.S. Naval Ocean Systems Center under contract number N00014-91-J-4061. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of NSF, NOSC, or the U.S. Government.

While real-time systems and multimedia systems share a basic concern for time, there are also many differences in the degree to which timely computation must be supported. Many embedded hard real-time systems are safety-critical, and system failure can lead to loss of property, life and/or increased costs. On the other hand, multimedia systems are frequently used to provide a friendly interface to humans for purposes of communication, information access or even entertainment. Real-time systems tend to be static in terms of the task set(s) for which the system is designed, whereas advanced multimedia systems tend to be more dynamic with applications being created and terminated arbitrarily. Multimedia systems thus have some of the flavor of time-sharing systems or personal workstations. Hard real-time systems typically require their timing constraints to be strictly enforced, and dealine misses can lead to system failure. In multimedia systems, if timing constraints cannot be guaranteed, one can typically make quality of service adjustments without major repercussions. For example, if a video clip at 30 frames per second and 24 bits per pixel cannot be processed on time, the same clip may be transmitted and processed at 10 frames per second and 8 bits per pixel.

Despite these significant differences between real-time and multimedia systems, we claim that multimedia systems can indeed take advantage of the technologies that have been developed for real-time systems. We believe that the design and implementation of multimedia systems can greatly benefit from the experience and insights gained in managing time explicitly as a resource to achieve high CPU, bus and network utilization while still satisfying timing constraints. For example, multimedia systems can benefit from research on

- scheduling algorithms that use timing information to make decisions [7],
- programming models which provide high-level abstractions for programming with timing constraints [2, 10],
- language constructs for programming with timing constraints [1, 4],
- synchronization algorithms that are integrated with the processor scheduling algorithms to provide consistent system-wide scheduling [13], and
- preemptible services, such as network services, which promote consistent scheduling [11].

In the rest of this position paper, we argue that hard real-time techniques can be profitably applied to multimedia systems. For example, the notion of priority inheritance to avoid priority inversion problems [13, 14] is already used or being considered for use in multimedia-oriented systems [12, 3]. In this paper, we shall illustrate the applicability of hard real-time system techniques to multimedia systems using as an example our approach to processor reservation. This approach uses a hard real-time scheduling algorithm as a basis for the reservation model. This model supports not only soft real-time requirements such as multimedia applications, but also hard real-time programming as well. We conclude by advocating the augmentation of a hard real-time framework to bridge the gap between hard real-time rigidity and soft real-time flexibility.

## 2. A Unifying Framework For Hard Real-Time and Multimedia Systems

In this section, we outline our approach used in the Real-Time Mach operating system to support both hard real-time and multimedia applications. Our approach is based on the following observations about the goals of each kind of system.

The goals and performance metrics of a hard real-time system may be summarized as follows:

- *Predictable scheduling of time-constrained tasks:* Given the timing constraints of a set of tasks, the feasibility of the task set should be analyzed, and if determined to be feasible, its correct execution should be guaranteed with appropriate run-time support.

- *Fast response time for asynchronous events:* The system should be highly preemptive since those events whose timing constraints are particularly stringent require fast response.

- *Stability under overload conditions:* If a transient overload develops, the system must still be able to perform its critical functions.

Multimedia systems represent a much more dynamic context than hard real-time systems, and hence need to be more flexible. Some of their fundamental goals from a systems perspective are as follows:

- *Freedom to mix non-time-constrained programs with real-time programs:* A multimedia computing systems would ideally support traditional general-purpose computing as well as multimedia applications.

- *A scheduling policy that would accommodate a dynamically changing mix of programs:* The system should support dynamic creation and termination of multimedia and non-time-constrained applications. In particular, the scheduling policy must ensure that this flexibility does not interfere with the correct operation of the time-constrained programs.

- *Ability to change timing constraints dynamically based on performance feedback:* A single multimedia application may need to change its timing constraints to provide better quality of service or degraded quality of service. The system should provide flexible support for this mode of operation.

- *Support for reasonable supervisory control of the system:* A dynamic system requires some method for a system administrator to control activities on the system. It is possible for real-time programs to misbehave in such a system, and the system must prevent the situation where a misbehaving application completely takes over and hangs the system.

- *Accurate computation time usage measurement:* Measurement of resource usage times are useful in all types of systems but more so in hard real-time and multimedia systems. Many time-sharing systems use sampling to acquire approximate usage information, but in real-time systems and multimedia systems, long-term sampled statistics are not adequate. Usage measurements are useful for off-line performance tuning, quality of service changes as well as on-line scheduling.

Based on our experiences with the Real-Time Mach operating system, we believe that it is possible and practical to build a single framework which can satisfy both these classes of goals. A priority-based preemptive scheduling approach such as those based on rate-monotonic analysis has been used in practice to achieve predictable timing behavior in hard real-time systems. In order to deal with the dynamic requirements of multimedia systems, this priority-driven approach can be supplemented by two mechanisms: enforcement of resource usage, and two-way performance feedback between the application and the system. First, our processor reservation model discussed in the next section is an example of enforcing the maximum time that a task can use a processor. Secondly, feedback from the system can allow an application to scale its requirements up or down depending upon the system load. Such augmentation to a priority-driven approach leads to a unifying framework which combines hard real-time and multimedia system features in a way that allows the system to act as a hard real-time system when the task set is static, and as a soft real-time system when the task set is dynamic.

We now describe our processor reservation model.

## 3. Processor Capacity Reserves

*Processor capacity reserves* provide an abstraction for allocating processor cycles in a real-time system [8, 9]. It allows programs to request processor reservations which are expressed as a percentage of the processor and an associated period called the *reservation period*. For example, a program can request 30% of the processor every 100 *ms* which amounts to 30 *ms* of computation time every 100 ms. Alternatively, a program could request 30% of the processor every 10 ms, i.e. 3 *ms* of computation time for every 10 *ms* of real time.

This method of specifying reservations leads naturally to a periodic scheduling framework where each reserved activity gets the processor for its reserved computation time during each reservation period. This allows us to use a rate monotonic-based scheduling algorithm [5, 7]. Programs that are periodic fit easily into this framework, and aperiodic programs can be accommodated in one of two ways. For aperiodic activities which have short computation times and low response time requirements, we can set the reservation period based on the response time requirements using the aperiodic server methods [6, 15]. For aperiodic programs with longer computation times and longer deadlines, we can pick an appropriately long period as long as the delay in completing the activity does not exceed its deadline.

In a multimedia context, applications can be started at any time. Since there is a limit on the number of multimedia applications that can be supported simultaneously, an admission control policy regulates which applications will be guaranteed a minimum quality of service. The admission control policy should also reflect that fact that the system will require some percentage of the processor for certain housekeeping activities including overhead. Also, a percentage will be set aside for use by the system administrator so that no matter what other activities are going on in the system, the administrator will always be able to gain control of the system and any of its activities. In contrast, if vanilla fixed priority scheduling is used, a misbehaving high priority activity may take over the system, and the administrator has no recourse other than to reboot the system.

We note that the assumptions made in the analysis for the scheduling algorithms and admission control policies mentioned above must be satisfied for the reservation system to work. One important assumption is that periodic activities have fixed or at least bounded computation times. In hard real-time systems, this assumption is typically guaranteed by extensive simulation, instrumentation and/or analysis, ensuring that programs cannot overrun their allotted computation times. In dynamic systems, however, the programs themselves cannot be trusted to behave as advertised, and the scheduler must take responsibility for preventing misbehaving programs from interfering with the execution of other programs. In our model of processor capacity reserves, we accurately measure the processor usage of each reserved activity, and we incorporate a mechanism to identify programs that would overrun their reservations and thereby prevent the overruns from taking over cycles from other activities. The design is presented in more detail in a technical report [9], and in another paper we present a brief description of the model and an implementation in Real-Time Mach [8]. We are currently extending this enforcement philosophy of resource usage to other system resources as well.

## 4. Concluding Remarks

In this paper, we have explored the similarities and differences between hard real-time systems and multimedia systems. The similarities allow us to apply the accumulated techniques and knowledge yielded by real-time systems research to multimedia systems. However, multimedia systems also bring in new requirements mostly in terms of the need to support a dynamic environment. We propose that a real-time system framework, priority-driven preemptive scheduling, be augmented with additional enforcement and feedback mechanisms. These mechanisms relax the stringent requirements of hard real-time systems, and allow us the flexibility to make the system more dynamic or to tune the system for performance. We are currently experimenting with these mechanisms in the Real-Time Mach operating system. Our model of processor capacity reserves illustrates how real-time technology can be exploited in the more dynamic and often forgiving environment of a multimedia system.


## Acknowledgements

The authors would like to express their appreciation to Stefan Savage for his implementation effort on the processor reservation system in Real-Time Mach, and to the members of the ART and Mach projects at CMU.


## References


[1] Y. Ishikawa, H. Tokuda, and C. W. Mercer. An Object-Oriented Real-Time Programming Language. *IEEE Computer*, October 1992.

[2] K. Jeffay, D. L. Stone, and F. D. Smith. Kernel Support for Live Digital Audio and Video. *Computer Communications (UK)*, 15(6):388–395, July-August 1992.

[3] S. Khanna, M. Sebree, and J. Zolnowsky. Realtime scheduling in SunOS 5.0. In *Proceedings of the Winter 1992 USENIX Conference*, pages 375–390, San Francisco, CA, January 1992.

[4] E. Kligerman and A. D. Stoyenko. Real-Time Euclid: A Language for Reliable Real-Time Systems. *IEEE Transaction on Software Engineering*, SE-12(9):941–949, September 1986.

[5] J. P. Lehoczky, L. Sha, and Y. Ding. The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. In *Proceedings of the 10th IEEE Real-Time Systems Symposium*, pages 166–171, December 1989.

[6] J. P. Lehoczky, L. Sha, and J. K. Strosnider. Enhanced Aperiodic Responsiveness in A Hard Real-Time Environments. In *Proceedings of 8th IEEE Real-Time Systems Symposium*, pages 261–270, December 1987.

[7] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment. *JACM*, 20(1):46–61, 1973.

[8] C. W. Mercer, S. Savage, and H. Tokuda. Processor Capacity Reserves: An Abstraction for Managing Processor Usage. In *Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, October 1993.

[9] C. W. Mercer, S. Savage, and H. Tokuda. Processor Capacity Reserves for Multimedia Operating Systems. Technical Report CMU-CS-93-157, School of Computer Science, Carnegie Mellon University, May 1993.

[10] C. W. Mercer and H. Tokuda. The ARTS Real-Time Object Model. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, December 1990.

[11] C. W. Mercer and H. Tokuda. An Evaluation of Priority Consistency in Protocol Architectures. In *Proceedings of the IEEE 16th Conference on Local Computer Networks*, pages 386–398, October 1991.

[12] R. Needham and A. Nakamura. An Approach to Real-time Scheduling but is it Really a Problem for Multimedia? In *Proceedings of the Third International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 30–37, November 1992.

[13] R. Rajkumar. *Synchronization in Real-Time Systems: A Priority Inheritance Approach*. Kluwer Academic Publishers, 1991.

[14] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.

[15] B. Sprunt, L. Sha, and J. P. Lehoczky. Aperiodic Task Scheduling for Hard Real-Time Systems. *The Journal of Real-Time Systems*, 1(1):27–60, June 1989.