

# Integrated Processor Scheduling for Multimedia

Jason Nieh and Monica S. Lam  
Computer Systems Laboratory  
Stanford University  
Stanford, CA 94305  
{nieh, lam}@cs.stanford.edu

---

*The advent of multimedia ushers forth a growing class of applications that must manipulate digital audio and video within well-defined timeliness requirements. Existing processor schedulers are inadequate in supporting these requirements. They fail to allow the integration of these continuous media computations with conventional interactive and batch activities. We have created a new scheduler that provides integrated processor scheduling for all classes of computational activities. Our solution achieves optimal performance when all timeliness requirements can be satisfied, and provides graceful degradation when the system is overloaded. Though unique in the degree to which it allows users control over the dynamic sharing of processing resources, the scheduler does not impose any draconian demands on the user to provide information he does not have or does not choose to specify.*

## 1 Introduction

Applications that manipulate digital audio and video represent a new class of computations executed by workstation users. This new class of computations is known as *continuous media*. Continuous media activities are characteristic of applications that manipulate sampled digital media, such as television or teleconferencing. These activities must process and transport media samples within well-defined timeliness requirements. Their integration into the workstation environment requires that the operating system manage resources to meet their time constraints, while at the same time supporting the interactive and batch activities found in conventional applications today. As continuous media activities alone can consume the resources of an entire machine, the operating system must manage resources effectively not just when the system is underloaded, but more importantly, when the system is overloaded. (When all time constraints can be satisfied, the system is said to be underloaded, otherwise it is overloaded.) In particular, as processor cycles are often the most oversubscribed resource, effective processor scheduling is of paramount importance.

Anticipating that processor scheduling based on traditional timesharing would not be suitable for the support of multimedia applications, attempts have been made to adapt real-time schedulers [6][8][12] to support the timeliness requirements of multimedia applications [4][9]. To allow the co-existence of continuous media, interactive, and batch activities, they rely on artificial rate or deadline parameters to force-fit interactive and batch activities into an unsuitable real-time model. The result is that conventional applications are unable to share resources properly without hand-tuning artificial rates and deadlines through trial-and-error for each mix of activities. When the system is overloaded, the drawbacks of these schemes are even

more egregious. At best, they load shed by relying on unique user-specified importances for all activities, and ignore the common case when many activities are of unspecified and indistinguishable importance from one another. At worst, they rely on first-come-first-serve admission control and deliver low utilization, while compelling users to hand-tune their specifications through trial-and-error to fit within their static reserve abstractions.

Because of the difficulty of scheduling conventional interactive and batch activities together with real-time continuous media activities, proposed solutions with actual implementations have predominantly been static two-level schemes [1][3][5]. By allowing multiple policies on top of a base-level mechanism [7], they attempt to avoid the integration problem by supporting separate conventional and real-time scheduling policies. Their drawback is that no matter how sophisticated the layered scheduling policies or how general the base-level mechanism may be [13], scheduling among computations in different policies is limited by the premature loss of information in the static mapping to the base-level mechanism. In particular, none of these schemes can account for timeliness requirements among activities in separate policies. At best, the static isolation of real-time continuous media activities limits the utilization of the system and artificially constrains the range of behavior the system can provide. At worst, such hybrid schemes lead to experimentally demonstrated unacceptable behavior, allowing runaway real-time activities to cause basic system services to lock up, and the user to lose control over the machine [10]. All of these schemes lack the desired degree of control for the user, and lack the ability to provide integrated support and effective overload management for multimedia applications.

## **2 An Overview of our Scheduler**

We have created a new processor scheduler that provides integrated support and effective overload management for all classes of computational activities, whether real-time or conventional, such as those found in multimedia applications. When used to schedule real-time applications, our unified scheduler has the desirable behavior of a typical real-time scheduler: it delivers optimal performance by satisfying the specified deadlines whenever possible. When used to schedule conventional applications, our scheduler has the desirable behavior of a conventional scheduler: it provides good system responsiveness for interactive activities with steady forward progress for batch activities. More importantly, not only does our unified scheduler handle each type of activity effectively, it also handles the combination of both types of tasks seamlessly, without requiring any user parameters.

To support both real-time and conventional tasks, the key problem that must be addressed is how to allocate processing resources in overload. Our solution to this problem is based on fairness. All tasks are given a fair allocation of the processor. Real-time tasks are given their fair allocation first and executed in earliest-deadline-order to meet their deadlines. Any task that cannot meet its deadline within its allocation of processor time is notified that it will miss its deadline and is shed by the scheduler. After the real-time tasks have run, conventional tasks are given the

remaining time to run using a round-robin discipline. Since the real-time tasks receive no more than their fair allocation of the processor, conventional tasks are assured that the remaining time enables them to run for their fair allocation of processor time as well.

Unlike previous real-time approaches which confuse the notions of urgency and importance, our scheduler does not give real-time tasks priority over conventional tasks simply because they have well-defined deadlines. Among tasks of unspecified and indistinguishable importance, all tasks should have equal rights to the processor. A more urgent real-time task should run earlier, but it should not run for more than its fair share of the processor to the starvation of conventional tasks; conventional tasks are expected to make reasonable forward progress as well. Unlike timesharing or static two-level schemes, the scheduler actively seeks to meet the deadlines of real-time tasks by giving them their fair allocation before the conventional tasks. Our combination of fairness-based allocation with deadline-aware scheduling provides a consistent default policy for sharing processor resources among all classes of computations.

As different users may have different preferences as to how processing resources should be shared for a given mix of applications, the scheduler provides simple controls to allow users to bias the processor allocation away from fairness in accordance with user preferences. The controls are used for two kinds of sharing policies: a traditional priority-based policy where important tasks, which can be either real-time or conventional jobs, can monopolize the resources, and a weighted-share policy where different types of activities can obtain a portion of the machine in proportion to their weighting. By incorporating user preferences when the user chooses to specify the information, the system can bias the allocation of resources to maximize the value it delivers to the user. The scheduler is unique in the degree to which it allows users control over the dynamic sharing of processing resources, and yet does not impose any draconian demands on the user to provide information he does not have or does not choose to specify.

Due to space constraints, the details of our scheduling algorithm and performance measurements based on our scheduler implementation in the Solaris operating system [2] are not presented here; they can be found in [11].

### **3 Conclusions**

This paper introduces a novel solution to the difficult problem of scheduling multimedia applications, which have a mix of activities that have very different expected performance characteristics and resource requirements. Our solution handles mixes of conventional interactive and batch activities and real-time continuous media activities in a unified and tightly integrated manner, even when the system is overloaded. Unlike previous approaches, it does not rely on hand-tuning artificial rate or deadline parameters for conventional activities, it does not require specifying importances for all activities, nor does it limit system behavior or utilization as in static two-level schemes. Instead, our scheduler provides all activities with their fair allo-

cation of resources toward satisfying their time constraints without the need for any user parameters. As different users may have different preferences for the behavior of a mix of applications, simple controls are provided that allow the user a high degree of predictable control over the dynamic sharing of processing resources.

## 4 Acknowledgments

We thank J. Duane Northcutt and James G. Hanko for many enlightening discussions. This work was supported in part by an NSF Young Investigator Award and Sun Microsystems Laboratories.

## 5 References

1. AT&T: UNIX System V Release 4 Internals Student Guide, Vol. I, Unit 2.4.2., AT&T, 1990.
2. J. R. Eykholt, S. R. Kleiman, S. Barton, R. Faulkner, et. al.: *Beyond Multiprocessing...Multithreading the SunOS Kernel*, USENIX Summer 1992, San Antonio, Texas.
3. D. B. Golub: *Operating System Support for Coexistence of Real-Time and Conventional Scheduling*, Technical Report CMU-CS-94-212, School of Computer Science, Carnegie Mellon University, November 1994.
4. J. G. Hanko, E. M. Kuerner, J. D. Northcutt, G. A. Wall: *Workstation Support for Time-Critical Applications*, Proceedings of the Second International Workshop on Network and Operating Systems Support for Digital Audio and Video, November 1991.
5. J. G. Hanko: *A New Framework for Processor Scheduling in UNIX*, Abstract talk from the Fourth International Workshop on Network and Operating Systems Support for Digital Audio and Video, November 1993.
6. J. P. Lehoczky, L. Sha, J. K. Strosnider: *Enhanced Aperiodic Responsiveness in Hard Real-Time Environments*, Proceedings of the IEEE Real-Time Systems Symposium, December 1987.
7. R. Levin, E. Cohen, W. Corwin, F. Pollack, W. Wulf: *Policy/Mechanism Separation in Hydra*, Proceedings Fifth Symposium on Operating Systems Principles, ACM, November, 1975.
8. C. D. Locke: *Best-Effort Decision Making for Real-Time Scheduling*, Ph.D. Thesis, Department of Computer Science, Carnegie Mellon University, May, 1986.
9. C. W. Mercer, S. Savage, H. Tokuda: *Processor Capacity Reserves: Operating System Support for Multimedia Applications*, Proceedings of the IEEE International Conference on Multimedia Computing and Systems, May 1994.
10. J. Nieh, J. G. Hanko, J. D. Northcutt, G. A. Wall: *SVR4 UNIX Scheduler Unacceptable for Multimedia Applications*, Proceedings of the Fourth International Workshop on Network and Operating Systems Support for Digital Audio and Video, November 1993.
11. J. Nieh, M. S. Lam, J. G. Hanko, J. D. Northcutt: *Integrated Processor Scheduling in Support of Multimedia Applications*, submitted for publication.
12. S. Ramos-Thuel, J. P. Lehoczky, *On-Line Scheduling of Hard Deadline Aperiodic Tasks in Fixed-Priority Systems*, Proceedings of the IEEE Real-Time Systems Symposium, December 1993.
13. M. Ruschitzka, R. S. Fabry: *A Unified Approach to Scheduling*, Communications of the ACM, July 1977.