Multimedia Applications Require Adaptive CPU Scheduling*

Veronica Baiceanu, Crispin Cowan, Dylan McNamee, Calton Pu, and Jonathan Walpole
Department of Computer Science and Engineering
Oregon Graduate Institute of Science and Technology
Portland, Oregon, USA
{*synthetix-request*}*@cse.ogi.edu*

November 20, 1996

Abstract

CPU scheduling and admission testing for multimedia applications have been extensively studied, and various solutions have been proposed using assorted simplifying assumptions. However, we believe that the complexity and dynamic behavior of multimedia applications and systems make static solutions hard to apply in real-world situations. We are analyzing the difficulties that arise when applying the rate-monotonic (RM) scheduling algorithm and the corresponding admission tests for CPU management, in the context of real multimedia applications running on real systems. RM requires statically predictable, periodic workloads, and while multimedia applications appear to be periodic, in practice they exhibit numerous variabilities in workload. Our study suggests the need for more adaptive scheduling mechanisms, which would allow complex applications to dynamically respond to variations in workload and resource availability. Furthermore, we believe there is a need for a more abstract characterization of applications and resources for admission testing purposes. We conclude that adaptive CPU scheduling policies should address the needs of CPU scheduling and reservation for current multimedia applications.

1 Introduction

Rate-monotonic (RM) scheduling [5, 6] and its corresponding admission tests are widely used techniques in CPU management for multi-media applications [7, 9]. However, implementing RM and admission testing in real systems is problematic due to insufficient operating system support and the inherent difficulty of characterizing variable media and complex applications. RM scheduling provides real-time guarantees *provided* that tasks are perfectly periodic, independent of each other, have constant computation times, and that the OS is preemptive. We have found that these requirements are difficult to meet, particularly the requirement to accurately characterize the application's workload. Relaxing some of these conditions alleviates some of the problems, but the application

*This project is supported in part by grants from DARPA and the National Science Foundation, and donations from Tektronix, Hewlett-Packard and the Portland Trail Blazers.

is still hard to characterize, and the resultant performance achieved by the system will be below its capacity.

Our study considers the client side of a multiple stream distributed Internet MPEG player [2, 3, 4], running on a UNIX system, and allowing interactive user specification of desired QoS (resolution, frame rate, window size). We believe that this type of multimedia application and environment encompasses present and future multimedia systems, and thus the problems we have identified are characteristic to general multimedia environments.

A prominent strategy for performing resource management for multimedia systems is QoS-driven management, in which quality requirements such as resolution and frame rate are translated into resource requirements such as computation burst frequencies and durations. This resource information is then used for admission testing and resource reservation. The motivation for this translation from application level requirements to resource requirements is to enable admission testing for a certain QoS.

However, there are several difficulties with this approach. We have discovered that the highly dynamic and complex nature of multimedia systems makes accurate static prediction of the behavior of the system difficult. Our experience indicates the need for more adaptive mechanisms that can react to changes in the resource needs of multimedia applications to ensure that they get the resources necessary to provide the requested QoS, and that they do not consume excess resources. Such mechanisms allow a less precise characterization of applications and systems, and more relaxed, higher level admission tests, providing higher resource utilization. While we consider only RM scheduling, we believe that the problems identified are common to all *static* CPU management strategies that try to statically characterize a given multimedia system, and attempt to reserve resources based on this static analysis.

This paper is structured as follows: Section 2 describes the environment where we have studied the applicability of RM scheduling and admission testing to multimedia systems. Section 3 outlines the multiple difficulties we have noticed when trying to apply RM scheduling and static admission test to our multimedia system. Finally, Section 4 describes our conclusions and suggests some future directions in multimedia resource management.

2 Rate-Monotonic Scheduling and Multimedia System

We have assessed the applicability of the RM scheduling algorithm and admission testing in real-world systems by considering the case of our MPEG player [2, 3, 4]. To make our study possible, we have created a simplified model of our player, as represented in figure 1.

The relevant processes are the *decoding process* and the *display process*. Buf 1 holds frames assembled from network packets. The decoding process reads compressed frames from Buf 1, placing the decoded frames in Buf 2, which are subsequently displayed by the display process.

Our study simulates such streams by implementing tasks with CPU service times derived from the real computation times of the decoder and display processes in the player, and service frequencies

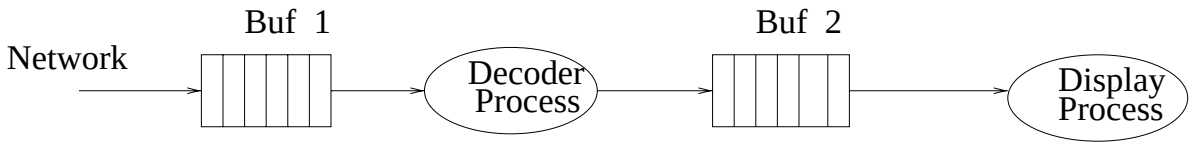

Figure 1: Simplified Architecture of One Stream at MPEG Player Client Site

corresponding to typical video frame rates. We then execute the the simulated tasks on a real system and measure their behavior.

We approximate RM scheduling by assigning priorities to tasks according to their periods, using UNIX real-time (non-aging) priorities. We use Lehoczky et al's exact admission test [5].

We did not study relaxed deadline constraints for the decoding process, even though buffering allowed it: we just assigned the same frame rates to both the decoder and display processes. We consider a stream to have missed its deadline if either its decoder or display processes misses a deadline. This simplifies analysis, but is a pessimistic assumption due to buffering. Variable decode times for MPEG frame types is also not modeled; rather the tasks have workloads corresponding to the average decode times in the player for three different resolutions.

3 Problems Encountered with RM

We encountered several problems in applying RM scheduling to guarantee Multimedia QoS. Sections 3.1 through 3.3 explain how multimedia applications fail to meet RM's preconditions, and so we made simplifying assumptions about the application. Despite these simplifying assumptions, our scheduling discipline is still only an approximation to RM, as explained in Section 3.4. Thus our attempt to use RM scheduling and admission testing highlights the difficulty in applying RM to real multimedia systems. This section describes these problems in detail, along with other features that complicate system characterization and management using RM.

3.1 Complexity of Multimedia Application Architectures

We have only considered a single, simplified pipeline from our multimedia player. Our prototype MPEG player supports multiple video pipelines, possibly synchronized with each other, and with one or multiple audio pipelines [4]. The process structure is also simplified – in a real player, the display process might be broken into two different processes, one for dithering and image scaling, and another one for rendering the image on the display [4]. A "buffering" process should also be considered, which would assemble packets arriving over the network into frames, and store them into Buf 1. Buffer disposition and buffer sizes could also vary depending on design preferences [4]. Controller processes might synchronize some of the streams or perform other functions.

Thus, the overall application architecture can be quite complex and dynamic. Analyzing the behavior of tasks in this context is difficult, because it is likely that many tasks do not conform to the periodic model, and may exhibit critical dependencies. Even ignoring synchronization among streams, tasks within a pipeline clearly depend on each other's results. While deeper analysis could allow admission testing of inter-dependent tasks, a static analysis would result in over-reserving resources. Furthermore, the characterization of the tasks (which is required for RM to work) will change with each platform, depending on local CPU speed and memory availability, virtually eliminating device independence in the multimedia application.

3.2 Variation in Multimedia Data Streams

Our player delivers MPEG compressed data across the Internet. The three kinds of MPEG encoded frames (I, P, B) vary significantly in size and also in necessary software decoding time. Even for frames of the same type, image content causes significant irregularity in frame size.

There are numerous admission testing issues related to the irregularity of MPEG decoding times. A pessimistic admission tester will consider the worst-case decode time for all the frames. However, this will significantly over-reserve the CPU. Rather, the period for the decoding process should be relaxed in proportion to the amount of buffer space between the decoding and the displaying processes, and before the decoding process. However, this implies knowing the exact amount of buffering that would compensate for a certain variation in frame decoding time. The variation itself is hard to determine, because it depends on image content, MPEG encoding parameters, and variation in the client's CPU speed.

3.3 Complexity of Resource Interplay

We have found that CPU management cannot be isolated from memory and network management. Network burstiness can result in unpredictable variation of CPU workload, so static CPU management strategies could not guarantee performance. Buffering relaxes the deadline constraints of the tasks, and compensates for the variability in task computation times and task arrival times. Thus admission testing without considering buffering effects will dramatically over-reserve the CPU.

In fact, buffering and CPU management can interact in surprising ways. For instance, if there are multiple concurrent pipelines such as in Figure 1, the RM disciple of giving a higher priority to processes with higher frame rate will result in decoder processes running continually until they blocks on Buf 1 empty or Buf 2 full. This will starve a lower priority pipeline, unless the amount of buffering for that pipeline is sufficiently larger than the one in the higher priority pipeline.[1] Thus, buffering for some streams can result in *degrading* the performance of other streams. Whether a more complex buffer management strategy is appropriate or not, this example clearly shows that CPU management cannot ignore buffer management.

[1] Assuming that processes in both pipelines have the same CPU service times.

3.4 Insufficient Operating System Support for Multimedia

Our previous work [1] analyzed the effects of various UNIX features on RM scheduling and admission testing. We used the simulation of player streams described in Section 2, and assigned priorities to tasks according to their periods to approximate RM on top of UNIX. Implementing RM on top of UNIX introduces imprecision in the scheduling mechanism, as RM (in its original form) requires preemptive scheduling, while on UNIX preemption is not possible at granularity lower than time-slice length (usually 10 ms). Additional imprecision is introduced through context switch overhead, timer interrupt imprecision, non-preemptability in system calls and during I/O blocks.

We used Lehoczky's RM admission test [5], which assumes perfect preemptability of tasks. In this sense, our test is optimistic: Some of the tasks it accepted should have been rejected, even ignoring the influence of non-real-time UNIX characteristics. Note that we could have used an admission test that would not have required perfectly preemptable tasks. For instance, we could have used the more restrictive admission test described by Nagarajan and Vogt [8] for nonpreemptable tasks, but this would over-reserve the CPU, and complicate analysis.

Our study showed that the UNIX features that generate imprecision for our RM scheduling mechanism and admission test does not generally cause significant performance degradation compared to RM applied in ideal conditions [1]. However, when the CPU utilization factor is close to 100%, this slight imprecision can lead to catastrophic degradation of quality for the stream with the lowest priority in the system. Thus, the imprecision introduced by the operating system contributes to the overall unpredictable behavior of multimedia systems.

3.5 Potentially Frequent User Demand for QoS Changes

Multimedia applications should allow users to interactively specify QoS requirements in different dimensions. Our MPEG player currently allows users to specify the desired frame rate or resolution for any of its multiple streams, and to modify the window size by simply stretching the window as the video plays [4]. This kind of interface suggests a potentially frequent demand for admission testing, together with the need of translating each of the "points" in this QoS space into corresponding requirements in the resource space.

The complexity of both the QoS space and the resource space suggests that perfect characterization is hard to achieve, so it would be desirable to have an adaptive mechanism that would obviate accurate characterization. The same adaptive mechanism could then detect and adapt to changes in user QoS requirements.

4 Conclusions

Our study of RM applied to multimedia systems has highlighted the difficulty of this approach. While our analysis concentrates on the CPU resource, and the RM mechanism in particular, we

believe that these problems are inherent to all mechanisms that attempt to statically characterize multimedia systems, and to manage future resources based on this characterization.

Rather than propose enhancements to RM or other static mechanisms for scheduling and admission testing, we believe that the problems we have encountered demonstrate a need for more adaptive mechanisms, possibly derived from the mechanisms that perform static resource management, and that consider the interplay among heterogeneous resources in the system. Such mechanisms should independently strive to achieve the desired QoS, in an environment with variable resources, as well as complex and variable application demands. These mechanisms need to be applicable with only approximate knowledge of the application and of the resources available. Such a facility would enable admission testing with only with an abstract characterization of resource demand and resource availability.

References

[1] Veronica Baiceanu. CPU Management for UNIX-based MPEG Video Applications. Available at `http://www.cse.ogi.edu/ baiceanu/papers/`, 1996.

[2] Shanwei Cen, Calton Pu, Richard Staehli, Crispin Cowan, and Jonathan Walpole. A Distributed Real-Time MPEG Video Audio Player. In *Proceedings of the 1995 International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, pages 151–162, New Hampshire, April 1995.

[3] Crispin Cowan, Shanwei Cen, Jonathan Walpole, and Calton Pu. Adaptive Methods for Distributed Video Presentation. *ACM Computing Surveys*, 27(4):580–583, December 1995. Symposium on Multimedia.

[4] Rainer Koster. Design of a Multimedia Player with Advanced QoS Control. Master's thesis, Oregon Graduate Institute of Science & Technology, 1997.

[5] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Proc. IEEE 10th Real-Time Systems Symp.*, pages 166–171, December 1989.

[6] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM*, 20(1):46–61, January 1973.

[7] C. W. Mercer, S. Savage, and H. Tokuda. Processor Capacity Reserves: Operating System Support for Multimedia Applications. In *Proc. of the International Conference on Multimedia Computing and Systems*, pages 90–99, May 1994.

[8] R. Nagarajan and C. Vogt. Performance of Multimedia Traffic over the Token Ring. Tech. report, IBM-ENC, Heidelberg, 1992.

[9] K.K. Ramakrishnan, Lev Vaitzblit, Cary Gray, Uresh Vahalia, Dennis Ting, Percy Tzelnic, Steve Glaser, and Wayne Duso. Operating Systems Support for a Video-On-Demand File Service. In *Proceedings of the 1993 International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'93)*, pages 225–236, Lancaster, U.K., November 1993.