The Real-Time Execution Performance Agent
An Approach for Balancing Hard and Soft Real-Time Execution for Space Applications

Sam Siewert, Gary Nutt, and Elaine Hansen
Department of Computer Science
University of Colorado, Boulder, CO 80309-0430
siewerts@rodin.colorado.edu

ABSTRACT

Use of AI (Artificial Intelligence) algorithms such as adaptive planners, intelligent monitors, and data miners can help optimize overall return from space systems by providing adaptive operations that can exploit opportunities. Typically, space systems involve many hard real-time functions including: attitude, thermal, propulsion, and mechanism control; detector/sensor data stream processing; telemetry gathering and packetization; command handling; and many other periodic tasks which must be executed such that processing is completed by a periodic deadline. While there has been a concerted effort to design AI algorithms to have predictable execution requirements (e.g. anytime algorithms), most of these applications are relegated to running in a best effort fashion using slack time left over from the hard real-time periodic tasks which must be given higher priority to ensure safety and control. The problem with executing the AI algorithms in slack time is that this makes their execution performance impossible to predict. The alternative of requiring AI algorithms to be anytime algorithms so that they can be treated like a hard real-time task with a deterministic minimum response time can be prohibitive since such algorithms are hard to design and the minimum response may not provide much of an optimization. This paper describes an third alternative which provides an intelligent execution control mechanism, the EPA (Execution Performance Agent), that ensures execution of algorithms based on required reliability and confidence in meeting deadlines rather than priorities. The EPA provides predictable and safe execution of hard real-time safety critical and soft real-time mission optimizing tasks. By analogy, the EPA provides a balancing capability much like the everyday ability people have to walk without tripping while contemplating how to build a better career. It does this by executing tasks in specific execution reliability and confidence space and monitoring actual execution times to determine when resources must be adjusted. The EPA is currently being evaluated in a digital control and continuous video media testbed at the University of Colorado. Based upon testbed results, the EPA is also being considered for execution control of real-time operating system tasks including AI and digital control applications on a small spacecraft, Citizen Explorer, being built by the Colorado Space Grant College. The EPA was inspired by experience with a Space Grant Space Shuttle small payload which included control of three instruments and optimization of their operations using an adaptive planner and an intelligent monitoring system from the NASA Jet Propulsion Laboratory. The requirements for both hard real-time tasks and the use of AI applications on Citizen Explorer will be more demanding, and it is hoped the EPA can be shown to increase reliability and predictability of such systems. Details of the EPA mathematical formulation, the testbed implementation, performance results, and results of the analysis to determine if the EPA meets the Citizen Explorer requirements will be discussed in the paper.

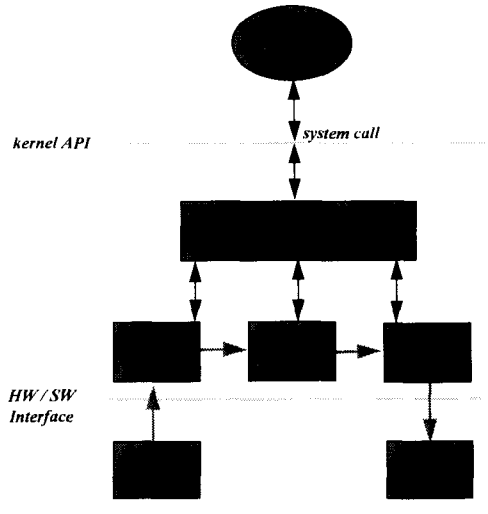
1.0 Introduction

Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space, 1-3 June 1999 (ESA SP-440)

The RTEPA ("Real-Time Execution Performance Agent") mechanism introduced in this paper is intended to provide time-critical applications with quantifiable assurance of system response using a simple EPA ("Execution-Performance Agent") interface to the deadline monotonic scheduling algorithm. In addition, the RTEPA provides a system call and signal interface which allows applications to monitor and control pipeline real-time performance on-line, and therefore significantly extends existing work on "in-kernel" pipelines. The set of applications requiring this type of performance negotiation support from an operating system is increasing with the emergence of virtual reality environments [Nu95], continuous media [Co94], multimedia [Ste95], digital control, and "shared-control" automation [Bru93][SiNu96]. The RTEPA mechanism is being implemented in the VxWorks microkernel, and is being tested in a rail-guided air-powered vehicle testbed incorporating continuous media, digital control, and "shared-control" pipelines. Likewise, the RTEPA is being tested with a 5 DOF robot arm that provides basic pick-and-place capabilities.

Traditionally, if an application requires service time assurances, there are three approaches: best-effort systems, hard real-time systems, and application specific embedded systems. Best-effort systems rely upon adequate resources always being available whenever an arbitrary task requests service, and can make no guarantees when they are even temporarily overloaded. Hard real-time systems require that the application provide resource bounds (e.g., the "Worst-Case Execution Time" or WCET) so that the operating system can mathematically check schedulability and admit only tasks whose complete execution can be guaranteed by hard deadlines. Embedded systems typically include cooperative tasks implemented in a single protection domain. Each task is designed with full knowledge of all other tasks and resource demands; it is difficult to change or scale embedded software. These three approaches do not provide controllable real-time reliability or ability to make on-line tradeoffs.

Figure 1: In-Kernel Pipe with Filter Stage and Device Interface Modules



In contrast, the RTEPA mechanism supports a broad spectrum of contemporary applications ranging from virtual environments to semi-autonomous systems [Si96]. The RTEPA facility allows an application developer to construct a set of real-time kernel modules that manage an input (source) device; apply simple processing stages on the input stream (pipeline stage filters); control individual processing stage behavior through parameters obtained from a user-space application; provide performance feedback to the controlling application; and manage the output (sink) device. This basic "in-kernel" pipeline design is very similar to the *splice* mechanism [Fal94], but the EPA and scheduling control are much different. Each RTEPA module, shown in Figure 1, is implemented as a kernel thread configured and controlled through the EPA and scheduled by the DM ("Deadline Monotonic") algorithm. The controlling application executes as a normal user thread. The RTEPA mechanism is efficient due to removal of overhead associated with protection domain crossings between device and

processing buffers, and reliable due to kernel thread scheduling (compared to split-level scheduling of user threads). The RTEPA interface provides configuration and execution flexibility on-line, with performance-oriented "reliable" execution (in terms of expected number of missed soft deadlines and missed termination deadlines).

The EPA interface is intended to allow an application to specify desired service and adjust performance for both periodic pipelines requiring isochrony and aperiodic pipeline execution. Many scenarios exist for on-line RTEPA service renegotiation for continuous media, digital control, etc. [Si96]. For example, a continuous media application might initially negotiate reliable service for a video pipeline with a frame-rate of 30 fps, and later renegotiate on-line for 15 fps so that an audio pipeline may also be executed. An application loading pipeline stages must specify the following parameters for a service epoch:

1) Service type common to all modules in a single pipeline; ***<guaranteed, reliable, or best-effort>***

i) Computation time type; ***<$C_{worst\text{-}case}$ for guaranteed, $C_{expected}$ for reliable, or none for best-effort>***

ii) Off-line execution samples for ***$C_{expected}$;<{Sample-array}, [distribution-free or (normal, σ, $C_{expected}$)]>***

2) Input source or device interface designation (source must exist as stage or device interface); ***<source>***

3) Input and output block sizes; ***<S_{in}, S_{out}>***

The application must also provide and can control these additional parameters on-line during a service epoch:

5) Desired termination and soft deadlines with confidence for ***reliable***; ***<D_{term}, D_{soft}, term-conf, soft-conf>***

6) Minimum and optimal time for output response (earlier responses are held by EPA); ***<R_{min}, R_{opt}>***

7) Release period (expected minimum interarrival time for aperiodics) and I/O periods; ***<T, Tin, Tout>***

The approach for scheduling RTEPA thread execution is based on the EPA interface to the fixed priority DM scheduling policy and admission test called the EPA-DM approach here. The EPA-DM approach supports reliable soft deadlines given pipeline stage execution times in terms of an execution time confidence interval instead of deterministic WCET. Also noteworthy, the RTEPA facility uses two protection domains; one for user code and one for operating systems code. However, the RTEPA facility allows "untrusted" code to be executed in the kernel protection domain. We have focused on the functionality of architecture, relying on the existence of other technology such as that used in the "SPIN" operating system [Be95] to provide compile time safety checking. The negotiative control provided by RTEPA is envisioned to support isochronous and event-driven applications which can employ and control these pipelines for guaranteed or reliable execution performance.

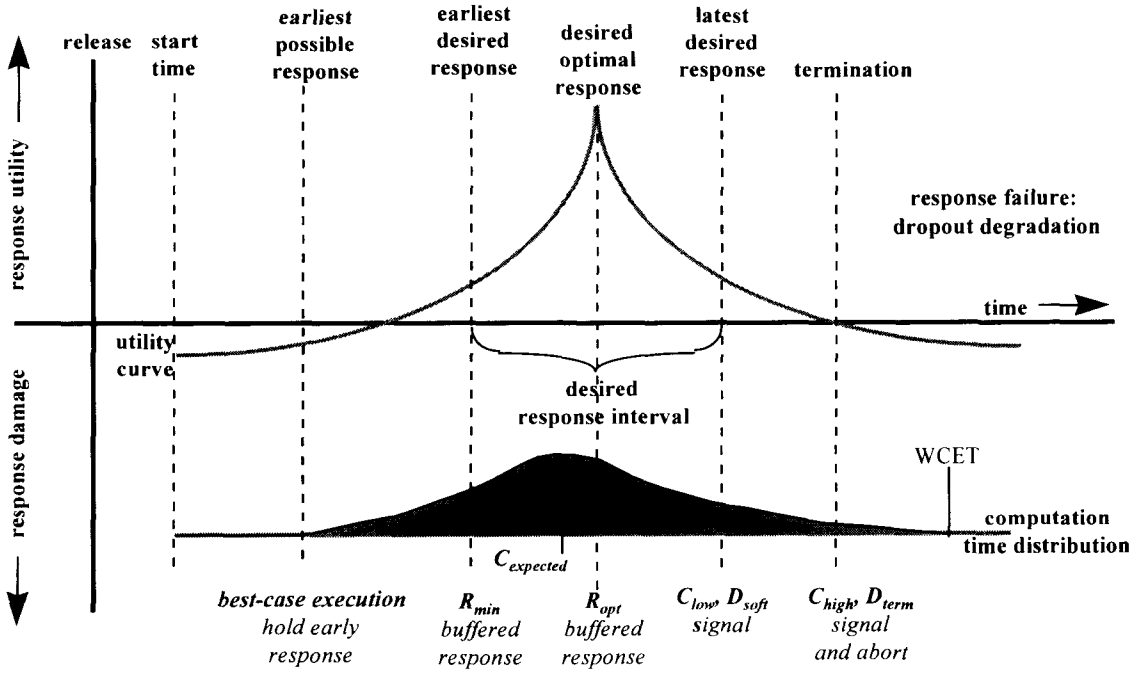
2.0 EPA-DM Approach to Thread Scheduling

The concept of EPA-DM thread scheduling for pipeline stages is based upon a definition of soft and termination deadlines in terms of utility and potential damage to the system controlled by the application [Bu91]. The concept is best understood by examining Figure 2, which shows response time utility and damage in relation to soft and termination deadlines as well as early responses. In this design, the EPA will signal the controlling application when either deadline is missed, and specifically will abort any thread not completed by its termination deadline. Likewise, the EPA will buffer early responses for later release at R_{opt}, or at R_{min} worst case. The EPA allows execution beyond the soft deadline. Signaled controlling applications can handle deadline misses according to specific performance goals, using the

EPA interface for renegotiation of service. For applications where missed termination deadline damage is catastrophic (i.e. termination deadline is a "hard deadline"), the pipeline must be configured for guaranteed service rather than reliable service.

The well established DM scheduling policy and schedulability test are used due to their ability to handle execution where deadline does not equal period [Au93]. This may often be true for the applications to be supported. One major drawback of the DM scheduling policy is that to provide a guarantee, the WCET of each pipeline stage thread must be known along with the release period. Otherwise, for performance-oriented applications -- where occasional soft and termination deadline failures are not catastrophic, but simply result in degraded performance -- the "reliable" option with quantifiable assurance is provided, given expected execution time. Despite the ability to opt for no guarantee, this mechanism does not just provide "best effort" execution. Instead, a compromise is provided based on the concept of execution time confidence intervals and the EPA interface to the DM scheduler. An example of the EPA-DM approach is given here with a simple two-thread scenario preceded by a review of the goals for the EPA-DM approach.

Figure 2: Execution Events and Desired Response Showing Utility



The EPA-DM schedulability test eases restriction on the DM admission requirements to allow threads to be admitted with only expected execution times (in terms of an execution confidence interval), rather than requiring deterministic WCET. The expected time is based on off-line determination of the execution time confidence interval. Knowledge of expected time can be refined on-line by the EPA each time a thread is run. By easing restriction on the WCET admission requirement, more complex processing can be incorporated, and pessimistic

WCET with conservative assumptions (e.g. cache misses and pipeline stalls) need not reduce utility of performance-oriented pipelines which can tolerate occasional missed deadlines (especially with probability of misses).

With this approach, the DM schedulability tests, which consider computation time and interference for a thread set, can still be used by the EPA as stages are loaded. Basic DM scheduling formulas are extended to return expected number of missed soft and termination deadlines to the controlling application. For this capability, when a module is loaded, the computation time must be provided with a sufficient sample set for distribution-free confidence estimates, or an assumed distribution and a smaller sample set of execution times measured off-line. From this, the computation time used in the schedulability tests is computed based upon desired confidence for meeting soft and termination deadlines. All interfering threads are pessimistically assumed to run to their termination deadline where they either will have completed or are aborted. For example, for thread i, let C(i) = expected execution time; $D_{soft}(i)$ = soft deadline; $D_{term}(i)$ = termination deadline; and T(i) = period; with the DM condition that C(i) <= $D_{soft}(i)$ <= $D_{term}(i)$ <= T(i). The worst-case confidence interval execution times $C(i)_{low}$ and $C(i)_{high}$ used in the extended DM schedulability tests below are based on desired confidence in execution time and probability of late response. In cases where the actual execution time is greater than the worst-case confidence interval execution time, deadlines will be missed. The expected number of missed deadlines will be less-than or equal to expected execution times outside the confidence interval resulting in response beyond a given deadline. So, if a thread has an execution time confidence of 0.999 and passes the admission test, then it is expected to miss its associated deadline 0.1% of the time or less.

For example, consider two threads that have a normal distribution of execution times (the normal distribution assumption is not required, but greatly reduces the number of off-line samples needed compared to assuming no distribution), so that unit normal distribution quantiles Zp_{low} and Zp_{high} can be used, and assume that WCET(i) is known for comparison, so that we have:

thread i=1: $C_{expected}(1)$=40, $\sigma(1)$=15, $N_{trials}(1)$=32, $Zp_{low}(1)$=3.29 for soft-conf=99.9%, $Zp_{high}(1)$=3.72 for term-conf=99.98%, WCET(1)=58, $D_{soft}(1)$=50, $D_{term}(1)$=60, and T(1)=250

thread i=2: $C_{expected}(2)$=230, $\sigma(2)$=50, $N_{trials}(2)$=32, $Zp_{low}(2)$=1.96 for soft-conf=95%, $Zp_{high}(2)$=3.72 for term-conf=99.98%, WCET(2)=310, $D_{soft}(2)$=400, $D_{term}(2)$=420, and T(2)=500

If these threads can be scheduled based on the EPA inputs to the admission test, then thread one has a probability of completing execution before D_{soft} of at least 99.9% expressed $P(C_{low} < D_{soft}) \geq 0.999$. Similarly, probability $P(C_{high} < D_{term}) \geq 0.9998$. Likewise thread two has respective deadline confidences $P(C_{low} < D_{soft}) \geq 0.95$ and $P(C_{high} < D_{term}) \geq 0.9998$. Based on sufficient, but not necessary schedulability tests for DM [Au93] with EPA execution time confidence intervals inputs rather than just worst-case execution time, the schedulability with desired confidence in deadlines can be derived from the execution time confidence intervals, as shown below.

From execution time confidence intervals and sufficient (but not necessary) DM schedulability test:

eq 1: From probability theory for a normal distribution, $C_{low\ or\ high}(i) = C_{expected}(i) + Zp_{low\ or\ high}(i)\left(\frac{\sigma(i)}{\sqrt{N_{trials}(i)}}\right)$

eq 2: EPA-DM admission test: $\forall i: 1 \leq i \leq n$: $\left(\frac{C_{low\ or\ high}(i)}{D_{soft\ or\ term}(i)}\right) + \left(\frac{I_{max}(i)}{D_{soft\ or\ term}(i)}\right) \leq 1.0$?

eq 3: $I_{max}(i) = \sum_{j=1}^{i-1} \text{ceiling}\left(\frac{D_{term}(i)}{T(j)}\right) D_{term}(j)$; where

$I_{max}(i)$ is the interference time by higher priority

threads j=1 to i-1 which preempt and run up to the "ceiling term" number of times during the period in which thread i runs.

Can thread i=1 be scheduled given execution time confidence and desired D_{soft} and D_{term} confidence? Yes

using eq 1: $C_{high}(1) = 40 + Zp_{high}(1)\left(\frac{15}{\sqrt{32}}\right) =$ 49.86; and likewise $C_{low}(1) = 48.72$

using eq 2&3: $\left(\frac{48.72}{50}\right) \le 1.0$ and $\left(\frac{49.86}{60}\right) \le 1.0$ for $C_{low}(1)$and $C_{high}(1)$; likewise $\left(\frac{58}{60}\right) \le 1.0$ for WCET

C_{low}, C_{high} can be scheduled. (*note: highest priority thread has no interference, so $I_{max}(i)=0$*)

Can thread i=2 be scheduled given execution time confidence and desired D_{soft} and D_{term} confidence? Yes

using eq 1: $C(2)_{high} = 230 + 3.72\left(\frac{50}{\sqrt{32}}\right) =$ 262.88; and likewise $C(2)_{low} = 247.32$

using eq 2&3: $\left(\frac{C_{low\ or\ high}(2)}{D_{soft\ or\ hard}(2)}\right) + \left(\frac{I_{max}(2)}{D_{soft\ or\ hard}(2)}\right) \le 1.0$?; $I_{max}(2) = \text{ceiling}\left(\frac{D_{term}(2)}{T(1)}\right) D_{term}(1)$

In the worst case, given the abort policy for incomplete threads reaching their termination deadline, maximum interference occurs when all higher priority threads execute until they are aborted by the EPA.

simplifying eq 2&3: $\left(\frac{247.32}{400}\right) + 2\left(\frac{60}{400}\right) \le 1.0$ and $\left(\frac{262.88}{420}\right) + 2\left(\frac{60}{420}\right) \le 1.0$;

simplifying eq 2&3: $\left(\frac{310}{420}\right) + 2\left(\frac{60}{420}\right) \le 1.0$? is FALSE ; **WCET can not be scheduled**

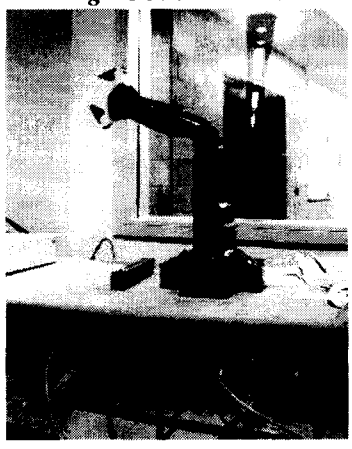
C_{low}, C_{high} can be scheduled. (*note: thread 1 interferes up to its termination deadline twice in this example*)

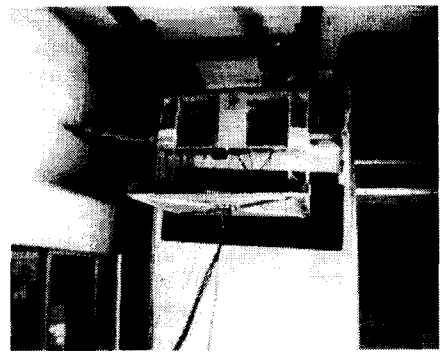
These formulas show that the two threads can be scheduled using non-WCET execution time such that desired performance is achieved. Note that the basic DM formulas show that the thread set is not considered schedulable if only WCET is considered. In this case, WCET, which is a statistical extreme, lead to rejection of a thread set which can be scheduled with ≥ 99.98% probability of successfully meeting termination deadlines.

3.0 In-Work Implementation, Experimentation and Evaluation

The mechanism is being implemented VxWorks with modifications to rate monotonic scheduling of real-time kernel threads to implement the EPA-DM approach. The kernel is also being modified to incorporate the pipeline EPA system call and signal interface with functionality for loading and controlling pipe stage modules and device interface modules. The RACE (Rail-Guided Air-Powered Control Experiment) testbed (Figure 3) has been built using off-the-shelf "68HC11" microcontrollers for sensor and actuator control, with a serial interface to an Intel x86 computer for implementation of the digital control, continuous media, and "shared-control" pipelines. The RACE testbed experiments with the RTEPA mechanism on-board the RACE vehicle use a basic set of device commands (***safe***, ***pitch-motors <angle>***, ***thrust <left|right> <level> <duration>***, ***read compass***, ***read vertical range***, ***read forward range***). These commands can be used in digital control pipelines to implement ramp station keeping and yaw control. The ramp ranging is provide with continuous media video-based ranging from an on-board "QuickCam" output piped to ranging estimation and control algorithm. Likewise, the digital compass output is piped to a yaw estimation and control algorithm.

Figure 3: 5 DOF Robotic Testbed (left); RACE Digital Control Testbed (right)





4.0 Related Work

A number of pipeline mechanisms for continuous media have been developed [Gov91], [Co94], [Fal94]. However, most common implementations include application-level processing with device buffers mapped from kernel space into user-space rather than an "in-kernel" mechanism for executing user code loaded into kernel space. Likewise, these memory-mapped implementations also employ user-level threads with split-level scheduling or bindings of user threads onto kernel threads. The *splice* mechanism is most relevant since it operates "in-kernel" using loadable modules or simple streaming as the RTEPA will, and was shown to have up to a 55% performance improvement [Fal94]. However, to our knowledge, *splice* does not provide a configuration and on-line control interface like the EPA.

Many examples of periodic hard real-time digital control streams exist [Kl94], but no general mechanism for "reliable" real-time control of pipelines is known to exist. Research on process control requirements for digital control indicate that parametric control of a number of kernel pipes within a general operating system environment would be useful for sophisticated industrial applications. Finally, many real-time semi-autonomous and "shared control" projects are in progress [Bru93] [Fle95], including applications where occasional missed deadlines would not be catastrophic [Pa96] [Bro95].

5.0 Conclusion

Experiments will be implemented using both the RTEPA and user-level applications to compare performance. However, the RTEPA is not just expected to improve throughput compared to application-level processing, but is more significantly expected to provide reliable configuration, monitoring, and control of this type of efficient mechanism through its EPA interface to the DM scheduler. A fundamental aspect of the EPA performance control is based on the EPA-DM confidence interval approach for reliable execution. Thus, the EPA will be evaluated in terms of how well pipelines are able to meet expected and desired performance in terms of missed deadlines. Finally, experiments are being evaluated in terms of real-time parameters such as video stream dropouts, latency variation, overshoot and drift to evaluate the reliability afforded by the EPA to applications. These experiments will be run individually and simultaneously to evaluate use of the RTEPA mechanism for complex real-time applications involving multimedia and interaction between users for complex applications such as "shared" control.

6.0 References

[Au93] Audsley, N., Burns, A., and Wellings, A., "Deadline Monotonic Scheduling Theory and Application", Control Engineering Practice, Vol. 1, pp 71-8, 1993.

[Be95] Bershad, B., Fiuczynski, M., Savage, S., Becker, D., et al., "Extensibility, Safety and Performance in the SPIN Operating System", Association for Computing Machinery, SIGOPS '95, Colorado, December 1995.

[Bu91] Burns, A., "Scheduling Hard Real-Time Systems: A Review", Software Engineering Journal, May 1991.

[Bro95] Brooks, R., "Intelligence Without Reason", In Steels, L. and Brooks, R., eds., The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1995.

[Bru93] Brunner, B., Hirzinger, G., Landzettel, K., and Heindl, J., "Multisensory shared autonomy and tele-sensor-programming - key issues in the space robot technology experiment ROTEX", IROS '93 International Conference on Intelligent Robots and Systems, Yokohama, Japan, July, 1993.

[Co94] Coulson, G., Blair, G., and Robin, P., "Micro-kernel Support for Continuous Media in Distributed Systems", Computer Networks and ISDN Systems, pp. 1323-1341, Number 26, 1994.

[Fal94] Fall, K., and Pasquale, J., "Improving Continuous-Media Playback Performance With In-Kernel Data Paths", Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS), pp. 100-109, Boston, MA, June 1994.

[Fle95] Fleischer, S., Rock, S., Lee, M., "Underwater Vehicle Control from a Virtual Environment Interface", Association for Computing Machinery, 1995 Symposium on Interactive 3D Graphics, Monterey CA, 1995.

[Gov91] Govindan, R., and Anderson, D., "Scheduling and IPC Mechanisms for Continuous Media", 13th ACM Symposium on Operating Systems Principles, 1991.

[Kl94] Klein, M., Lehoczky, J., and Rajkumar, R., "Rate-Monotonic Analysis for Real-Time Industrial Computing", IEEE Computer, January 1994.

[Nu95] Nutt, G., Antell, J., Brandt, S., Gantz, C., Griff, A., Mankovich, J., "Software Support for a Virtual Planning Room", Technical Report CU-CS-800-95, Dept. of Computer Science, University of Colorado, Boulder, December 1995.

[Pa96] Paulos, E., and Canny, J., "Delivering Real Reality to the World Wide Web via Telerobotics", IEEE International Conference on Robotics and Automation.

[Si96] Siewert, S., "Operating System Support for Parametric Control of Isochronous and Sporadic Execution in Multiple Time Frames", Ph.D. dissertation proposal, Univ. of Colorado Boulder, 1996.

[SiNu96] Siewert, S., and Nutt, G., "A Space Systems Testbed for Situated Agent Observability and Interaction", In the Second ASCE Specialty Conf. on Robotics for Challenging Environments, Albuquerque, New Mexico, June 1996.

[Ste95] Steinmetz, R., and Wolf, L., "Evaluation of a CPU Scheduling Mechanism for Synchronized Multimedia Streams", in Quantitative Evaluation of Computing and Communication Systems, Beilner, H. and Bause, F. eds., Lecture Notes in Computer Science, No. 977, Springer-Verlag, Berlin, 1995.

[To90] Tokuda, H., Nakajima, T., and Rao P., "Real-Time Mach: Towards a Predictable Real-Time System", Proceedings of USENIX Mach Workshop, October 1990.