Summer June 9, 1997

A Real-Time Execution Performance Agent Interface to Parametrically Controlled In-Kernel Pipelines

Sam B Siewert
Gary Nutt, *University of Colorado Boulder*
Marty Humphrey, *University of Virgina*

A Real-Time Execution Performance Agent Interface to Parametrically Controlled In-Kernel Pipelines

Sam Siewert, Gary Nutt, and Marty Humphrey*
Department of Computer Science, University of Colorado, Boulder, CO 80309-0430

Abstract

This paper presents work-in-progress to build a confidence-based in-kernel pipeline execution performance interface to a fixed priority deadline monotonic scheduler. The interface provides performance-controlled pipeline execution, allowing applications to specify expected execution times, negotiate desired deadline confidence and to configure and control pipelines. The confidence-based scheduling interface and in-kernel pipeline are being evaluated on an unoccupied air vehicle incorporating digital control, continuous media, and event-driven pipelines.

1.0 Introduction

The PCIP ("Parametrically Controlled In-Kernel Pipeline") mechanism introduced in this paper is intended to provide time-critical applications with quantifiable assurance of system response using an EPA ("Execution-Performance Agent") interface to the deadline monotonic scheduling algorithm. The Real-Time EPA/PCIP provides a system call and signal interface allowing applications to monitor and control pipeline performance on-line, and therefore significantly extends existing work on "in-kernel" pipelines. The set of applications requiring this type of performance negotiation support is increasing with the emergence of virtual reality environments [11], continuous media [5], multimedia [16], digital control [18], and "shared-control" automation [4][15]. The RT EPA/PCIP mechanism is being implemented in the RT-Mach microkernel as well as Solaris 2.5.1, and will be tested on a UAV ("Unoccupied Air-Vehicle") testbed incorporating continuous media, digital control, and "shared-control" pipelines.

*Department of Computer Science and Engineering, University of Colorado at Denver, Denver, CO

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.

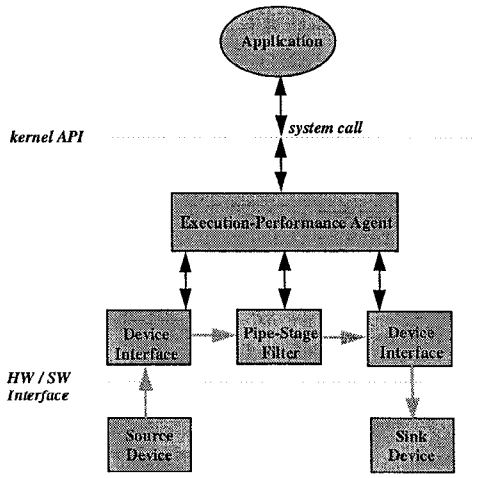

Figure 1: RT EPA/PCIP Kernel Modules

In contrast, the RT EPA/PCIP mechanism supports a broad spectrum of contemporary applications ranging

from virtual environments to semi-autonomous systems [14]. The RT EPA/PCIP 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 with a user-space controlling application; obtain performance feedback; and manage the output (sink) device. It is inefficient and unreliable to implement the types of applications the RT EPA/PCIP is intended to service completely in application space due to protection domain crossing overhead and split-level scheduling complexity. By loading pipeline stages as kernel modules, the RT EPA/PCIP will not require kernel rebuilds for new pipelines. Also noteworthy, the RT EPA/PCIP facility uses two protection domains; one for controlling application user code and one for operating systems and RT EPA/PCIP code. The RT EPA/PCIP does allow "untrusted" module code to be executed in the kernel protection domain. We have focused on RT EPA/PCIP functionality, relying on technology such as that used in "SPIN" [2] to provide module compile time safety checking.

The PCIP design is similar to the *splice* mechanism [6], but the EPA and scheduling control are much different. Each PCIP 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 user thread. The PCIP mechanism, like *splice*, is efficient due to removal of protection domain crossing overhead for data transfer between device and processing buffers. It is reliable due to kernel thread scheduling (compared to split-level scheduling). The EPA interface provides configuration and execution flexibility on-line, with performance-oriented "reliable" execution (in terms of expected number of missed soft and termination deadlines).

The EPA interface is intended to allow an application to negotiate and adjust performance for both periodic pipelines requiring isochrony and aperiodic pipeline execution. Many scenarios exist for on-line RT EPA/PCIP service renegotiation for continuous media, digital control, etc. [14]. 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 pipeline modules; ***<guaranteed, reliable, best-effort>***
   i) Module computation time; ***<WCET for guaranteed, expected execution time C_{exp} for reliable, none for best-effort>***
   ii) Off-line module execution samples for ***C_{exp};<{Sample-array}, [distribution-free or (normal, σ, C_{exp})]>***
   iii) Deadlines; ***<computed-deadlines if confidence specified, specified (D_{term}, D_{soft})>***
2) Interference assumption common to all ***reliable*** pipeline modules; ***<maximum, high-conf>***
3) Input source stage or device interface; ***<source>***
4) Input and output block sizes; ***<S_{in}, S_{out}>***
5) Desired termination and soft deadline confidence for ***reliable***; ***<computed-conf if deadlines specified, specified (high-conf, low-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); ***<T_{in}, T_{out}>***

Interface parameters 5-7 can be controlled on-line during a service epoch, whereas 1-4 are set up for the life of a service epoch. The application can negotiate services by one of the following methods: 1) specifying desired deadline confidences asking the EPA to compute corresponding soft and termination deadlines; 2) specifying desired soft and termination deadlines and asking the EPA to compute corresponding maximum deadline confidences; or 3) specifying deadlines and desired confidences which are checked for viability by the EPA. The computations used for all three are based upon an EPA interface to DM which supports reliable soft deadlines given pipeline stage execution time confidence intervals instead of deterministic WCET. The negotiative control provided by RT EPA/PCIP is envisioned to support isochronous and event-driven applications which can employ and control pipelines for guaranteed or reliable execution performance.

2.0 EPA-DM Thread Scheduling

The 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 [3]. 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 a soft or termination deadline is missed, and specifically will abort any thread not completed by its termination deadline. Likewise, the EPA will buffer

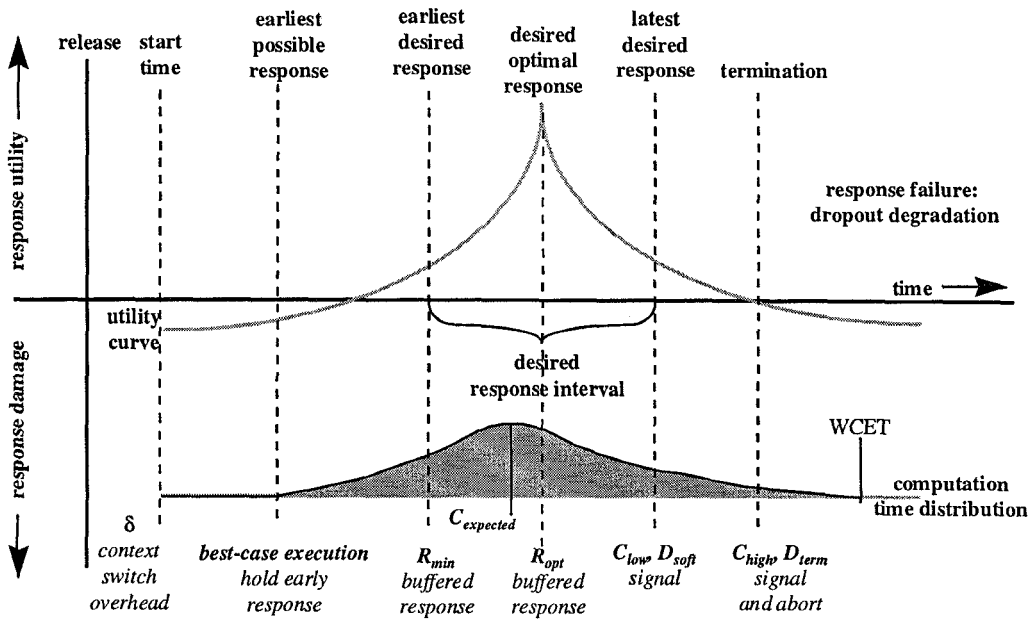

Figure 2: Execution Events Showing Utility and Desired Response

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 (a "hard deadline"), the pipeline must be configured for guaranteed service rather than reliable. The DM scheduling policy and admission test are used due to their ability to handle execution where deadline does not equal period [1]. 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.

The EPA-DM schedulability test eases restrictions on the DM admission requirements to allow threads to be admitted with only expected execution times (based on an execution confidence interval), rather than requiring WCET. The expected time is based on off-line trials to determine the execution time confidence interval. Knowledge of expected time is refined on-line by the EPA each time a thread is run. By removing the WCET requirement, more complex processing can be incorporated, and pessimistic WCET assumptions (e.g. cache misses and pipeline stalls) need not reduce utility of performance-oriented pipelines which can tolerate occasional missed deadlines (especially with known 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. When a module is loaded, a sufficient sample set of computation times from off-line measurements must be provided for either distribution-free confidence estimates, or a smaller set for an assumed distribution. High variance difficult-to-model execution stages such as MPEG (Moving Picture Experts Group) compressed digital video, may require a large number of off-line samples for distribution-free confidence estimates. However, from the sample-derived model, the computation time used in the schedulability tests is computed based upon desired confidence for meeting soft and termination deadlines. 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 $C_{exp}(i)$ = expected execution time; $D_{soft}(i)$ = soft deadline; $D_{term}(i)$ = termination deadline; δ = context switch overhead, and T(i) = period; with the DM condition that $C_{exp}(i) + \delta \leq D_{soft}(i) \leq D_{term}(i) \leq T(i)$. The worst-case confidence interval execution times $C(i)_{low} + \delta \leq D_{soft}(i)$ and $C(i)_{high} + \delta \leq D_{hard}(i)$ 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, ignoring δ, consider two threads that have a normal distribution of execution times (the normal distribution assumption 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: $C_{exp}(1)=40$, $C_{low}(1)=48.7$, $C_{high}(1)=49.9$, $\sigma(1)=15$, $N_{trials}(1)=32$, $Zp_{low}(1)=3.29$ for low-conf=99.9%, $Zp_{high}(1)=3.72$ for high-conf=99.98%, WCET(1)=58, $D_{soft}(1)=50$, $D_{term}(1)=60$, and T(1)=250

thread i=2: $C_{exp}(2)=230$, $C_{low}(2)=247.3$, $C_{high}(2)=262.9$, $\sigma(2)=50$, $N_{trials}(2)=32$, $Zp_{low}(2)=1.96$ for low-conf=95%, $Zp_{high}(2)=3.72$ for high-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 i=1 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 i=2 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 [1] with EPA execution time confidence interval inputs rather than just worst-case execution time, the schedulability with desired confidence in deadlines can be derived as shown below.

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

eq 1: From probability for a normal distribution:

$$C_{low\ or\ high}(i) = C_{exp}(i) + Zp_{low\ or\ high}(i)\left(\frac{\sigma(i)}{\sqrt{N_{trials}(i)}}\right)$$

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

$I_{max}(i)$ 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. Each interfering thread may run to its termination deadline $C_{term} = \max(D_{term}, C_{high})$. A less pessimistic interference assumption based on C_{high} only rather than $\max(D_{term}, C_{high})$ can be specified during negotiation for service with the EPA. However, it is preferable to simply allow the EPA to compute deadlines or confidences such that $C_{high} + \delta = D_{term}$ and $C_{low} + \delta = D_{soft}$. In the following example, deadlines and confidences are specified, and it should be noted that the desired confidences are much less than the maximum possible confidence.

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: $\left(\frac{48.72}{50}\right) \leq 1.0$ and $\left(\frac{49.86}{60}\right) \leq 1.0$

for $C_{low}(1)$and $C_{high}(1)$; likewise $\left(\frac{58}{60}\right) \leq 1.0$ 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: $\left(\frac{C_{low\ or\ high}(2)}{D_{soft\ or\ hard}(2)}\right) + \left(\frac{I_{max}(2)}{D_{soft\ or\ hard}(2)}\right) \leq$

1.0 ?; $I_{max}(2) = \text{ceiling}\left(\frac{D_{term}(2)}{T(1)}\right) C_{term}(1)$

In the worst case, maximum interference occurs when all higher priority threads execute until they are aborted by the EPA.

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

simplifying eq 2&3: $\left(\frac{310}{420}\right) + 2\left(\frac{60}{420}\right) \leq 1.0$?

WCET cannot be scheduled. *(inequality is not true)*
C_{low}, C_{high} can be scheduled. *(note: thread i=1 interferes up to its termination deadline twice here)*

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 cannot be scheduled if only WCET is considered. In this case, WCET, a statistical extreme, leads to rejection of a thread set which can be scheduled with ≥ 99.98% probability of meeting termination deadlines.

3.0 Implementation and Evaluation

The mechanism is being implemented in RT-Mach [17] with modifications to the rate monotonic kernel thread scheduling interface 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 configuring pipeline modules and device interface modules. The RT EPA/PCIP is also being prototyped as a Solaris 2.5.1 module set using the real-time kernel thread scheduling class. Both kernels provide the basic features needed for the RT EPA/PCIP, including: 1) fully preemptable kernel, kernel threads, priority inheritance [13], fixed priority preemptive scheduling, and a real-time thread class which preempts all non-real-time threads.

The 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 UAV is a blimp constructed of mylar and neoprene weather balloons with a basswood undercarriage for the sensors, actuators, motors, computer, and microcontrollers.

The UAV testbed experiments using the RT EPA/PCIP mechanism on-board the UAV will use a basic set of UAV 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 steady-level flight, climbing, turning and station keeping. Likewise, the digital control loops will be parametrically controlled by a shared-control application to coordinate more complex semi-autonomous functions such as coverage patterns. Finally, the operator will be provided with continuous media video from on-board "QuickCam" output piped to a ground display through "WaveLAN" radio frequency point-to-point transport.

4.0 Related Work

A number of pipeline mechanisms for continuous media have been developed [5], [6], [8]. However, most common implementations include application-level processing with user-level threads and device buffers mapped from kernel space into user-space rather than an "in-kernel" modules. The *splice* mechanism is most relevant since it operates "in-kernel" using loadable modules or simple streaming as the RT EPA/PCIP will, and was shown to have up to a 55% performance improvement [6]. However, to our knowledge, *splice* does not provide a configuration and on-line control interface like the EPA. The RT-Mach processor capacity reserves interface to the rate monotonic fixed priority scheduler provides a performance-oriented interface to an existing real-time scheduling policy like the EPA, but with a very different abstraction from confidence [10].

Many examples of periodic hard real-time digital control streams exist [9], 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 kernel pipes within a general operating system environment would be useful for sophisticated industrial applications [18]. Finally, many real-time semi-autonomous and "shared control" projects are in progress [4] [7], including applications where occasional missed deadlines are not catastrophic [12].

5.0 Conclusion

Experiments will be implemented using both the RT EPA/PCIP and user-level applications to compare performance. However, the RT EPA/PCIP 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 pipelines through its EPA interface. A fundamental aspect of the EPA performance control is based on the EPA-DM confidence interval approach. 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.

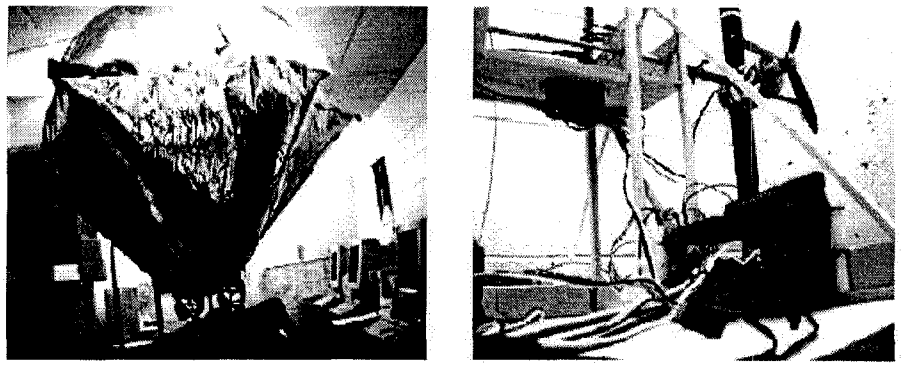

Figure 3: Prototype (left); Full-scale (right)

Finally, experiments will also be evaluated in terms of real-time parameters such as video stream dropouts, latency variation, UAV heading drift, etc., to evaluate the reliability afforded by the EPA to applications. These experiments will be run individually and simultaneously to evaluate use of the RT EPA/PCIP mechanism for complex real-time applications involving multimedia and interaction between users and the semi-autonomous UAV with "shared" control RT EPA/PCIP automation.

6.0 References

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

[2] 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.

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

[4] 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 Int'l Conf. on Intelligent Robots and Systems, Yokohama, Japan, July, 1993.

[5] Coulson, G., Blair, G., and Robin, P., "Micro-kernel Support for Continuous Media in Distributed Systems", Computer Networks and ISDN Systems, No. 26, 1994.

[6] Fall, K., and Pasquale, J., "Improving Continuous-Media Playback Performance With In-Kernel Data Paths", Proceedings of the IEEE Int'l Conf. on Multimedia Computing and Systems, Boston, MA, June 1994.

[7] Fleischer, S., Rock, S., Lee, M., "Underwater Vehicle Control from a Virtual Environment Interface", ACM Symp. on Interactive 3D Graphics, Monterey CA, 1995.

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

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

[10] Mercer, C., Savage, S., Tokuda, H., "Processor Capacity Reserves: Operating System Support for Multimedia Applications", Proceedings of the IEEE Int'l Conf. on Multimedia Computing and Systems, May, 1994.

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

[12] Paulos, E., and Canny, J., "Delivering Real Reality to the World Wide Web via Telerobotics", IEEE Int'l Conf. on Robotics and Automation, 1996.

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

[14] Siewert, S., "Operating System Support for Parametric Control of Isochronous and Sporadic Execution in Multiple Time Frames", Ph.D. proposal, U. of Colo. Boulder, 1996.

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

[16] 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.

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

[18] Törngren, M., "On the Modelling of Distributed Real-Time Control Applications", Proceedings of the 13th IFAC Workshop on Distributed Computer Control Systems, Toulouse-Blagnac, France, September, 1995.