Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

Evaluation of a CPU Scheduling Mechanism for Synchronized Multimedia Streams

Ralf Steinmetz, Lars C. Wolf

IBM European Networking Center
Vangerowstraße 18, D-69115 Heidelberg, Germany
Phone: +49-6221-59-4280, Fax: +49-6221-59-3300
{steinmetz, lwolf}@vnet.ibm.com

Abstract: Multimedia applications handling audio and video data have to obey time characteristics of these media types – for a single stream of multimedia data as well as for the synchronization of related streams. Correctness with respect to time constraints requires mechanisms which lead to favored processing of multimedia operations. CPU scheduling techniques based on the experience from real-time operating systems offer a solution and provide multimedia applications with the ability to meet time-related Quality of Service (QoS) requirements.This presentation starts with an overview on the required Quality of Service for synchronized audio and video streams. Subsequently it describes an implementation of a CPU scheduler designed to run under IBM's AIX. The evaluation of the implementation based on measurements shows that the scheduler is able to support the time requirements of multimedia applications and that such mechanisms are indeed necessary since otherwise deadline violations occur.

Keywords: multimedia, real time, scheduling, synchronization, operating system support, quality of service, QoS

1 Introduction

In accordance with [StNa95] we understand multimedia in the following way: A multimedia system is characterized by the integrated computer-controlled generation, manipulation, presentation, storage, and communication of independent discrete and continuous media. The digital representation of data and the synchronization between these various data are the key issues for integration. Synchronization is needed to ensure a temporal ordering of events in a multimedia system.

The temporal ordering must also be applied to related data streams, where one of the more common relationships is the simultaneous playback of audio and video in 'lip synchronization'. Both media must be 'in sync' otherwise the result will not be adjudged as satisfactory. In general synchronization involves relationships between all kinds of media including pointers, graphics/images, animation, text, audio, and video. As human perception varies from individual to individual it is usual in subjunctive experiments to carry out experiments with a sample of individuals to obtain a reasonable cross-section of results.

The lack of in-depth analysis of synchronization between the various kinds of media and, in particular lip and pointer synchronization led us to conduct some experi-

Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

ments of our own to obtain results that allow us to quantify the quality of service (QoS) requirements for multimedia synchronization. The details are reported in [Stei96].

These QoS constraints must be achieved by the multimedia system. Real-time CPU scheduling techniques which serve multimedia application processing with respect to their time-criticality provide a solution to these problems. For a more detailed description of various methods to express time, the implementation of the respective scheduler and a larger set of measurements of experimental evaluation see [WBVo94].

In this paper we first discuss the results of a series of experiments on human media perception that may be used as ‘Quality of Service’ guidelines. The results show that humans perceive media streams as ‘in sync’ even when some skew between related data streams exists. Sections 3 and 4 discuss the chosen scheduling algorithm and its implementation. In Section 5 we give an evaluation of the scheduler’s ability to provide real-time guarantees. This is part of our work on the transport system HeiTS (Heidelberg Transport System) [WoHe94] which offers real-time communication support for distributed multimedia applications.

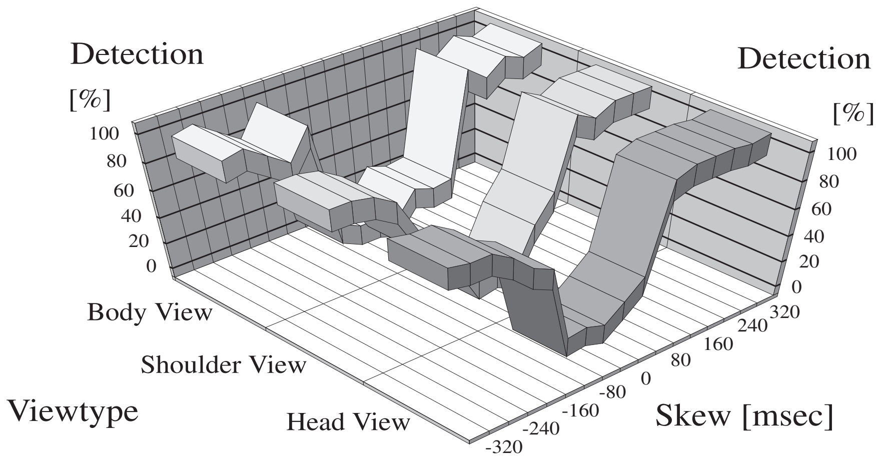

Figure 1: Detection of synchronization errors with respect to the three different views. Left part, negative skew; video ahead of audio; right part, positive skew; video behind audio

2 Synchronization Quality of Service

‘Lip synchronization’ refers to the temporal relationship between an audio and video stream for the particular case of humans speaking. The time difference between related audio and video data is known as ‘skew’. Streams which are perfectly ‘in sync’ have no skew, i.e., 0 ms. We conducted experiments and measured the skews that were perceived as ‘out of sync’. In our experiments users often mentioned that something is wrong with the synchronization, but this did not disturb their feeling for the quality of the presentation. Therefore, we additionally evaluated the tolerance of the users by asking if the data out of sink affects the quality of the presentation.

Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

In order to get accurate and good skew tolerance levels we selected a speaker in a TV news environment in three different views: head, shoulder and body. We recorded the presentation and then re-played it with artificially introduced skews created with professional editing equipment skewed at intervals of 40ms i.e. -120ms, -80ms, -40ms, 0ms, +40ms, +80ms, +120ms.

Figure 1 provides an overview of the results. The vertical axis denotes the relative number of test candidates who detected a synchronization error, regardless of being able to determine if the audio was before or after the video. On the left side of the central axis the graph relates to negative skew values where the video is ahead of the audio and on the right side where the audio is ahead of the video. The curves in the figure are asymmetrically, the right hand side of the curves are steeper than the left sides. The reason for this is that we are used to the situation where the motions of the lips are perceived a little before the audio is heard from our daily experience because the velocity of light is larger than that of sound.

The ‘body view’ curve is broader than the ‘head view’ curve as in the former a small skew is easier to notice. The ‘head view’ is also more asymmetric than the ‘body view’, due to the fact that the further away we are situated, the less noticeable an error is.

We identified the ‘in sync’ region that spans a skew between -80 ms (audio behind video) and +80 ms (audio ahead of video). In this zone most of the test candidates did not detect the synchronization error. Very few people said that if there was an error it did affect their notion of the quality of the video. Additionally, some results indicated that the perfect ‘in sync’ clip was ‘out of sync’. Our conclusion is that lip synchronization can be tolerated within these limits.

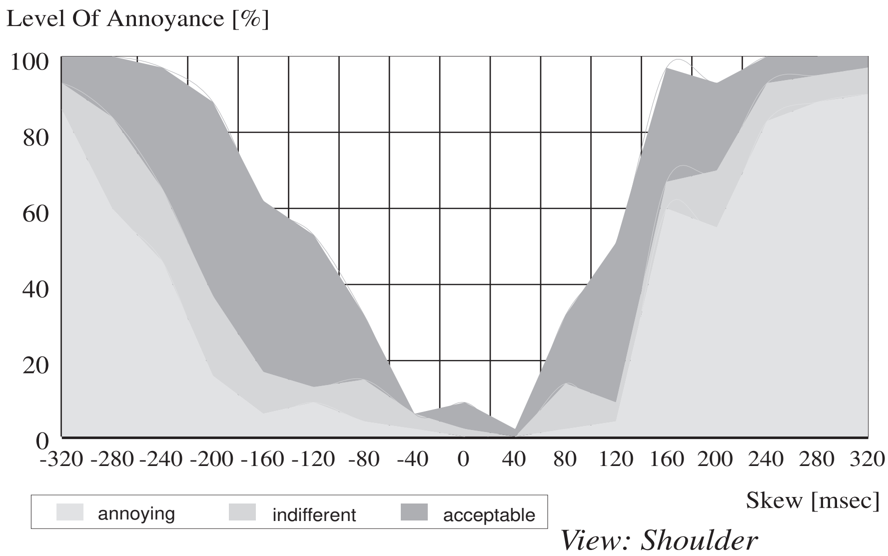

Figure 2: Level of annoyance at shoulder view

Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

Just as important as the error itself is the effect which such an ‘out of sync’ video clip has on perception. Therefore the test candidates were asked to qualify a detected synchronization error in terms of being acceptable, indifferent, or annoying. Out of these answers we derived a ‘level of annoyance’ graph, Figure 1, and verified the borders of the in-sync area.

The envelope curve (the upper edge of the dark area) defines the amount of candidates who detected a synchronization problem. This is the same curve for the ‘shoulder view’ as shown in Figure 1 (just without a spline interpolation).

The dark grey areas relate to all test candidates who detected a synchronization error and found the clip watchable with this synchronization error. In a small follow-on experiment we selected a few test candidates who would tolerate such a skew and showed them a whole movie with a -160 ms skew where the video was ahead of the audio. Annoyances were reported just after the beginning of the film but soon after it was noted that the candidates concentrated on the content instead of being attracted/ distracted by the synchronization offset. The curve at the bottom of the dark grey area shows an asymmetry between sound and light as mentioned before.

The light grey area indicates the people who found the skew distracting. During the evaluation phase of this study on synchronization we introduced a skew of +80 ms and -80 ms into two whole movies which were shown to a few candidates who found it irritating but still could concentrate on the content. The same experiment however with a skew of -240 ms or +160 ms would lead to a real distraction from the content and to a severe feeling of annoyance.

The required QoS for synchronization is expressed as the allowed skew. The QoS values shown in Table 1 relate to presentation level synchronization. Most of them result from exhaustive experiments and experiences, others are derived from literature as referenced in [Stei96]. To our understanding; they serve as a general guideline for any QoS specification in [Stei96]. As first order result to serve as a general guidance, these values may be relaxed depending on the actual content.

We can therefore conclude that skews between -80 ms and +80 ms are deemed acceptable by most casual observers.

3 QoS provided through Scheduling

QoS management in multimedia systems is based on two models [Vogt95]. The workload model is used to describe the load an application will place onto the system. The QoS model is used by an application to define its performance requirements and by the system to return corresponding performance guarantees.

The QoS model used in HeiTS has three parts: (1) The throughput part describes the bandwidth required for or granted to a multimedia connection. It consists of the three parameters of the workload model described below. (2) The delay part defines the maximum delay a multimedia packet can experience on its way from the source to the sink of the connection. (3) The reliability part describes how packet losses and bit errors within packets are handled. They can be ignored, indicated or corrected.

In order to meet synchronization QoS of two streams (respectively two threads), their delay jitter must be less or equal to the skew discussed in the previous section.

Media		**Mode, Application**	**QoS**
video	animation	correlated	+/- 120 ms
	audio	lip synchronization	+/- 80 ms
	image	overlay	+/- 240 ms
		non overlay	+/-500 ms
	text	overlay	+/- 240 ms
		non overlay	+/-500 ms
audio	animation	event correlation (e.g. dancing)	+/- 80 ms
	audio	tightly coupled (stereo)	+/- 11 ms
		loosely coupled (dialog mode with various participants)	+/- 120 ms
		loosely coupled (e.g. background music)	+/- 500 ms
	image	tightly coupled (e.g. music with notes)	+/- 5 ms
		loosely coupled (e.g. slide show)	+/- 500 ms
	text	text annotation	+/- 240 ms
	pointer	audio relates to showed item	-500 ms, + 750 ms[1]

Table 1: Quality of Service for synchronization purposes

1. pointer ahead of audio for 500 ms, pointer behind audio for 750 ms

The workload for multimedia systems is periodic by nature – consider for instance an application presenting audio or video data where data packets must be transmitted at certain instants. To describe the load induced into the system, HeiTS uses the *Linear Bounded Arrival Process (LBAP)* as its workload model. The LBAP model assumes data to be processed as a stream of discrete units (*packets*) characterized by three parameters: S=maximum packet size, R=maximum packet rate (i.e., maximum number of packets per time unit), and W=maximum workahead.

Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

3.1 Schedulability Test and Priority Assignment Scheme

The target operating system for the implementation is AIX, IBM's UNIX derivate. In addition to the well-known *multi-level-feedback* (MLFB) scheduling it provides a set of *fixed* priorities at the highest priority levels (priorities 0-15), which are even higher than the AIX scheduler's priority. Unlike the other (MLFB) priorities these priorities are not modified by the AIX scheduler and can be used for real-time processing.

Assigning priorities to processes produces a considerable overhead that cannot be neglected. Therefore, we do not utilize a *dynamic* scheme such as earliest deadline first (EDF) but use a *static* priority assignment scheme according to the rate monotonic (RM) algorithm where a process with a short period (i.e., a high rate) receives a high priority [LiLa73][LSDi89][Stei95]. Priorities are computed at application establishment time and are not changed dynamically during application lifetime. Only when a newly established application needs a priority level that is already in use the existing priorities are shifted to make room for the new application handling process. The priorities are ordered in a way that guaranteed processes possess the highest priorities and statistical processes use the lower part of the real-time priorities. All processes not subject to real-time constraints are handled by the AIX system scheduler and use priorities below the real-time priorities.

4 Implementation

The actual scheduling is performed through a set of kernel functions (AIX provides mechanisms for adding such system calls) that must be called by the process that wants to be scheduled. This is more efficient than implementing the scheduler as a separate process (like the AIX system scheduler) because it saves the context switch between the process to be scheduled and the scheduler process itself.

Requiring that the process calls the scheduler function explicitly leads to "voluntary scheduling" and may seem dangerous. However, all code allowed to run in an environment where it is possible to use real-time priorities has to be established by an authorized user. Thus, only approved code will be subject to real-time scheduling and, therefore, especially with reflection on the performance gain, this approach can be regarded as secure.

To achieve proper scheduling of real-time processes some assumptions about the structure of the processes have to be made. As shown in Figure 3, it is assumed that after creating an application the process responsible for handling the data of this application is performing a program loop and processes one data packet (e.g., a video frame) in every iteration. This continues until the application is finished and the process is not subject to real-time scheduling any more.

Before processing a newly arrived data packet the scheduler must check whether accepting this packet would violate the LBAP characteristic (i.e. the workload specification) of the data stream. This check can be done in a blocking or a non-blocking way. The blocking test is performed by the function `LBAP_enforce` and enforces the observance of the LBAP property of the data stream: The process is left in a wait state until the logical arrival time of the packet is reached.

Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

In the non-blocking test implemented in the function `LBAP_poll` the scheduler simply returns the calculated logical arrival time of the data packet and the information whether accepting this packet violates the LBAP properties of the data stream or not.

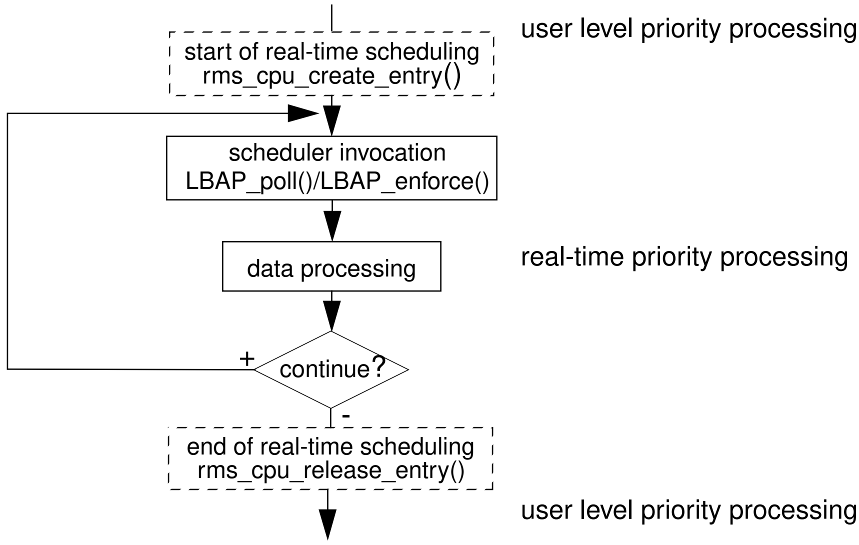

Figure 3: Processing structure

5 Evaluation

To show the effect of using the scheduler for different multimedia applications we performed a series of measurements. We wanted to answer the following question: In which way does the use of the scheduler influence the behavior of the application and the system as a whole, i.e., are deadline violations indeed avoided and to what extend. The CPU scheduler function `LBAP_enforce` was instrumented in such a way that it generates events describing the laxity of the calling process, i.e., the time until the process reaches its deadline. Positive values indicate that the process still has time before the deadline is reached, therefore, it is operating correctly; negative values indicate that the process violated its deadline, it is not able to perform its function in time.

In those cases where several real-time processes were running concurrently the events are given in generation-time order, i.e., they are not ordered by processes unless otherwise stated. The charts shown below are extracts from much longer measurements series to increase readability. Each of them shows 200 values which have been taken from the middle of the sequence of values (the generation of measurement values started later than the processes under consideration to reduce start-up effects), each point in a graph represents a single event. The measurement values are given in seconds.

All measurements were performed on a mostly idle workstation (IBM RISC System /6000, Model 360 with AIX 3.2.4) which was not modified during the measure-

Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

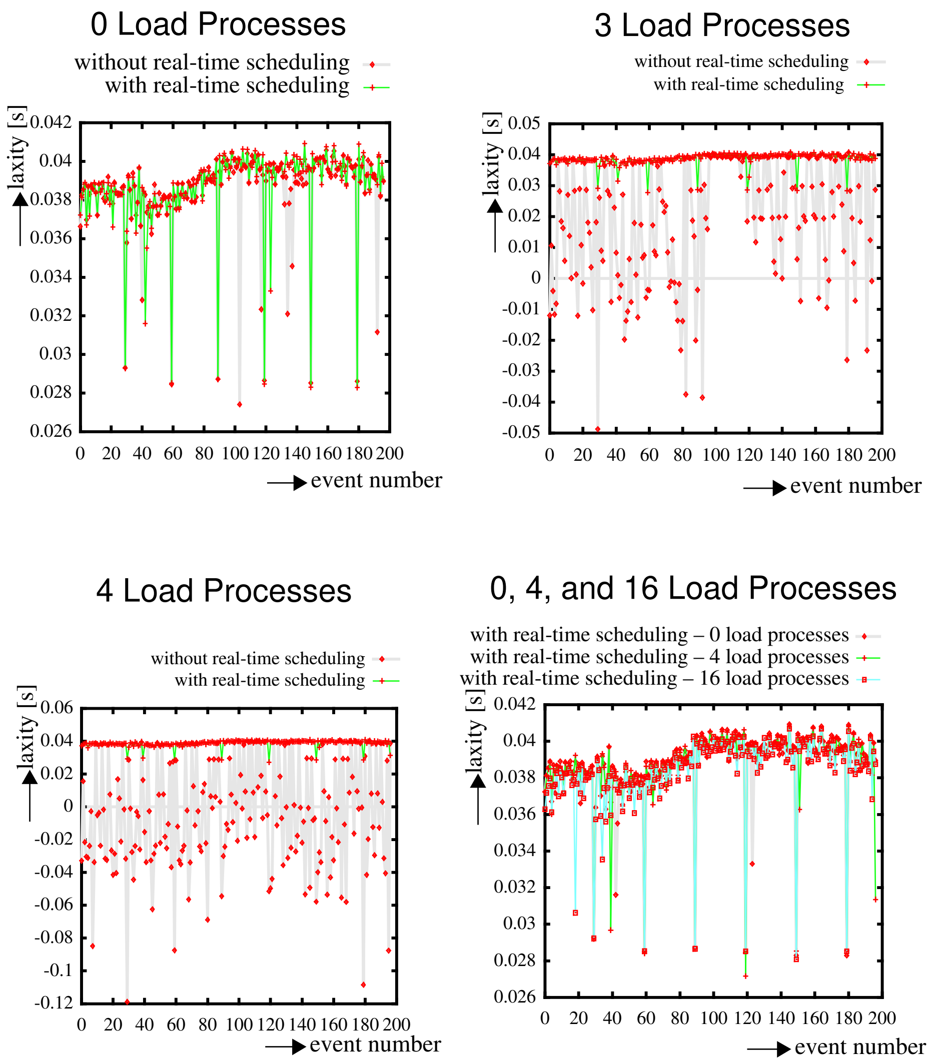

Figure 4: Video playback application

ments, e.g., simple applications such as mail, etc. were running as usual. However, none of these programs used much CPU processing time. These types of applications are running during normal workstation operation periods as well, thus, disabling them during the measurements might lead to slightly more regular measurement results but not to results which are better applicable to real world scenarios.

The measurements were performed with a varying system load (background load). The system load was generated artificially by synthetic, non real-time, computation processes performing simple integer calculations. Hence, in principle these processes were always ready to run which also lead to low priority due to UNIX scheduler characteristics. Therefore, normal, user created system load might be even harder than this

Joint Conference: 8th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation and 8th GI/ITG Conference on Measuring, Modelling and Evaluating Computing and Communication System, September 18-22, 1995, Heidelberg, Germany

synthetic load. We used 0, 1, 2, 3, 4, or 16 of these load processes during the measurements. Running 16 processes leads to a heavily loaded system, the other loads resemble loads easily created during normal workstation operation.

The measurements were performed with programs using the CPU scheduler's real-time characteristics followed by measurements with the same programs without performing real-time scheduling (using the time provision mechanisms of the scheduler, i.e., executing with the specified rate). The load generated by the programs is the same in both cases (since we use the static RM scheduling algorithm without workahead scheduling, we have no additional costs for the real-time processes during run time).

Different application scenarios with different setups were investigated, here we focus on video playback as an endsystems applications. The measured program uses one process for its operations. The chosen video consists of 15 frames/s, i.e., 66.6 ms/frame which was also set as the processing rate of the program. The processing time needed per period is in the average approximately 28 ms which results in a total CPU usage of about 42%.

The compressed data read by the program was stored in a local file which was cached into main memory by running the program first without measuring it. The file was small enough to fit into the cache.

Figure 4 shows the results for measurements with varying loads. If no load except the measured process exists in the system, no deadline violations occur even without using real-time scheduling.

If a load of medium size (three or more processes) is introduced into the system, the considered application is not able to provide acceptable service to the user. The last graph in the figure illustrates that by using real-time scheduling, the application does not suffer from any deadline violations, even if we introduce a high load (up to 16 processes) into the system.

All plots of applied real-time scheduling show a laxity less than 42ms. The minimum laxity is never less than 26 ms, hence, the laxity is always in the interval [26 ms, 42 ms]. In term of lip synchronization QoS, our system is able to provide a skew of less than 30ms. Hence, we can meet the demanded lipsynch QoS of less then 80 ms.

6 Conclusions

In this paper we tied together our work in user perception of media synchronization [Stei95] with the implementation and evaluation of appropriate scheduling techniques [WBVo94] to meet the required QoS demands: Lip synchronization for the playback of multimedia data.

Further research should be performed to refine the given table of synchronization QoS; the values shall be verified and refined by extensive user perception tests. Our mapping strategy of skew onto maximum allowable delay jitter of the stream is a straight forward approach. It is not clear if such a skew can alternatively be directly integrated into the scheduling models.

The measurements comprises a set of (what we consider) representative applications. However, further applications might lead to different results. The dependency between scheduling and synchronization is still in its infancy. To the knowledge of the authors so far no multimedia operating system primitives allow for the specification of a synchronization skew. No scheduling mechanism takes into account the timing relationship to other request, i.e., a synchronization skew.

Acknowledgments

The authors would like to thank W. Burke, C. Vogt, C. Engler and M. Ehrmantraut for the work and discussions related to the topic of this paper.

References

[FBZh92] Domenico Ferrari, Anindo Banerjea, Hui Zhang, “Network Support For Multimedia: A Discussion of the Tenet Approach”, TR-92-072, International Computer Science Institute, Berkeley, CA, USA, 1992.

[LiLa73] C.L. Liu and James W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Realtime Environment”, Journal of the ACM, vol. 20, no. 1, pp. 47-61, 1973.

[LSDi89] John Lehoczky, Lui Sha and Ye Ding, “The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior”, Proceedings of the Tenth IEEE Real-Time Systems Symposium, pp. 166-171, Santa Monica, CA, USA, 1989.

[NaSt95] Klara Nahrstedt, Ralf Steinmetz, “Resource Management in Networked Multimedia Systems”, IEEE Computer, Vol.28 No. 5, May 1995, pp. 52-64.

[Stei95] Ralf Steinmetz, “Analyzing the Multimedia Operating System”, IEEE Multimedia, vol. 2 no. 1, Spring 1995, pp. 68-84.

[Stei96] Ralf Steinmetz, “Human Perception of Jitter and Media Synchronization”, to appear in IEEE JSAC, vol. 14 no. 1, January 1996.

[StNa95] Ralf Steinmetz, Klara Nahrstedt, “Multimedia: Computing, Communication and Application”, Prentice Hall, July 1995, ISBN 0-13-324435-0.

[Vogt95] Carsten Vogt, “Quality-of-Service Management for Multimedia Streams with Fixed Arrival Periods and Variable Frame Sizes”, ACM Multimedia Systems, vol. 3, no. 2, pp. 66-75, May 1995.

[WoHe94] Lars C. Wolf and Ralf G. Herrtwich, “The System Architecture of the Heidelberg Transport System”, ACM Operating Systems Review, vol. 28, no. 2, pp. 51-64, April 1994.

[WBVo94] Lars C. Wolf, Wolfgang Burke, Carsten Vogt, “Evaluation of a CPU Scheduling Mechanism for Multimedia Systems”, Technical Report 43.9407, IBM European Networking Center Heidelberg, Germany, 1994.