An Intermedia Skew Control System for Multimedia Data Presentation[1]

T.D.C. Little and F. Kao

Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
tdcl@bu.edu

MCL Technical Report 11-01-1992

Abstract–Delivery of multimedia traffic from distributed sources to a single destination relies on an accurate characterization or guarantee of the behavior of each involved system component. If anticipated bandwidth, delay, and loss characteristics are violated, statistical scheduling approaches fail, resulting in the loss of established source-to-destination synchronization and the introduction of intermedia skew.

We propose an intermedia skew control system for accommodating a class of anomalous behaviors including spurious channel overload, data losses due to corruption or congestion control, and variations in source and destination clock rates. The control system is designed to operate independently from the data source, accommodating the data delivery requirements of an arbitrarily corrupted input stream. Intermedia skew, buffer underflow, and buffer overflow are controlled by regulating the playout rate of each stream through frame drop and duplication. This mechanism is used in conjunction with a statistical source pacing mechanism to to provide an overall multimedia transmission and resynchronization system supporting graceful service degradation.

Keywords: Multimedia synchronization, flow control.

[1]In *Proc. 3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992, pp. 121-132. This material is based upon work supported in part by the National Science Foundation under Grant No. IRI-9211165.

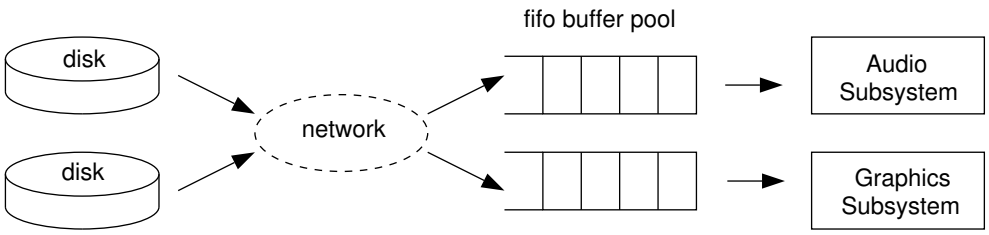

Figure 1: Data Retrieval from Multiple Independent Sources and Multimedia Playout

1 Introduction

One of the difficulties in supporting multimedia data delivery is the synchronization of related media in what we call *intermedia synchronization*. The lip-sync of audio and video is typical, but synchronization to both real-time and to other media can be required in a multimedia application. When data are transmitted (or stored) via different channels (disks) and synchronization is required, a mechanism is necessary to ensure synchronous presentation, or *playout*, to the application. Interleaving, or multiplexing of interrelated media can be used for multimedia storage or transmission. However, it can only be applied to a single storage subsystem or communication channel. Furthermore, the interleaving requires that all media have the same quality of service [3], and prohibits the use of congestion control mechanisms that reduce load by dropping data of a single medium. Initial attempts to satisfy the synchronization problem were achieved by allowing the destination to track the source [6, 13]. In our work we assume a general model supporting multiple media, allowing intermedia synchronization among multiple sources in a multidrop fashion (Fig. 1).

Previously, we investigated a session establishment protocol for reservation of channel resources [11]. In our current work we achieve intermedia synchronization after connection establishment by monitoring and controlling skew and queue levels by selective frame drop and duplication. This scheme prevents disruptions in playback caused by unanticipated temporary overloads due to events such as file transfers, and effectively extends the buffering capability of the receiver without increasing memory use or latency. Although this would seem to duplicate the efforts provided by a statistical resource allocation mechanism for bandwidth and delay, we anticipate both short- and long-term guarantee violations that must be accommodated in any real system. In addition, data frames can be lost in the network (e.g., via asynchronous transfer mode congestion control) which will require the source to manage intermedia synchronization in the presence of frame losses. Our system is

unique in its ability to recreate a nominal playout sequence in the presence of these aperiodic losses, possibly due to format conversion (e.g., 10 frames/s to 30 frames/s). Furthermore, this system operates independently of the sources in the absence of clock rate mismatches. In essence, a statistical resource allocation mechanism can provide a course reservation of suitable resources, but the fine-tuning required at playout time is provided by our control system.

Drop and duplication of frames has also been used to reconcile mismatches in data playout and generation rates by Anderson and Homsy [1]. Our proposal differs in its provision of source–destination independence, and its generality in providing fine and gross skew correction due to data losses. Furthermore, our solution is designed to correct skew without introducing noticeable discontinuities in data playout. Drop and duplication is also proposed to correct intermedia skew by Rangan et al. [17]. With this approach, synchronization is ensured (in the absence of a global clock) by computing relative timing for a set of distributed sources. In contrast, our system is designed to be easily implementable, independent of the data source, and able to accept data at arbitrary arrival rates. Delay and delay variations are absorbed by buffering while losses or rate conversion are accommodated by the playout correction algorithm.

Other related work in multimedia synchronization includes the study of real-time operating systems for supporting audio and video synchronization [2, 12, 14, 16, 19], feedback control systems for locking on source delivery rates [4, 5, 6, 15], and rate-based transmission control [7, 9]

In the remainder of this paper we describe our system for providing intermedia synchronization. In Section 2 we define intermedia synchronization and describe its control. In Section 3 we outline the algorithms supporting the control mechanisms. Section 4 discusses alternative control policies and observations from our simulations. Section 5 concludes the paper.

2 Intermedia Synchronization

Synchronization is defined as the occurrence of multiple events at the same instant in time. Intermedia synchronization describes a similar timing constraint among a set of multimedia streams. In this section we characterize intermedia synchronization and introduce a framework for satisfying such timing constraints in a general-purpose computer environment.

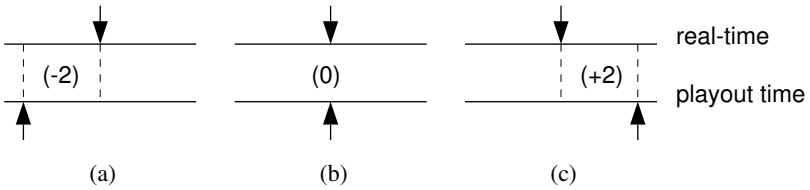

Figure 2: Skew: (a) Lagging, (b) None, (c) Leading

2.1 Characterization of Synchronization

Timing parameters can characterize intermedia and real-time synchronization for the delivery of periodic (e.g., audio and video) and aperiodic data (e.g., text and still images). Parameters applicable to aperiodic data are *maximum delay, minimum delay,* and *average delay* as measured with respect to real time or with respect to other aperiodic data. For periodic data, maximum, minimum, and average delay are also applicable to individual data elements, but in addition, *instantaneous* delay variation or *jitter* is important for characterizing streams. *Skew*, related to jitter, describes an average jitter over an interval.

For periodic data such as audio and video, data can be lost resulting in *dropouts* or gaps in playout. Such losses cause the stream of frames to advance in time or cause a stream *lead*. Similarly, if a data frame is duplicated, it causes the stream to retard in time or a stream *lag* (Fig. 2). By dropping or duplicating frames it is possible to control the rate of playback. Initiation of frame dropouts is permissible for audio and video which have substantial short-term temporal data redundancy. However, for text and graphics, this is not true as their tolerance to delay is greater.

Synchronization is usually defined as absolute, occurring at an instant in time. To provide a tolerance to timing variations, we adopt the following definition that extends a synchronization instant to an interval (similar to Gibbs' definition [8] and Ravindran's divergence vector [18]).

Definition 1 *A composite object with actual playout times* $P = \{\rho_i\}$ *is synchronized with playout reference times* $\Pi_i = \{\pi_i\}$ *iff* $\forall i, |\rho_i - \pi_i| \leq \theta_i$ *where* $\Theta = \{\theta_i\}$ *are the synchronization tolerances between each element and the reference.*

Skew can be measured with respect to real-time as an offset to some mutual presentation start time between the source and destination, or can be measured with respect to another

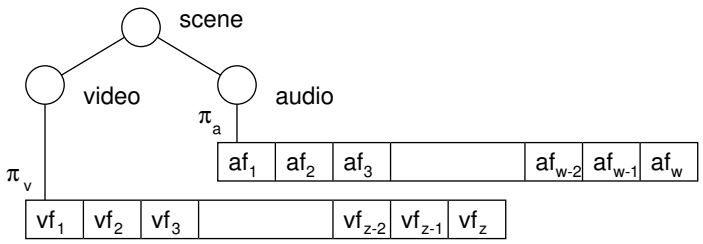

Figure 3: Blocking for Continuous Media

stream. Because many streams are possible, we characterize both intermedia and real-time reference skew for k streams using a matrix representation as $skew = sk_{p,q}$, where $sk_{p,q}$ describes the skew from stream p to stream q (q to p is negative) and the $k+1$th element corresponds to a real-time reference. We also define an interstream tolerance $\Theta = \theta_{p,q}$ and target skew matrix $TSK = tsk_{p,q}$ which indicate tolerances and target values between streams and can be interpreted by a skew control function. Related to skew is data *utilization*. Utilization U describes the ratio of the actual presentation rate to the available delivery rate of a sequence of data. Frame drops will decrease utilization whereas duplicates will increase utilization from its nominal unit value. Either skew or utilization can be used as a control variable.

Skew best measures intermedia synchronization for continuous media. For characterization of discrete events associated with timed playout of text, graphics and still images, we can apply real-time scheduling terminology as already mentioned (e.g., maximum and minimum delay). However, it is often advantageous to decompose segments of continuous media into *blocks* to permit efficient storage and manipulation [10]. With this decomposition, blocks associate a single start deadline with a sequence of periodic media frames, as illustrated in Fig. 3. In this example the decomposition is performed on a motion picture and blocks of audio and video are associated with a single logical scene.

2.2 Coarse Synchronization and Session Scheduling

The nature of a packet-switched communications mechanism is inherently asynchronous as is the retrieval of data from rotating storage subsystems. We model the retrieval and transmission of data across a network as an asynchronous system component which must be managed in order to support real-time data delivery required for multimedia presentation. Flow-control (window) protocols can provide feedback necessary to prevent buffer overflow in

bulk data transfers. For real-time data streams, rate-based flow-control is more appropriate [7]. However, feedback approaches alone cannot provide consistent presentation quality under wide bandwidth variations due to network delays or multimedia object sizes. For this reason, statistical reservation techniques are used.

Coarse synchronization in a multimedia session is facilitated by reservation of adequate resources. This scheduling process consists of resource reservation, connection establishment, and data transfer initiation. In relation to intermedia synchronization, a scheduler manages data transmission times with respect to the source whereas the destination must provide buffering for delay variations and accommodation for intermedia skew.

We assume the availability of a statistical scheduler [11] that, given the characteristics of a channel and a multimedia data stream (D_v, D_p, D_t, C, S_m, π_i, σ_i, $P(fail)$, corresponding to variable, fixed, and channel transmission delays, channel capacity, packet size, playout deadline, object size, and probability of lateness), a schedule can be constructed. The resultant schedule indicates the times to put objects onto the channel between the source and destination and can be used to establish the worst case buffering requirement. In the event that the source and destination clocks are unequal, periodic resynchronization can correct clock differences among involved sites and prevent rate mismatches that lead to queue underflow or overflow.

2.3 Control of Intermedia Synchronization

By using a statistical reservation service, the destination can be configured with a buffer of sufficient length to accommodate measured or guaranteed channel delay variations. However, changes in the channel delay characteristic in time or other spurious violations in delay and bandwidth guarantees can cause buffer overflow or underflow resulting in anomalous playout behavior. We propose a control mechanism to monitor and control levels of queued periodic stream data as well as intermedia synchronization as specified by a target skew matrix. This mechanism can then provide graceful degradation of playout quality during periods of spurious network or device behavior.

One of our assumptions in the design of a control mechanism is that the playout interval for periodic data streams is constant. For example, video frames have a playout interval of 33 ms. Because this playout interval cannot be changed dynamically, we control the playout rate instead by changing the frame drop and duplication rates. If two streams are skewed we can either drop frames from the lagging one to increase its speed or duplicate

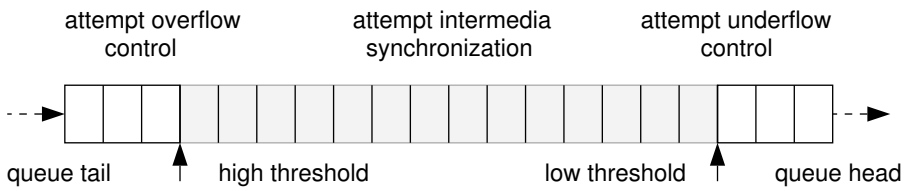

Figure 4: Three Queue States

frames of the leading one to decrease its speed (or both). The tradeoff between dropping versus duplicating is the loss of data versus the increase in end-to-end latency that affects interactive sessions. Depending on the priority of these two considerations and the current queue levels, the appropriate policy can be applied.

We propose two related control paradigms. The first is intended to maintain intermedia synchronization between streams, or between a stream and a real-time reference, when operating at nominal queue levels (Fig. 4). The second is for providing graceful degradation when queue underflow or overflow is pending and attempts at intermedia synchronization are abandoned. For intermedia synchronization, we are investigating three policies with respect to intermedia synchronization: (1) minimize real-time skew, (2) minimize inter-stream skew, and (3) minimize session aggregate inter-stream skew. The first policy targets synchronization of playout with a constant end-to-end delay established during connection set-up. This policy is appropriate for streams such as high-quality digital audio. The second policy places priority on maintaining synchronization between streams rather than between a stream and a real-time reference, but could be used in conjunction with the first policy. This scenario is suitable when relative timing between media is more significant than cumulative skew with respect to real-time. The third policy is to minimize skew over a set of streams within a multimedia session. For queue level control, our goal is to provide graceful degradation prior to queue under or overflow. Control takes effect upon reaching either low or high thresholds and results in modification of the playout rate via frame drop or duplication.

To implement intermedia synchronization we use a control mechanism that monitors the queue level and playout skew (at playout time). Control is provided by changing the playout rate (or utilization) using frame drop and duplication. Stability of the system is provided by determination of appropriate time constants for queue level and skew measurement. Utilization, ideally a continuous parameter nominally of unit value, is representative of the playout rate. Duplication causes $U > 1$ while dropping causes $U < 1$. Utilization over an interval comprised of n frames can be determined using the formula, $U = (pass + slip)/pass$, where

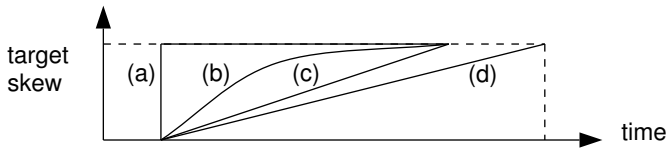

Figure 5: Correction Alternatives. (a) No Control, (b) Non-linear, (c) Constant-rate, (d) Constant-time

pass and *slip* correspond to played and dropped/duplicated frames. For dropped frames, *pass* indicates the total number of frames evaluated, and *slip*, a positive integer, corresponds to the number to drop from *pass* passing frames. For duplicated frames, *pass* has the same interpretation, however, *slip* indicates the number of duplications of the last frame in the sequence. This formulation leads to a fast evaluation at playout time.

To set the control variables *pass* and *slip*, we define control functions which are dependent on the tolerable rate of drop and duplication for each medium. These functions can be instantaneous, introducing a large discontinuity in playout, or can be more gradual (Fig. 5). The control algorithm supports any type of selective drop or duplication function. We have initially investigated a constant rate-based correction function, but other functions can be supported as well.

3 Intermedia Synchronization Mechanism

In this section we describe our intermedia control mechanism for providing presentation-time synchronization and quality of service adjustment. The approach relies on an initialization phase, a retrieve/transmit process, a playout process, and a monitor/control process. We have formulated these components using an independent source–destination model with the goal of supporting either a distributed or a local data model. Each component will be described in the following subsections.

3.1 The Initialization Phase

In the course of a multimedia application's execution, the establishment of a session can be requested, requiring the activation of the delivery and playout subsystem. Typical scenarios

Table 1: Data Manipulated During Initialization

$object_A$	composite multimedia object
$tree_A$	object temporal representation
$\Pi = \{\pi_i\}$	ordered sequence of object playout times (deadlines)
$\Sigma = \{\sigma_i\}$	component object sizes (bits)
D_v, D_p, D_t	queuing, propagation, and transmission delays for packet of size S_m
S_m	packet size for medium m
C	channel capacity
T_n	control time for nth block
$P(late)$	requested fraction of late packets/blocks
$\Phi = \{\phi_i\}$	ordered sequence of retrieval/transmission times (deadlines)
$\Phi^c \subseteq \Phi$	set of aperiodic deadlines from Φ

for this include establishing a video conference or selecting a multimedia motion picture for presentation. Once selected, the session timing requirements must be interpreted for connection establishment and maintenance.

For session establishment, a statistical scheduling framework can be applied based on the characteristics of the requested session and the current system resources. The result is the generation of a set of deadlines which can be used by the retrieve/transmit process as local timing information. Included in this initialization phase are playout time identification [10]; data size characterization; bandwidth, delay and buffer reservation; computation of retrieval/transmission schedules; and initiation of data transfer. The data involved in this initialization phase are summarized in Table 1.

3.2 The Retrieval/Transmission Process

The retrieval/transmission process is perceived to be independent of the monitor and play-out/receive processes to support both the local and distributed data scenarios. It is therefore an asynchronous process, whereas the transfer of data from the buffer pool to the display mechanism can be synchronous. The **retrieval/transmission** process is outlined as follows. Initially, the multimedia object is characterized in the initialization phase and component object deadlines are determined (Π, Φ). After transfer initiation, data objects are sent to the destination based on their deadlines and sequence number.

1. while $i + j \leq 2m + 1$ do

(a) if $\phi_i \in \phi_i^c$ and $i \leq m$ then {aperiodic parts of schedule}

i. if $clock \geq \phi_i + start$ then

A. {send object i to destination with appended playout deadline π_i}

B. $i := i + 1$

(b) if ϕ_i not $\in \phi_j^c$ and $j \leq m$ then {periodic parts of schedule}

i. if $(clock \leq \pi_j + start \leq clock + T_E + start)$ then

A. {send object j to the destination with appended playout deadline π_j}

B. $j := j + 1$

The algorithm operates by evaluating the deadlines of objects awaiting transmission. The first conditional statement in the algorithm tests for impending retrieval deadlines for objects in the culled (Φ^c) retrieval schedule. The second conditional statement identifies the retrieval time of the remaining objects not in the culled schedule. The value of *start* reflects the variation of end-to-end delays for a session that uses multiple channels and allows each source to begin transmission synchronously. After initiation, the retrieve/transmit processes transfer data from sources to the destination with attached deadlines and sequence numbers for playout. At the receiver, the arrived data are queued based on deadlines and sequence numbers inside individual blocks. If no space is available to buffer arriving data frames, they are assumed to be discarded.

3.3 The Playout Process

The playout process must schedule the presentation of aperiodic events such as text and graphic displays and initiate the playout of segments of periodic streams of audio and video. Once established, the ordering of these segments relies on sequence numbers rather than individual deadlines associated with each frame. For example, a motion picture scene can have an initial playout initiation deadline, but its component frames can be played out with respect to this initial deadline and their individual sequence numbers. This mechanism provides less overhead than managing a deadline for each frame. Deadlines are always arranged to be monotonically increasing in time [10], and objects are assumed to arrive in sequence. The **playout** algorithm for managing presentation deadlines is outlined below.

1. if $clock \geq \pi_i + start$ then

(a) call $playout(object_i)$ {initiate playout of block}

2. for each stream k in $session$ do

   (a) if $clock = t_{play}$ then {time to play next frame}

      i. call $playout(rh_k)$ {play frame at head of queue}

      ii. if $slip < 0$ then {drop required}

      iii. elsif $slip > 0$ then {duplicate required}

      iv. else $slip = 0$ {nominal playout}

   (b) $t_{play} := t_{play} + \Delta_k$ {next playout time}

This algorithm performs two critical operations. First, the current time is evaluated against the set of scheduled deadlines. At the appropriate times, their playout is initiated. The second operation performs a similar action on periodic stream deadlines as indicated by sequence numbers. For each stream the current frame is played-out prior to the update of the queue head pointer based on the *slip* and *pass* control parameters set by the monitor and control process. The details of the pointer manipulation are not shown here.

Note that the buffer-to-playout transfer is synchronous. At the appropriate intervals for each medium (e.g., via interrupt), the current frame is evaluated for dropping or duplication. In either case, a frame is always passed to the presentation device for playout. In contrast, data arriving from the source are put in the receiving queue asynchronously.

3.4 The Monitor and Control Process

The monitor/control process serves three functions: monitoring queue and skew values for each stream, providing playout rate control, and providing source timing feedback to initiate time offset correction. The frequency of execution is dependent of the playout rates and block sizes of the individual presentation streams. An outline of the **monitor/control** algorithm is shown below.

1. for each stream k do

   (a) if $qlevel_k < qll_k$ then {low queue level}

      - $(pass, slip) := under(qlevel_k)$ {initiate stream lag}

   (b) elsif $qlevel_k > qlh_k$ then {high queue level}

      - $(pass, slip) := over(qlevel_k)$ {initiate stream lead}

(c) else {nominal queue level}

- $(pass, slip) := sync(qlevel_k, skew(k, l))$ {synchronize stream k to l}

2. {update queue level, skew, and lost frame statistics}

3. if $avg\_qlevel_k > threshold$ then

(a) send $offset$ to source

On each iteration, this algorithm invokes one of the control functions under(), over(), or sync() depending on the queue level. If it is high or low, the algorithm provides service degradation in the form of drops or duplications until service is restored. If the level is nominal, then intermedia synchronization control is applied. In all cases, the values of *pass* and *slip* are manipulated to control the playout rate and to effectively control skew.

4 Discussion

We have exercised our intermedia control mechanism with a number of sequences of simulated audio and video traffic. Correction functions have been selected to provide constant-rate skew correction and underflow/overflow prevention. The results indicate predicted skew correction and demonstrate the validity of our approach. The simulations have also revealed several areas for further study. These include the investigation of appropriate correction functions for each medium, and the development of a predictive solution to prevent underflow and overflow.

When a skew, underflow, or overflow correction function is performed, a resultant reduction in quality of service occurs. Our control mechanism attempts to distribute this degradation over an interval. In the initial simulations, constant-rate functions were used for under(), over(), and sync(). The range of acceptable correction rates depends on tolerated degradation in quality of service. We are currently exploring these ranges and alternate correction functions in relation to specified skew tolerances. Clearly there are differences in correction functions for each medium. Because video data have ample temporal redundancy, they are the target of congestion control schemes, being dropped when the delivery mechanism becomes saturated, and can be subsequently corrected by our scheme. Audio data are more perceptible when lost, yet can have periods of silence during which skew correction can be achieved without perceivable interruption.

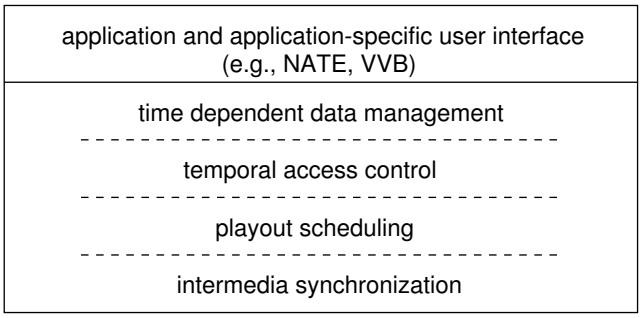

Figure 6: Layers of Timing Management

Another observation of our experiments is the need to monitor the rate of queue level change to better predict underflow and overflow. This can be achieved by maintaining additional queue statistics. Furthermore, large instantaneous skew transitions can be eliminated by monitoring frame losses at the time of arrival rather than at playout time.

5 Conclusion

We have proposed a mechanism to enable intermedia synchronization in a distributed multimedia information system. The mechanism is designed to operate independently of a source scheduling mechanism, providing a graceful skew correction for lost frames or other anomalous system behavior. Given suitable correction rate parameters, the algorithm can reconstruct a full-rate playout stream from a stream missing an arbitrary number of elements, as can result from congestion control or when limited bandwidth channels are used.

The intermedia synchronization mechanism fits into a larger system framework of managing time dependencies of multimedia data in a distributed multimedia information system (Fig. 6). We are currently incorporating the intermedia skew control system into into our Virtual Video Browser (VVB) application for content-based query and information display as applied to digital movies, and into our News at Eleven (NATE) application which facilitates multimedia information retrieval and presentation.

References

[1] Anderson, D.P., Homsy, G.: A continuous media I/O server and its synchronization mechanism. Computer **24** (1991) 51–57

[2] Bulterman, D.C.A., van Liere, R.: Multimedia synchronization and UNIX. Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video, Heidelberg, Germany (1991)

[3] Cambell, A., Coulson, G., Garcia, F., Hutchison, D.: A continuous media transport and orchestration service. Proc. SIGCOMM'92, Baltimore, MD (1992)

[4] Chao, H.J., Johnston, C.A.: A packet video system using the dynamic time division multiplexing technique. Proc. Globecom'86, Houston, TX (1986) 767–772

[5] Cochennec, J.Y., Adam, P., Houdoin, T.: Asynchronous time-division networks: terminal synchronization for video and sound signals. Proc. Globecom'85, New Orleans, LA (1985) 791–794

[6] De Prycker, M., Ryckebusch, M., Barri, P.: Terminal synchronization in asynchronous networks. Proc. ICC'87, Seattle, WA (1987) 800–807

[7] Ferrari, D., Verma, D.C.: A scheme for real-time channel establishment in wide-area networks. IEEE J. on Sel. Areas in Comm. **8** (1990) 368–379

[8] Gibbs, S., Dami, L., Tsichritzis, D.: An object-oriented framework for multimedia composition and synchronisation. Object Composition, Tech. Rept., University of Geneva (1991) 133–143

[9] Gilge, M., Gussella, R.: Motion video coding for packet-switching networks–an integrated approach. Proc. SPIE, Boston, MA (1991)

[10] Little, T.D.C., Ghafoor, A., Chen, C.Y.R.: Conceptual data models for time-dependent multimedia data. Proc. MMIS'92, Tempe, AZ (1992) 86–110

[11] Little, T.D.C., Ghafoor, A.: Scheduling of bandwidth-constrained multimedia traffic. Comp. Comm. **15** (1992) 381–387

[12] Milazzo, P.G.: Shared video under Unix. Proc. Usenix Conf., Nashville, TN (1991) 369–383

[13] Montgomery, W.A.: Techniques for packet voice synchronization. IEEE J. on Sel. Areas in Comm. **SAC-1** (1983) 1022–1028

[14] Nakajima, J., Yazaki, M., Matsumoto, H.: Multimedia/realtime extensions for the Mach operating system. Proc. Summer 1991 Usenix Conf., Nashville, TN, (1991) 183–198

[15] Northcutt, J.D., Kuerner, E.M.: System support for time-critical applications. Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video, Heidelberg, Germany, (1991)

[16] Pasieka, M., Crumley, P., Marks, A., Infortuna, A.: Distributed multimedia: how can the necessary data rates be supported. Proc. Usenix Conf., Nashville, TN, (1991) 169–182

[17] Rangan, P.V., Ramanathan, S., Vin, H.M., Kaeppner, T.: Media synchronization in distributed multimedia file systems. Proc. MM'92, Monterey, CA, (1992) 315–328

[18] Ravindran, K.: Real-time synchronization of multimedia data streams in high speed networks. Proc. MMIS'92, Tempe, AZ, (1992) 164–188

[19] Szabo, B.I., Wallace, G.K.: Design considerations for JPEG video and synchronized audio in a Unix workstation. Proc. Usenix Conf., Nashville, TN (1991) 353–368