Management of time-dependent multimedia data [1]

T.D.C. Little and J.F. Gibbon

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, jfg@bu.edu

MCL Technical Report 09-01-1992

Abstract–A number of approaches have been proposed for supporting high-bandwidth time-dependent multimedia data in a general purpose computing environment. Much of this work assumes the availability of ample resources such as CPU performance, bus, I/O, and communication bandwidth. However, many multimedia applications have large variations in instantaneous data presentation requirements (e.g., a dynamic range of order 100,000). By using a statistical scheduling approach these variations are effectively smoothed and therefore more applications are made viable. The result is a more efficient use of available bandwidth and the enabling of applications that have large short-term bandwidth requirements such as simultaneous video and still image retrieval. Statistical scheduling of multimedia traffic relies on accurate characterization or guarantee of channel bandwidth and delay. If guaranteed channel characteristics are not upheld due to spurious channel overload, buffer overflow and underflow can occur at the destination. The result is the loss of established source–destination synchronization and the introduction of intermedia skew.

In this paper we present an overview of a proposed synchronization mechanism to limit the effects of such anomalous behavior. The proposed mechanism monitors buffer levels to detect impending low and high levels on frame basis and regulates the destination playout rate. Intermedia skew is controlled by a similar control algorithm. This mechanism is used in conjunction with a statistical source scheduling approach to provide an overall multimedia transmission and resynchronization system supporting graceful service degradation.

Keywords: Multimedia synchronization, scheduling.

[1] In *Proc. SPIE Symposium OE/FIBERS'92, (Enabling Technologies for Multi-Media, Multi-Service Networks)*, Boston, Massachusetts, September 1992, SPIE Vol. 1785, pp. 110-121. This material is based upon work supported in part by the National Science Foundation under Grant No. IRI-9211165.

1 Introduction

Multimedia as a technology can now be interpreted to describe computer systems supporting audio and video as data types. Characteristic of these data types is the need for timely data retrieval and delivery to the user. A multimedia computer system must overcome any system delays caused by storage, communication, and computational latencies in the procurement of data for the user. These latencies are usually random in nature, being caused by shared access to common system resources such as networks, storage devices and system buses. In a distributed multimedia information system (DMIS), multiple data sources including databases, video cameras, and telephones can be connected to user workstations via high-speed packet-switched networks. System latencies in a DMIS are problematic because several streams originating from independent sources can require synchronization to each other in spite of the asynchronous nature of the network.

To support the presentation of time-dependent data, scheduling disciplines associated with real-time operating systems are necessary. These data are either generated in real-time or are assumed to be retrieved and transmitted in real-time; coming from storage at a rate approximately equal to consumption at the receiver. However, the real-time requirements of multimedia data can be relaxed because data delivery can tolerate some occasional lateness, i.e., catastrophic results do not occur when data are not delivered on time. The design of a system to support time-dependent media must account for latencies in each system component used in the delivery of data, from its source to its destination. Therefore, specific scheduling is required for storage devices, the CPU, and communications resources.

Recent work supporting time-dependent data is for live data communications, data storage systems, and general distributed systems[8]. For data communications, live, periodic data sources such as packet audio and video are considered, and the capacity of the communication channel is assumed to not be exceeded. These configurations typically use a single connection because there is seldom any need to deliver synchronous data streams from multiple independent live sources. Packet delay variations introduced by queuing in the channel are limited at the receiver by buffering. Our view of an analogous system supporting multimedia sessions is illustrated in Fig. 1. In this model, data are transmitted from multiple independent sources and are buffered at the receiver prior to delivery to the appropriate presentation subsystems.

One of the difficulties in supporting multimedia sessions is the synchronization of related media in what we call *intermedia* synchronization. The lip-sync of audio and video is typical,

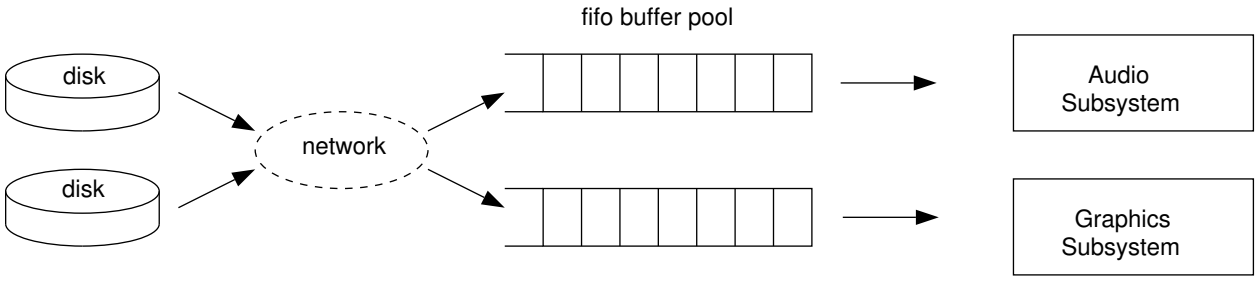

Figure 1: Data retrieval from multiple independent sources and multimedia playout

but synchronization to both real-time and to other streams 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[15]. However, it can only be applied to a single storage subsystem or communication channel. Furthermore, the interleaving process increases overhead, especially for compressed data formats, and requires that all media have the same quality of service[1]. Multicast channels introduce similar synchronization problems. Earlier techniques provide synchronization by allowing the destination to track the source for packet audio or video communications[4,12]. We generalize the single-medium model to multimedia sessions, allowing intermedia synchronization among multiple sources in a multidrop fashion. Furthermore, we extend previous work[8] by providing a mechanism to gracefully degrade time-dependent presentation during periods of resource overload and channel data loss due to network dropping.

Our approach is to monitor and control queue size and intermedia skew introduced after session establishment by 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 in any real system. Furthermore, 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.

Feedback can also be used to change the characteristics of the source though rate-based transmission control[5] and variable source resolution control[7,14]. Both of these approaches

address changing channel loading scenarios rather than specific playout synchronization. For playout synchronization, the use of feedback has been applied to audio and video delivery at the bit[2] and packet levels[3,4,13]. A digital phase-locked loop (DPLL) can track either the stream of incoming bits or packets to derive the source clock frequency for playout synchronization. However, for compressed, variable bit-rate (VBR) data formats, the bit synchronization does not apply. Neither of these schemes can deal with multiple sources because they cannot track more than one source. Furthermore, there is no provision for dealing with a fixed data presentation rate at the destination which prevents control of playout rate. We propose the use of an intermedia synchronization mechanism at the time of playout by using a frame-based control loop instead of a bit or packet-based loop. Playout times are assumed to be fixed at a constant value, with playout rate controlled by selective duplication and dropping. This proposal differs from other approaches that assume a variable playout rate necessary for the destination to track the source.

In the remainder of this paper we describe a system for managing synchronization for multiple streams of real-time multimedia traffic. In Section 2 we define the characteristics of intermedia synchronization and the framework for its control. In Section 3 we outline an implementation of the control mechanisms. Section 4 concludes the paper.

2 Characterization and Control of Intermedia Synchronization

Synchronization is critical for the operation of any multimedia presentation environment. General-purpose computer systems do not have the dedicated hardware data paths found in video cassette recorders that provide synchronous audio and video presentation. Instead, storage devices, the network, the system bus, and dynamic memory must be carefully reserved and scheduled to provide similar but more flexible functionality. At each component and at different levels of system abstraction the synchronization requirements differ in character and control. In this section we establish parameters affecting various levels and propose a model for controlling intermedia synchronization.

2.1 Characterization of Synchronization

The synchronization of multimedia traffic streams need not be viewed as absolute. We can define a quality of service (QOS) associated with intermedia synchronization. For ex-

ample, a synchronization tolerance defines the bounds on the differences in desired versus actual playout times for continuous data streams. QOS parameters can characterize performance characteristics at any level in the system including end-to-end delays, startup times, dropouts, packet error rate, etc. The values of the QOS parameters depend on each medium and are interpreted in different ways by system components and the user.

With a QOS framework, the needs of both the user and service provider can be managed. For the user, QOS parameters ensure acceptable service over a session by control of intermedia synchronization. For the service provider, resources can be allocated based on QOS to support many sessions. Ideally, QOS provides a relationship between cost and quality. However, many variables affect QOS for intermedia synchronization. We seek a smooth cost vs. quality parameter to allow graceful degradation of service or service change to support different system loadings. This would support graceful service degradation in response to load changes (for individual connections), and would support resource allocation decisions at the operating system level.

We define timing parameters characterizing 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. These parameters can describe time skew with respect to real-time as well as to other periodic streams in an manner analogous to a phase angle.

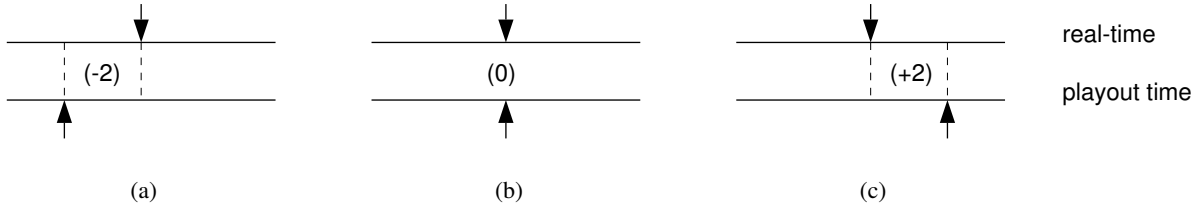

Figure 2: Skew: (a) lagging, (b) none, (c) leading

Synchronization implies the occurrence of multiple events at the same instant in time. If they occur at different instants they are *skewed* in time (Fig. 2). For two sequences of events (e.g., sequences of audio and video data frames) individual differences between corresponding events are called *jitter*, whereas the average difference over some interval of

n frames is called *skew*.

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*. By dropping or duplicating frames we are able to control the rate of playback assuming a constant playout interval between frame playouts. We can *monitor* skew of a periodic stream by keeping track of the number of dropped, lost, and duplicated frames. We can *control* skew by forcing frame drops or duplications. Initiation of frame dropouts is permissible for audio and video which have substantial short-term temporal data redundancy. For text and graphics, this is not true, however, their tolerance to delay is greater.

In our proposed mechanism we monitor skew at the time of playout when delay variations have the most impact on the user. We therefore are interested in the skew of the most recently presented frame, assuming a fixed playout rate (e.g., 30 frames/s for video). 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 stream. Because many streams are possible, we characterize both intermedia and real-time reference skew for k streams using a matrix representation as,

$$skew = \begin{bmatrix} 0 & sk_{1,2} & sk_{1,3} & sk_{1,k+1} \\ sk_{2,1} & 0 & sk_{2,3} & sk_{2,k+1} \\ sk_{3,1} & & 0 & sk_{3,k+1} \\ \\ sk_{k+1,1} & sk_{k+1,2} & sk_{k+1,3} & 0 \end{bmatrix}$$

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 a target skew matrix $tsk_{p,q}$ (analogous to Ravindran's divergence vector[16]) which indicates target values which can be interpreted by a skew control function. Related to skew is data *utilization*[9]. 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 stream 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

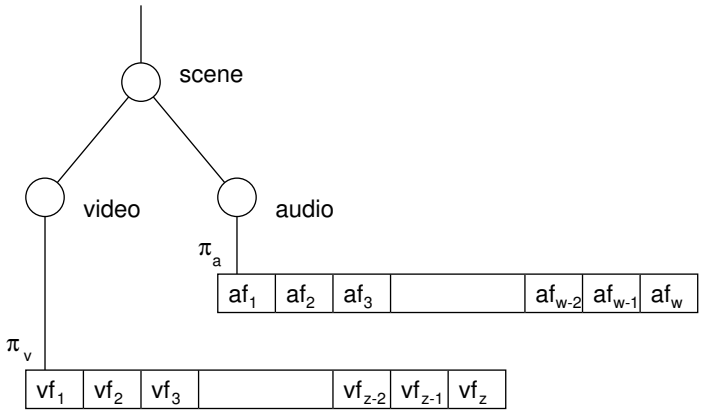

Figure 3: Blocking for continuous media

media into *blocks* to permit efficient storage and manipulation[10]. With this decomposition, blocks associate a single start deadline with a sequence of periodic stream 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 Control of Intermedia Synchronization

Intermedia synchronization can be approached several ways. One approach is to set up an open-loop path from source to destination and allow the destination to track the source clock. Buffering provides accommodation of delay jitter, and source tracking handles changes to long-term average delay. When multiple independent sources exist, this approach cannot work in the presence of clock variations. The destination must lose synchronization or take control of the source delivery rate via feedback. Furthermore, initiation of data transfer for multiple sources can cause additional skew between streams as initiation messages can traverse different network paths. At the destination, intermedia synchronization relies on the availability of data to playout. Maintaining a constant skew between streams assumes the presence of nominal levels of queued data frames. If short-term load anomalies occur then graceful degradation of service must be provided. To provide consistent service at the source, a statistical scheduling approach can be used to schedule the times when data must be retrieved and then transmitted to the destination. Subsequently, spurious load changes can be accommodated by buffering and service degradation using our proposed control algorithm. In summary, a statistical scheduler manages transmission times with respect to the source whereas the destination provides buffering for delay variations and service degradation for

transient load changes.

2.2.1 Statistical Source Pacing Control

For live sources such as video cameras, the destination has no control over the times when packets are generated and transmitted. The destination can only control the source clock (when not multicast to other destinations) or the end-to-end latency, or control time, between frame generation and playout. This latter approach has limited utility in conversational services such as videotelephony due to user intolerance to large end-to-end delay. For stored-data sources, the rate at which data are put on the channel can be fully controlled subject to available or reserved bandwidth constraints. We therefore present a synopsis of our statistical transmission scheduler for handling these two cases[11].

The approach uses *a priori* knowledge of data traffic characteristics to facilitate scheduling, when available. Otherwise, the statistical nature of the live multimedia traffic source is anticipated through existing statistical methods. Essentially, the dynamic bandwidth requirements of a multimedia object are fit into finite resources of delay and channel capacity rather than the resources dictating the feasibility of the application. The result is that delays are traded-off for the satisfaction of a playout schedule, even when the capacity of the channel is exceeded by the application.

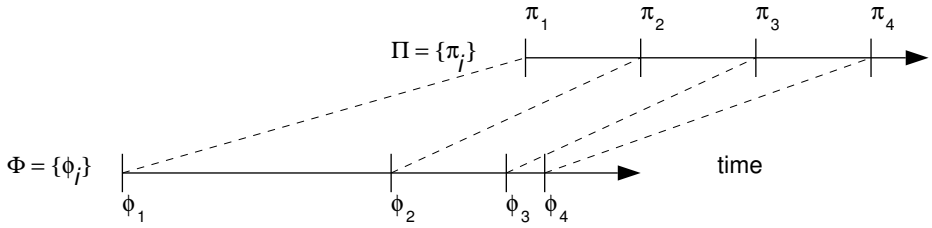

Figure 4: Example relation between playout and retrieval schedules

Depending on their size and playout times, transmission of a sequence of objects is either back-to-back or introduces slack time. We define an optimal schedule for a set to be one that minimizes the object's control times. Two constraints determine the minimum control time for a set of objects. These are the minimum end-to-end delay (MD constraint) per object, and the finite capacity (FC constraint) of the channel. The MD constraint simply states that an object cannot be played-out before arrival. This constraint must always be met. The FC constraint describes the relation between a retrieval time and its successor when the channel

is busy and accounts for the interarrival time due to transit time and variable delays. It also represents the minimum retrieval time between successive objects.

Given the characteristics of the channel and of a composite multimedia object ($D_v, D_p, D_t, C, S_m, \pi_i, \sigma_i, P$ corresponding to variable, fixed, and channel transmission delays, channel capacity, packet size, playout deadline, object size, and probability of lateness), a schedule is constructed[11]. Construction begins by establishing an optimal retrieval time for the final, nth object, i.e., $\phi_n = \pi_n - T_n$. The remainder of the schedule can be determined by iterative application of the MD and FC constraints for adjacent objects. The resultant schedule indicates the times to put objects onto the channel between the source and destination (e.g., Fig. 4), and can be used to establish the worst case buffering requirement. During slack periods objects are scheduled based on playout time and object size. During busy periods objects are scheduled based on channel availability as well.

Feedback from the destination to the source can provide a course adjustment to the precomputed data delivery schedule to eliminate initiation skew or long-term changes to the expected delay distribution. This is achieved by adding an offset to each data stream by passing this skew parameter to the retrieval/transmit scheduler (i.e., $\phi_i' = \phi_i + offset$). The delay can then be adjusted on an ongoing basis by feedback using the measured long-term queue level and playout skew. This is somewhat coarse because the schedule has been statically precomputed.

2.2.2 Buffer Level and Skew Control

By using the statistical source pacing control, 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 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.

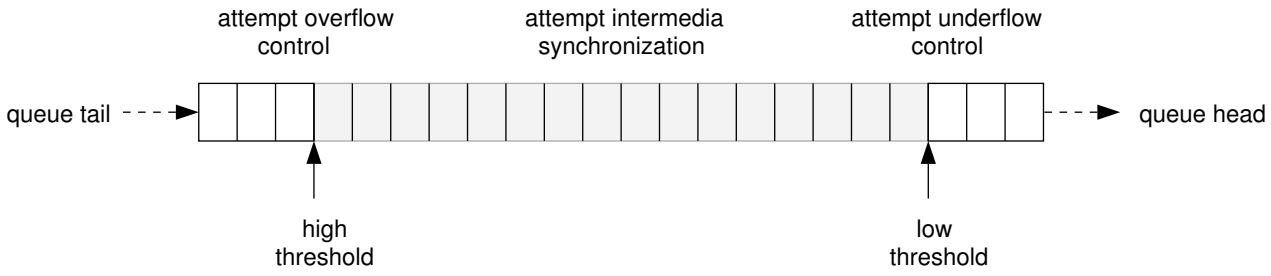

Figure 5: Three queue states

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. 5). 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. 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 loop that monitors the queue level and playout skew. Control is provided by changing the playout rate (or utilization) using frame drop and duplication. Oscillation of the system is prevented 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. However, because the drop and duplicate parameters rely on integral values, utilization takes on

discrete values. Duplication causes $U > 1$ while dropping (loss of frames) causes $U < 1$. Utilization over an interval comprised of n frames can be determined using the formula, $U = pass + slip/pass$, where $pass$ 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.

3 Intermedia Synchronization Mechanism

In this section we describe our overall system for managing real-time multimedia presentation and dynamic quality of service adjustment. Our approach relies on an initialization phase, a retrieve/transmit process, a playout process, and a monitor/control process. These processes are illustrated schematically in Fig. 6 where the source and destination are assumed to operate asynchronously. 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.

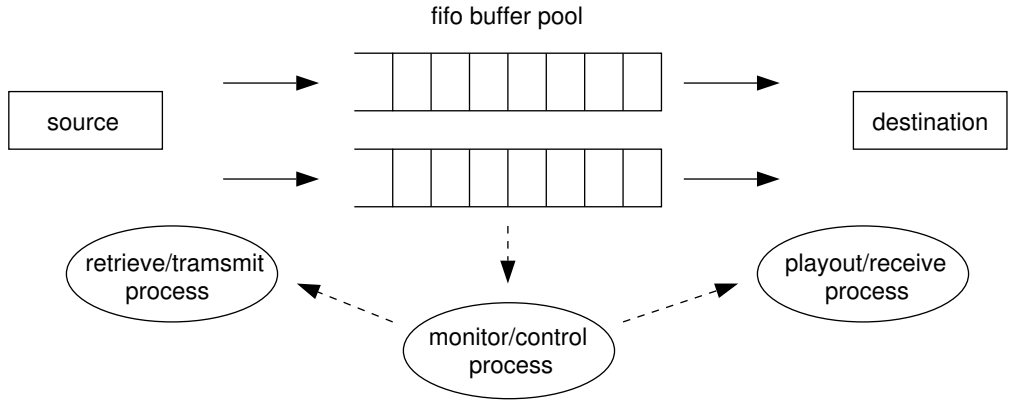

Figure 6: Processes involved in delivery of time-dependent data

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

for this include establishing a videoconference or selecting a multimedia motion picture for presentation[6,9]. Once selected, the session timing requirements must be interpreted for connection establishment and maintenance.

For session establishment, the statistical scheduling framework described in Section 2 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 a retrieval/transmission schedule, and initiation of data transfer as applied to either live or stored data sources. The data involved in this initialization phase are summarized in Table 1.

Table 1: Data manipulated in the initialization phase

$object_A$	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

3.2 The Retrieval/Transmission Process

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[5]. However, feedback approaches alone cannot provide consistent presentation quality under wide bandwidth variations due to network delays or multimedia object sizes.

A priori knowledge of the data characteristics can resolve this difficulty as typified by the approach described in Section 2.

The retrieval/transmission process is perceived to be independent of the remaining monitor and playout/receive processes to support both the local and distributed cases of data retrieval (from local or remote sources). 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. Feedback from the destination to its source can change the time offset of our static retrieval schedule. Data retrieval and transfer is provided by the following **retrieval/transmission** algorithm outlined below.

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) else
      i. $i := i + 1$
   (c) 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. $j := j + 1$
   (d) else
      i. $j := j + 1$

The algorithm operates by iterating on the set of objects waiting for transmission, assuming initially that no objects are sent. 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. *Start* is determined using $start_j = clock + T_o - TE_j$, where T_o is the aggregate control time among the channels of the session ($T_o = max(\{TE_j, \forall j\})$ and TE_j is the control time for the jth channel). After initiation, the retrieve/transmit processes transfer data from sources to the destination with attached deadlines and sequence numbers for playout. At the receiver (destination), 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.

Feedback can also be used to correct initial skew introduced at startup time. This process is achieved by subtracting a time offset from the source clock, i.e., $clock := clock - offset$. The offset can be sent as a control message from the destination to the source.

3.3 The Playout Process

The playout must schedule the presentation of aperiodic events such as text and graphic display and initiate the playout of segments of periodic streams of audio and video. Once established, sequencing 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}
   (b) mark $object_i$ used
   (c) $i := i + 1$
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}

- move queue head to cause frame drop

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

- move queue head to cause frame duplication

iv. else $\{slip = 0\}$

- move queue head to next frame

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

This algorithm performs two iterative loops. The first loop evaluates the current time against the set of scheduled deadlines. At the appropriate times, it initiates their playout. The second loop performs a similar function on periodic stream deadlines as indicated by sequence numbers. For each steam 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 process 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 through utilization U, 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 where the control variables are summarized in Table 2.

1. for each stream k do

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

- $(pass, slip) := under(qlevel_k)$ {set new control values to initiate stream lag}

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

- $(pass, slip) := over(qlevel_k)$ {set new control values to initiate stream lead}

(c) else {nominal queue level}

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

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

3. if $avg\_qlevel_k > threshold$ then

(a) send $offset$ to source

Table 2: Control variables for stream k

$qlevel_k$	current queue level in objects/blocks
qll_k	queue low level threshold
qlh_k	queue high level threshold
U_k	playout utilization (1 = nominal)
$pass_k$	number of frames to evaluate for drop/duplicate
$skip_k$	number of frames to drop or duplicate
$offset_k$	feedback signal to source to control delivery rate
$sk_{k,l}$	skew from stream k to l
$tsk_{k,l}$	target skew from stream k to l
$ring_k$	ring buffer (queue)
rh_k	ring buffer head pointer
rt_k	ring buffer tail pointer
$seq_{k,i}$	sequence number of frame i
Δ_k	frame playout period for medium k

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 duplicates until service is restored. If the level is nominal, then intermedia synchronization control is applied. In all cases, the values of *pass* and *skip* are manipulated to control the utilization at playout time and to effectively control the playout rate.

3.5 Data Structures for Scheduling

Critical for the performance of our scheduling mechanism is the maintenance of timing information to facilitate rapid playout scheduling and data flow. In addition to maintaining session information, the system must keep track of arrived, dropped, and processed frames. Because data flow though the system dynamically, a major concern is the management of

buffer space. To provide the FIFO buffering, an appropriate data structure is required to allocate and deallocate buffer space for the receiver and the playout processes.

A reasonable choice for a FIFO queue mechanism is a ring buffer[17]. This logical structure allows data to be written and read in FIFO fashion and also provides for memory reuse. As data are added to slots in the ring, obsolete elements are overwritten and reused. When the capacity of the ring is exceeded, the newest element can overwrite the oldest unused item providing a data dropping mechanism. However, because we desire graceful degradation when the buffer is at capacity, we want instead to selectively drop (or duplicate) elements from the middle of the buffer. For variable length frames, the removal of items from the middle of the ring introduces memory fragmentation and a significant free-space allocation problem. In our scheme, frame drop and duplication is always performed at the output of the buffer rather than the middle, therefore, the ring buffer is sufficient for our purposes with the stipulation that a full buffer cannot be overwritten. For a fixed size ring, variable length buffer cells cause a variable ring capacity in number of frames. This complicates the control paradigm somewhat because it assumes a fixed queue capacity measured in frames.

4 Conclusion

We have proposed a mechanism to enable intermedia synchronization in a distributed multimedia information system. Its interesting feature is the ability to perform graceful service degradation under anomalous system behavior. The proposed mechanism is not without deficiencies. Our scheduling approach does not address many of the real-time scheduling issues associated with operating system task management, i.e., we have assumed sufficient CPU resources can be obtained by the scheduler to perform its job. In the presence of multiple users, sessions, and other background tasks, this assumption can be invalidated. Furthermore, our present statistical scheduling algorithm is unable to to dynamically adjust to channel loading changes. We are currently addressing these issues.

The synchronization mechanism fits into a larger system framework of managing time dependencies of multimedia data in a distributed multimedia information system (Fig. 7). A DMIS must provide services for creation, modification, selection, retrieval, and presentation for a diverse set of applications. We intend to incorporate our intermedia synchronization mechanism into our virtual video browser (VVB) application[6] 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.

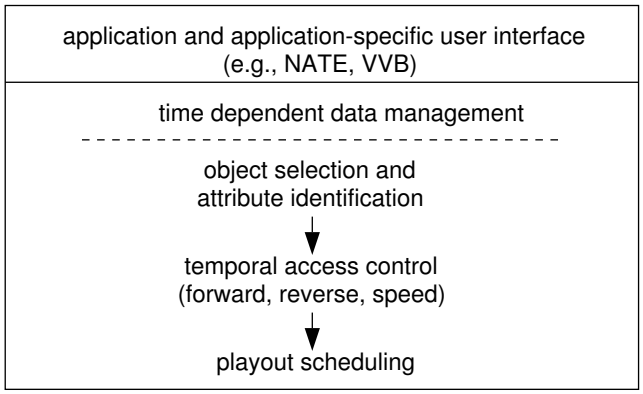

Figure 7: Application and time management views

5 Acknowledgments

This material is based upon work supported in part by the National Science Foundation under Grant No. IRI-9211165. We would also like to thank Frank Kao for his input into the development of the feedback mechanism.

6 References

1. A. Cambell, G. Coulson, F. Garcia and D. Hutchison, "A continuous media transport and orchestration service," *SIGCOMM'92* Baltimore, Maryland, August 1992.

2. H.J. Chao and C.A. Johnston, "A packet video system using the dynamic time division multiplexing technique," *Globecom '86 (IEEE Global Telecommunications Conference Record 1986)*, pp. 767-772, Houston, TX, December 1986.

3. J.Y. Cochennec, P. Adam, and T. Houdoin, "Asynchronous time-division networks: terminal synchronization for video and sound signals," *Globecom '85 (IEEE Global Telecommunications Conference Record 1985)*, pp. 791-794, New Orleans, LA, December 1985.

4. M. De Prycker, M. Ryckebusch, and P. Barri, "Terminal synchronization in asynchronous networks," *Proc. ICC '87* (IEEE Intl. Conf. on Communications '87), pp. 800-807, Seattle, WA, June 1987.

5. D. Ferrari and D.C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, Vol. 8, No. 3, pp. 368-379, April 1990.

6. R.J. Folz, J.F. Gibbon, T.D.C. Little, F.W. Reeve, and D.H. Schelleng, "A digital video-on-demand database supporting content-based queries," working paper, Multimedia Communications Laboratory, Boston University.

7. M. Gilge and R. Gussella, "Motion video coding for packet-switching networks–an integrated approach," *Proc. SPIE Conf. on Visual Communications and Image Processing*, Boston, MA, November 1991.

8. T.D.C. Little and A. Ghafoor,"Multimedia synchronization protocols for broadband integrated services," *IEEE Journal on Selected Areas in Communications* (Special Issue: Architectures and Protocols for Integrated Broadband Switching), Vol. 9, No. 9, pp. 1368-1382, December 1991.

9. T.D.C. Little and A. Ghafoor,"Distributed multimedia object management and composition," *IEEE Network* (Special Issue: Distributed Applications for Communications), Vol. 4, No. 6, pp. 32-49, November 1990.

10. T.D.C. Little, A. Ghafoor, and C.Y.R. Chen,"Conceptual data models for time-dependent multimedia data," *Proc. 1992 Workshop on Multimedia Information Systems (MMIS '92)*, pp. 86-110, Tempe, AZ, February 1992.

11. T.D.C. Little and A. Ghafoor,"Scheduling of bandwidth-constrained multimedia traffic," *Computer Communications* (Special Issue: Multimedia Communications), Vol. 15, No. 5, pp. 381-387, July/August 1992.

12. W.A. Montgomery, "Techniques for packet voice synchronization" *IEEE Journal on Selected Areas in Communications*, Vol. SAC-1, No. 6, pp. 1022-1028, December 1983.

13. J.D. Northcutt and E.M., Kuerner, "System support for time-critical applications," *Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

14. A. Park and P. English, "A variable rate strategy for retrieving audio data from secondary storage," *Proc. 1st Intl. Conf. on Multimedia Information Systems '91*, pp. 135-146, Singapore, January 1991.

15. P.V. Rangan and H.M. Vin, "Designing file systems for digital video and audio," *Proc. 13th Symp. on Operating Systems Principles (SOSP'91), Operating Systems Review*, Vol 25, No. 5, pp. 81-94, October 1991.

16. K. Ravindran, "Real-time synchronization of multimedia data streams in high speed networks," *Proc. 1992 Workshop on Multimedia Information Systems*, pp. 164-188, Tempe, Arizona, February, 1992.

17. L.C. Wolf, "A runtime environment for multimedia communications," *Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.