A Framework for Supporting Quality-Based Presentation of Continuous Multimedia Streams*

Thomas V Johnson and Aidong Zhang
226 Bell Hall
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260

Abstract

In this paper, we investigate a framework for supporting the continuous and synchronized presentation of multimedia streams in a distributed multimedia system. Assuming that the network transmission and the server provide sufficient support for delivering media objects, important issues regarding the enforcement of smooth presentations of multimedia streams must be addressed at client sites. We develop various presentation scheduling algorithms that are adaptable to quality-of-service parameters. The significance of the proposed algorithms is to adjust the presentation from its insynchrony by gradually catching up or slowing down rather than using skipping/pausing in an abrupt fashion. Experimental analysis conducted for the proposed approaches demonstrates that our algorithms can avoid hiccups in presentations. This framework can be readily used to facilitate various applications, including education and training.

1 Introduction

In a multimedia presentation system, multimedia data are usually organized into media objects and object classes, thus providing users with a clear-cut method for data specification. In order to re-present the original data stream to users, synchronization constraints among media objects must be specified and maintained. Moreover, if a composite stream is composed of media objects from different media streams, additional complications may arise with the timing relationships that may exist among the different types of media streams. Such media streams may not be merged prior to storage in a database as such a merger will vastly compound the difficulties of retrieving component media. Thus, the synchronization of multiple media streams becomes an essential prerequisite to any successful multimedia presentation application [LG90b, Ste90]. The results of this research will be supportive for many applications, including education and training [Fox95a, Fox95b]. The following example demonstrates an application.

*This research is supported by NSF under grant IRI-9632394.

Example 1 *Consider an application involving on-line computer-assisted learning in undergraduate education. Without loss of generality, we assume that there are two media streams, audio and slides (or video). Synchronization within the slide stream may require that two objects either overlap or be sequentialized. Synchronization within the audio stream requires only that objects be sequentialized. Additional Synchronization requirements between the two media streams are specified among slides and audio objects. In order to successfully deliver both streams to a student, the system must ensure that all time constraints placed on the individual delivery operations and the synchronization between slides and audio objects are preserved within a satisfiable scope. Deviations between slides and audio objects affects greatly the quality of the presentation of the streams.*

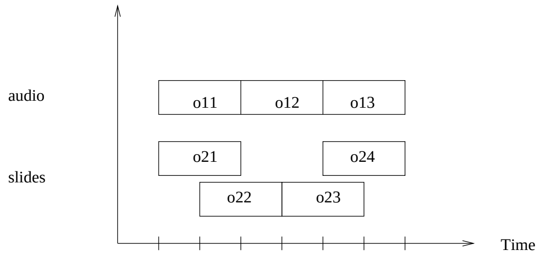


Figure 1: A presentation of streams.

1.1 Related Work

Several books with respect to multimedia have addressed the issue of synchronization as one of the most important research issues in multimedia research [Buf94, Fur96b, Fur96a, SN95]. Studies have investigated the modeling of the synchronization aspects of multimedia data from a conceptual perspective, including graphical models, Petri-Net based models, object-oriented models, and temporal abstraction models [Tom89, DC91, LG90b, SF89, Mas91, GBT94]. These models provide illustrative vehicles for specifying synchronization semantics at the application level. Importance of formulating quality-of-service (QoS) metrics for continuity and synchronization specifications in media streams has also been realized [VKvBG95, AFKN95, NS95, SWM95, WS96, ZG96]. Given these specification approaches, it is important to realize the appropriate QoS parameters for the presentation of various media streams and their synchronization in systems.

Substantial research has been directed toward the support of synchronization within operating systems and network architectures [RV93, Ste90, AH91, RRK93, GR93, ZF93, GVKR95]. Through this research, new behavioral concepts required for multimedia data have been identified and mechanisms have been proposed to enhance such conventional storage, synchronization, and communication mechanisms as random disk allocation, semaphores, monitors, or RPC. A significant amount of work has been contributed to server design, in particular, service time scheduling and admission control. Various QoS parameters have been incorporated into the algorithms in determining media data missing, skipping, and pauses. A significant amount of work has also been contributed to the network scheduling of media stream transmission. Various QoS parameters have also been incorporated into the consideration of network traffic. Different communication protocols have been proposed to support deterministic, probabilistic, or best effort transmission of media streams over network.

In a distributed or networked environment, approaches are also needed for the client sites to enforce synchronous presentation of multiple media streams. Research involving the synchronized presentation of multimedia data on this aspect has been started [Gha95, TK95a, TK95b]. Its importance and new issues have been realized in related workshops [NWS96, SUB96]. Chaudhuri et al [CGS95] have investigated the problem of continuously displaying composite objects that are dynamically specified. Courtiat et al [COC94] have proposed a conditional delivery mechanism to schedule media objects based on their synchronization requirements. Thimm and Klas [TK95b] introduced a rule-based scheduling approach to generate adaptive multimedia presentations. Following these efforts, advanced scheduling approaches need be developed to support elegant presentations of multimedia streams at client sites.

1.2 Our Contributions

In this paper, we will investigate a framework for supporting the continuous and synchronized presentation of multimedia streams at client sites in a distributed multimedia system. Assuming that the network transmission and the server provide sufficient support for delivering media objects, important issues regarding the enforcement of smooth presentations of multimedia streams must be addressed at client sites. We will develop various presentation scheduling algorithms that are adaptable to various QoS parameters. In case of delay, the recovery of the presentation from its insynchrony is performed by gradually catching up or slowing down rather than using skipping/pausing in an abrupt fashion. Experimental experience has demonstrated that skipping/pausing in an abrupt fashion could cause hiccups in presentations. This behavior would affect the effectiveness of the presentation dramatically, especially for supporting those presentations involving lip synchronization. Experimental analysis conducted for the proposed approaches below has shown that our algorithms can avoid hiccups in presentations. This framework can be readily used to facilitate various education and training applications, such as student training, human resource development, and asynchronous distance learning.

The remainder of this paper is organized as follows. Section 2 discusses the system and data models. In Section 3, we discuss the synchronization of multimedia presentations. In Section 4, we present the scheduling algorithms to ensure the continuous presentation of individual media streams. In Section 5, we present the scheduling algorithms to ensure the synchronous presentation of multiple streams. Section 6 discusses the algorithm performance. Concluding remarks are offered in Section 7.

2 The NetMedia System

2.1 The System Architecture

The system architecture (named NetMedia) was proposed in [GZ96b]. This architecture includes a set of NetMedia-servers that is distributedly superimposed upon top of a set of database (or file) management systems (DBMSs), a set of multimedia databases (or files), and a set of NetMedia-clients which support client access to each NetMedia-server. Each DBMS or file system manages the insertion, deletion, and update of the media data stored in the local database (or file). We assume that the clients accessing NetMedia system through the global NetMedia-client interface can only request multimedia data retrieval. Thus, no global updates on media data located in local media databases will be allowed.

Adaptability is incorporated into system design at both client and server ends. It is realized by dynamically adjusting system state to best service client requests as specified in the QoS requirements. To guarantee the continuity of multimedia data delivery over a network, we have investigated a two-phase buffer model and management. Buffer management is needed at both client and server sites to ensure that the retrieval (or transmission) of media streams will not cause the hiccups of their presentation. Buffer management at client sites will ensure that transmission of media streams from the servers will not cause hiccups in their presentation. Buffer management at server sites will ensure that the retrieval of media streams from disk to memory will not cause delays in their transmission. The details on this aspect have bee discussed in [GZ96a]. We will not discuss it further in this paper.

The NetMedia-client is responsible for synchronizing images, audio, and video packets received from the network and delivering them to the output devices on the client workstation. Thus, skipping or dropping of data to maintain presentation synchronization must be supported by the NetMedia-client. In addition, the NetMedia-client will request the server to vary the rate at which data are sent if changes are needed in the presentation schedule.

The NetMeida-server supports the multi-user aspect of media data caching and scheduling. It maintains real-time retrieval of media data from the multimedia database and transfers the data to the client sites through network. Skipping or dropping of data also need to be supported to avoid transferring useless data to clients. The interpretation of the media data is delegated to workstations at the client locations. Server functions also include admission control. Dynamic client requests of changes in the data transfer rates are supported.

In this paper, we will assume that the given transmission and the server provide sufficient support for delivering media objects. We will focus on the design of adaptable presentation scheduling at the client sites.

2.2 Data and Presentations

A media stream can be viewed abstractly at several levels. At the lowest level, a media stream is viewed as an unstructured *BLOB* (binary large objects) which can be further categorized within several higher-level object classes [Mas91, EM94, CAF$^+$91, GBT94]. Objects from different media streams may also be spatio-temporally combined into multimedia objects. Several conceptual data models which follow this general scheme have been proposed. We assume that each media stream is broken into a set of atomic objects. Each atomic object represents a minimum chunk of the media stream that bears some semantic meaning. Atomic objects in different media streams may have different internal structures. For example, a continuous video stream can be segmented into a set of atomic objects, each of which contains a set of video frames with specific semantic meaning. Similarly, a continuous audio stream can be segmented into a set of atomic objects, each of which contains a set of audio samples with specific semantic meaning. Higher levels of object classification are then formulated on atomic objects. Each atomic object is associated with a relative start time and a time interval which specifies the duration of its retrieval, with the initial atomic objects in the media stream assumed to start at time zero. The actual start time of a media object is usually dynamically determined. Once a media stream is invoked, it is associated with an actual *start time*; each media object within that stream will similarly be associated with an actual start time.

The atomic objects within a media stream are linked together through *intra-stream* constraints. These constraints may specify discrete, continuous, overlapping, or step-wise constant time flow relationships among the atomic objects. For example, some multimedia streams, such as audio and video, are continuous in nature, in that they flow across time; other streams, such as slide presentations and animation, have discrete, overlapping, or step-wise time constraints. It may, for example, be necessary to display two distinct slide objects jointly within a single slide presentation stream. In general, the temporal relationship between two atomic objects in a single stream may conform to any of the thirteen temporal relationships described in [All83]. Media objects from different streams may need to be linked through time constraints to specify their synchronization; such time constraints are termed *inter-stream* constraints. The temporal relationship between two atomic objects from different media streams may also conform to any of the thirteen temporal relationships described in [All83].

A *multimedia presentation* consists of a set of media streams upon which synchronization constraints are specified on the delivery operations to enforce both intra- and inter-stream synchronization. A delivery operation is performed on a media object.

3 Presentation Scheduling

3.1 QoS Requirements

We will now discuss QoS parameters and the effect of these parameters in the scheduling of multimedia presentations. The scheduling of a multimedia presentation must ensure the synchronized presentation among multiple media streams. We define *rendition rate* to be the instantaneous rate of presentation of a media stream. When there are no delays, the rendition rate for each stream would be equal to the nominal presentation rate. However, when there are delays, the rendition rate would be slower. A correctness criterion in this context must verify that delivery operations are performed according to a predefined synchronization pace and within the time constraints imposed on presentations. Since the correctness of time-based presentations depends on the accuracy of timing that must be maintained on media streams, the execution of a multimedia presentation is a question of quality rather than consistency. We must thus formulate new correctness criteria for the executions of multimedia presentations which define acceptable quality in real-time. Several important QoS parameters must be considered in these correctness criteria.

Different QoS parameters are defined at various layers in the system, including application, network, and system [SN95]. At the client layer, users submit their QoS requirements for the processing of their multimedia data. Various QoS parameters have been proposed to specify the requirements. For example, Little and Ghafoor [LG90a] have proposed several parameters to measure the QoS for multimedia data presentation. These QoS parameters are used to specify the requirements of multimedia data presentation. A more formal specification of QoS parameters are offered in [WS96], some of the important parameters are integrated into the presentation scheduling proposed below. In particular, we will consider the following QoS parameters:

- **Individual stream (intra-stream constraints)**:
  - *Maximum rate change*: denotes the range within which the rendition rate of stream s can vary. It is a fraction of the nominal rate for s.
  - *Maximum instantaneous drift*: denotes the maximum time difference between the presentation progress of the stream and the presentation progress when nominal presentation takes place at any given time.
- **Multiple streams (inter-stream constraint)**:
  - *Synchronization jitter*: denotes the maximum allowable drift between the constituent streams of the presentation. This drift is the difference between the presentation progress of the fastest and slowest streams.

The first two QoS parameters pertain to individual streams. The maximum instantaneous drift also implicitly provides the constraint on the overall maximum allowable drift for the presentation of the stream. The last one pertain to the entire presentation. Various system delays may result in the presentation of a stream to either slower or faster, which might cause jitter between the presentation of streams. We say that a presentation is *acceptable* if it is within the permissible QoS ranges. Several other QoS parameters discussed in [WS96] can also be incorporated into our framework.

Note that the scheduler at a client site need not consider the concurrent execution of multiple multimedia presentations. Rather, it must control the presentation of multiple media streams to a single user.

3.2 Synchronization Enforcement

We define a *synchronization point* to be a point held in common from all participating streams within a single multimedia presentation needing to be synchronized to enforce intra- and inter-stream constraints. We define the *granularity of synchronization* between a set of media streams to be the number of synchronization points that must be identified. Clearly, the finer the synchronization granularity, the more synchronization points will need to be identified.

We shall first discuss the sizes of media granules that is generally acceptable to be used for the purpose of enforcing intra-stream constraints. In this context, the actual presentation of a media stream is compared with its nominal defined presentation. Deviations between the two presentations must be within an acceptable range in order to guarantee satisfiable presentation of the stream.

When we display continuous media, there is ultimately a limit for the smallest part of the stream, known as stream granule that can be displayed atomically. This is determined by the hardware and also by the particular software implementation. The hardware limit for media granularity in videos using a frame buffer display is a pixel. Similarly the hardware limit for audio is one audio sample. However, while scheduling audio or video streams, it may be better to handle larger granules. The overheads of scheduling each pixel or each sample may be very high. In our implementation, we deal with schedules for each video frame, and so the granularity is one video frame. Similarly, one audio granule has samples which correspond to the presentation time of one frame. As shown below, these sizes of media granules of audio and video offer quite impressive performance.

We shall now discuss the effects of granularity of synchronization, when inter-stream constraints are enforced. Consider the synchronization of video and audio streams. Under normal circumstances, it is the video stream which lags behind the audio stream. This is because video data tends to be larger, and overhead for displaying video frames is more than that for audio. If we let the two streams be presented without enforcing synchronization, the jitter between the two streams will gradually build up. A small amount of jitter is easily perceived when dealing with audio and video streams, especially in applications where speech is involved. Lip synchronization requires fine grained synchronization between the streams. However, overheads in computation will be encountered at each synchronization point. Hence, we desire to keep in balance the synchronization requirements, and the restrictions imposed by synchronization overhead. A synchronization granule may contain one or more media granules.

The granularity of synchronization, in addition to playing a role in inter-stream synchronization, also affects the smoothness of the presentation. Let us consider the case when the number of synchronization points are few, and one stream is slower than the other. The longer we wait to check the synchronization status of the streams, larger would be the relative drift between them. This drift may eventually cause the faster stream to pause at the synchronization point. Hence, lower the number of synchronization points, larger the drift, and consequently, larger the pauses in the faster stream. The situation is especially bad if it is the audio stream that is faster (which is usually the case). At each synchronization point, the audio stream will have to wait. The human ear is usually very sensitive to the pauses in audio that occur at these points.

Our primary goal in designing the presentation scheduling algorithms is to generate presentations with as few hiccups as possible. Thus, users will always be presented smooth presentations. Continuous media streams are typically generated by sampling the required data at a high rate. Each sample is a representation of the stream at the given instant. Though the sample does not have any continuity on its own, when the different samples are played back at particular rate, they are perceived as a continuous stream. We observe that when samples are skipped but the rate at which samples are displayed is kept the same, it appears as though the presentation rate has increased. If a very large percentage of samples are skipped, the viewer starts to perceive each sample as a distinct one, and the stream does not appear to be continuous. This percentage depends on the kind of stream (audio or video), and the sampling rate of the original stream. Hence at relatively low drop rates, there is no difference between dropping of samples and actually increasing the presentation speed by reducing the time between consecutive samples. Thus, dropping 5% of the media samples will cause the stream to appear 5% faster. Similarly, the stream can be slowed down by decreasing the presentation rate, or by introducing pauses between media streams.

4 Intra-stream Scheduling

In this section, we will discuss the scheduling principles and algorithms for the presentations of individual continuous streams.

Let the nominal rendition rate for a stream s_i be denoted by N_{s_i}. It is expressed as the number of stream granules per unit time. For example, the nominal rendition rate for video could be 30 frames per second, assuming one frame to be a granule. To meet this scheduling requirement, the time period between consecutive frames must be 1/30 seconds. Intra-stream synchronization can thus be achieved by displaying the frames at this rate. However, it may not be possible to always achieve this rate, because of various factors which affect the load on the system. So we must specify a reasonable range for the display operations. Following the discussion in Section 3, we will integrate two intra-stream QoS parameters, maximum rate change c_{s_i} and maximum instantaneous drift d_{s_i}, into the presentation of each stream.

The maximum rate change c_{s_i} represents the range within which the rendition rate of the stream s can vary. It is a fraction of N_{s_i},

$$N_{s_i} * (1 - c_{s_i}) \leq R_{s_i}(t) \leq N_{s_i} * (1 + c_{s_i}), \tag{1}$$

where $R_{s_i}(t)$ is the rendition rate of s_i at a time instant t.

We assume that, when a stream granule is being displayed, the rendition rate $R_{s_i}(t)$ of s_i remains constant. We now define *progress* $p_{s_i}(t)$ of a stream s_i at time t. This is the presentation time of the granules displayed so far, if they had been presented at the nominal rate. In general, progress is measured in terms of the granules being displayed. Let the presentation of stream s_i is started at time t_0. Stream s_i is said to be progressing at the nominal rate if at time t, the granules being displayed is $N_{s_i} * (t - t_0)$. In our discussion, instead of specifying progress in terms of granules, we specify it in terms of time. Stream s_i is progressing at nominal rate if at time t, the progress measured in time is also $t - t_0$. This is given by dividing the currently displayed granules ($N_{s_i} * (t - t_0)$ for nominal rate) by N_{s_i}. We can express the progress in terms of time by dividing the granules being displayed by N_{s_i}. If the progress is less than (or greater than) $t - t_0$, then the presentation is below (or above) the nominal rate. The presentation is at nominal rate if progress is equal to $t - t_0$.

The nominal length of presentation time of each granule in s_i is $1/N_{s_i}$. If, at time t, the kth granule of s_i is being displayed and the display of this granule started at time $t - \triangle t$, we then have the progress of s_i calculated by

$$p_{s_i}(t) = (k-1)/N_{s_i} + \triangle t * R_{s_i}(t - \triangle t)/N_{s_i}.$$

In the above formula, $(k-1)/N_{s_i}$ calculates in time the progress for the display of the previous $k-1$ granules, and $\triangle t * R_{s_i}(t - \triangle t)/N_{s_i}$ estimates the progress between the end of $k-1$th granule at time $t - \triangle t$ and t. We make the assumption that once a granule display has started, any change in rate that is brought about by the synchronization algorithm will take effect only after the granule has completed. Theoretically, it may be possible to maintain exactly how far a granule has progressed, and this value can be substituted for $\triangle t * R_{s_i}(t - \triangle t)/N_{s_i}$, but there are practical limitations to this approach.

The maximum instantaneous drift d_{s_i} denotes the maximum time difference between the progress of the stream and the progress when nominal presentation takes place at any given time. If the stream started at time t_0, then under nominal presentation rate, the progress at time t is $t - t_0$. Thus, we should have

$$\mid p_{s_i}(t) - (t - t_0) \mid \leq d_{s_i}.$$

Intra-stream scheduling must be performed so that the constraints imposed by c_{s_i} and d_{s_i} are met. We enforce the intra-stream constraints by determining the progress of the stream at set intra-synchronization points, and deciding a value for $R_{s_i}(t)$ such that the constraints are met at the next point. The intra-synchronization points occur at regular intervals $\triangle t_s$. The rendition rate is set at a time t and remains the same until time $t + \triangle t_s$.

Depending on the particular implementation, the rendition rate may or may not take effect immediately. In certain cases we may be able to guarantee a time within which the new rate takes effect. Certain other implementations may not be able to change the rate until the current granule being displayed has finished. In our discussion, we assume that the rendition rate does not take effect while a stream granule is being displayed. However, the equations which we develop can easily be modified to accommodate the former case. Only the value of t_{rem} described below needs to change.

If we have an intra-synchronization point at time t and $R_{s_i}(t)$ is the rendition rate that has been calculated, then we can determine the progress expected at time $t + \triangle t_s$. If, at time t, the kth granule is being displayed and the display of this granule was started at time $t - \triangle t$, we must first calculate the time during which the current granule is being displayed at the old rendition rate,

$$t_{rem} = max(0, 1/R_{s_i}(t - \triangle t) - \triangle t).$$

It may so happen that with the rate of $R_{s_i}(t - \triangle t)$, the granule should have finished by the current time t. This would give a negative value for $1/R_{s_i}(t - \triangle t) - \triangle t$. In this case, t_{rem} would be 0. What this means is that we assume that by the time we assign a new rendition rate, the presentation of the current granule would be completed. This may not be a valid assumption. But, in this case, we must employ some kind of heuristic which determines how much longer the granule would continue, based on previous history of the presentation.

Once the current granule has been completed, the remaining time until the next intra-synchronization point can be utilized to display the stream at the new rate. If $t_{rem} < \triangle t_s$, we have

$$p_{s_i}(t + \triangle t_s) = k/N_{s_i} + (\triangle t_s - t_{rem}) * R_{s_i}(t)/N_{s_i}. \tag{2}$$

Otherwise,

$$p_{s_i}(t + \triangle t_s) = (\triangle t + \triangle t_s) * R_{s_i}(t - \triangle t)/N_{s_i} + (k - 1)/N_{s_i}. \tag{3}$$

We know the range within which $R_{s_i}(t)$ can vary from Equation (1). Dividing Equation (1) by N_{s_i} we get

$$1 - c_{s_i} \le R_{s_i}(t)/N_{s_i} \le 1 + c_{s_i}.$$

We can now obtain the range of $p_{s_i}(t + \triangle t_s)$ satisfying the c_{s_i} constraint. The required range is

$$[k/N_{s_i} + (\triangle t_s - t_{rem}) * (1 - c_{s_i}), k/N_{s_i} + (\triangle t_s - t_{rem}) * (1 + c_{s_i})].$$

This is for the case when $t_{rem} < \triangle t_s$. Otherwise, there is only one value for $p_{s_i}(t + \triangle t_s)$ as given by Equation (3). The width of the range is 0.

In addition to this, there is a second range for $p_{s_i}(t + \triangle t_s)$ which satisfies the d_{s_i}. The nominal progress at time $t + \triangle t_s$ is $t + \triangle t_s - t_0$, where t_0 is the start of the stream. This gives a range

$$[t + \triangle t_s - t_0 - d_{s_i}, t + \triangle t_s - t_0 + d_{s_i}].$$

Combining the above two ranges, we get the range for $p_{s_i}(t + \triangle t_s)$ to satisfy both the intra-stream constraints,

$$\begin{aligned}[MAX(k/N_{s_i} + (\triangle t_s - t_{rem}) * (1 - c_{s_i}), t + \triangle t_s - t_0 - d_{s_i}),\\ MIN(k/N_{s_i} + (\triangle t_s - t_{rem}) * (1 + c_{s_i}), t + \triangle t_s - t_0 + d_{s_i})].\end{aligned} \tag{4}$$

If the lower part of the range is higher than the higher part of the range, then by varying the rate, we cannot meet the intra-stream constraints at the next intra-synchronization point. If

$t_{rem} \geq \triangle t_s$, then there is only one possible value for progress. If this progress does not satisfy d_{s_i} constraint, we may have to abort the presentation.

We can have different policies for choosing what the progress should be at the next intra-synchronization point. One possible policy is to minimize the change in rate, that is, to choose the progress which would require least change from nominal rate. Another policy is to choose the progress which would result in least deviation from $t + \triangle t_s$. We below discuss how to choose a specific rate from the given range, if it exists.

4.1 Minimizing Change in Rate

Change in rate would be minimized if we choose a rate that is closest to the nominal rate. From Equation 2, we can calculate the progress when $R_{s_i}(t) = N_{s_i}$. Let this progress be $pn_{s_i}(t + \triangle t_s)$. Depending on whether this progress lies to the left, right or within the range defined in (4), $p_{s_i}(t + \triangle t_s)$ would be different. Let the progress corresponding to the left and right of the range defined in (4) be $pl_{s_i}(t + \triangle t_s)$ and $pr_{s_i}(t + \triangle t_s)$, respectively, we then have the following policy for deciding the progress at the next intra-synchronization point,

$$p_{s_i}(t + \triangle t_s) = \begin{cases} pl_{s_i}(t + \triangle t_s) & \text{if } pn_{s_i}(t + \triangle t_s) < pl_{s_i}(t + \triangle t_s) \\ pr_{s_i}(t + \triangle t_s) & \text{if } pn_{s_i}(t + \triangle t_s) > pr_{s_i}(t + \triangle t_s) \\ pn_{s_i}(t + \triangle t_s) & \text{Otherwise} \end{cases} \qquad (5)$$

Substituting for this value of progress in Equation 2, we can determine the required rate. Again if $t_{rem} \geq \triangle t_s$, then there is only one possible value for progress which must also satisfy the d_{s_i} constraint.

4.2 Minimizing Drift

As in the previous case, drift can be minimized at $t + \triangle t_s$ if we choose a rate that gives progress closest to this value, as follows:

$$p_{s_i}(t + \triangle t_s) = \begin{cases} pl_{s_i}(t + \triangle t_s) & \text{if } t + \triangle t_s < pl_{s_i}(t + \triangle t_s) \\ pr_{s_i}(t + \triangle t_s) & \text{if } t + \triangle t_s > pr_{s_i}(t + \triangle t_s) \\ pn_{s_i}(t + \triangle t_s) & \text{Otherwise} \end{cases} \qquad (6)$$

Substituting for this value of progress in Equation 2, we can determine the required rate.

Note that the actual progress that is made may be different from the calculated value. This may depend on the current load on the machine, network loads and a host of other factors. Again, we would have to rely on some history based heuristic to see what progress may be made. The range given above serves as a reasonable estimate of the progress. In a practical implementation, we would have to check the actual progress at each intra-synchronization point. If the intra-stream synchronization requirements are not being met at sufficiently large number of intra-synchronization points, the presentation will be aborted.

5 Inter-Stream Synchronization

In the previous section, we have discussed scheduling requirements for a single stream. However, when multiple streams are involved, it is not sufficient to only meet user specifications for the display of the individual streams. All the streams must be synchronized. In this section, we will discuss the scheduling principles and algorithms for the synchronized presentation of multiple continuous streams.

We assume that the user can specify an inter-stream jitter j in order to enforce synchronization requirements. Suppose we have n streams, then at each synchronization point, in addition to determining $R_{s_i}(t)$ such that the intra-stream constraints of each stream are met, the inter-stream constraint imposed by j must be met.

Following the previous section, we can calculate $p_{s_i}(t + \triangle t_s)$ for each stream s_i at the synchronization point of time t, depending on the rendition rate we have determined. We can formally specify the inter-stream synchronization constraint in terms of the progress of the streams. Without loss of generality, let inter-synchronization points occur at regular intervals $\triangle t_s$. We first define $p_{s_{max}}(t + \triangle t_s)$ and $p_{s_{min}}(t + \triangle t_s)$, as follows:

$$p_{s_{max}}(t + \triangle t_s) = MAX(p_{s_i}(t + \triangle t_s)), i = 1 \cdot \cdot n$$
$$p_{s_{min}}(t + \triangle t_s) = MIN(p_{s_i}(t + \triangle t_s)), i = 1 \cdot \cdot n$$

The inter-stream synchronization requirements can be met if for all synchronization points at $t + \triangle t_s$,

$$p_{s_{max}}(t + \triangle t_s) - p_{s_{min}}(t + \triangle t_s) \leq j,$$

where t denotes the time at which the current synchronization point begins.

We present two algorithms which calculate rendition rates for the streams at a synchronization point occurring at time t.

5.1 Minimizing Jitter

The first algorithm determines $R_{s_i}(t)$ for all the streams $i = 1, ..., n$ (within constraints of c_{s_i} and d_{s_i}) so that jitter is minimized at the next synchronization point. In other words, the algorithm must minimize

$$p_{s_{max}}(t + \triangle t_s) - p_{s_{min}}(t + \triangle t_s).$$

The above value must be less than or equal to j for the inter-stream constraint to be met.

The second algorithm attempts to meet the user requirements while at the same time trying to maintain the streams 'close' to their nominal presentation rate. Any change from the nominal presentation rate results in degradation of the stream. For a given value of j, the algorithm determines rendition rates R_{s_i} for each stream s_i with nominal presentation rate N_{s_i} such that the sum of the individual stream rate changes

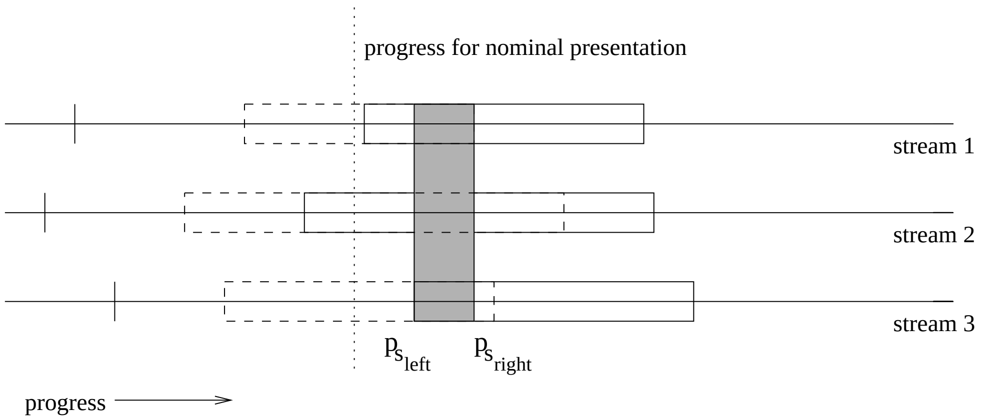


Figure 2: A presentation of streams.

$$\sum_{i=1}^{n} \frac{\mid R_{s_i}(t) - N_{s_i} \mid}{N_{s_i}}$$

is minimized. By minimizing this sum, the change of rendition rate in all the streams combined is minimized. We call this sum presentation rate change.

It is easier to explain the working of the algorithms by means of a figure. Figure 2 depicts three streams which have progressed differently at a synchronization point. The solid box depicts the progress stream s_i can make by limiting rendition rate according to c_{s_i}. The dashed box depicts the progress using rendition rates to satisfy d_{s_i}. The intersection of all these boxes is shown in dark. The range of progress that falls within the intersection is the range lying between and including the highest progress occurring among the start of the boxes (denoted $p_{s_{left}}(t + \triangle t_s)$ in the figure) and the minimum progress occurring among the end of the boxes (denoted $p_{s_{right}}(t + \triangle t_s)$ in the figure). No intersection is possible if the former value of progresses is greater than the later value. Using the definitions for $pl_{s_i}(t + \triangle t_s)$ and $pr_{s_i}(t + \triangle t_s)$ from the intra-stream scheduling section, we have,

$$p_{s_{left}}(t + \triangle t_s) = MAX(pl_{s_i}(t + \triangle t_s)), i = 1 \cdot \cdot n$$
$$p_{s_{right}}(t + \triangle t_s) = MIN(pr_{s_i}(t + \triangle t_s)), i = 1 \cdot \cdot n$$

We obtain the intersection to be $[p_{s_{left}}, p_{s_{right}}]$.

For any point within this intersection, the intra-stream constraints for all three streams will be met. If there is no intersection, it is still possible to schedule the streams, but by introducing some jitter. We handle both cases separately, as follows:

- Jitter $j = 0$, that is
$$p_{s_{max}}(t + \triangle t_s) - p_{s_{min}}(t + \triangle t_s) = 0,$$
which means that $p_{s_1}(t + \triangle t_s) = p_{s_2}(t + \triangle t_s) = \cdot\cdot = p_{s_n}(t + \triangle t_s)$. Any progress value falling within the intersection can be chosen to make jitter to be 0,
$$\forall i = 1 \cdot \cdot n, p_{s_{left}}(t + \triangle t_s) \le p_{s_i}(t + \triangle t_s) \le p_{s_{right}}(t + \triangle t_s).$$

The corresponding rate can be calculated as described in the previous section and can be assigned to each stream. However, we can do better than this. We can determine a progress value for all the streams such that the presentation rate change is minimized.

First, we determine $p_{s_i}(t)$ for all s_i when $R_{s_i}(t) = N_{s_i}$. For each stream this value of progress may fall to the left of the intersection box, within the intersection box, or to the right of the intersection box. For each stream, the progress values within the intersection box range correspond to some rendition rate.

Second, we determine the progress versus rate change function for each individual stream. Following (2), we define the rate change function for s_i:

$$f_{s_i}(p_{s_i}(t + \triangle t_s)) = \left| \frac{p_{s_i}(t + \triangle t_s) - k/N_{s_i}}{\triangle t_s - t_{rem}} - 1 \right| . \quad (7)$$

$f_{s_i}(p_{s_i}(t + \triangle t_s))$ represents various values of $\frac{|R_{s_i}(t) - N_{s_i}|}{N_{s_i}}$ for given progress $p_{s_i}(t + \triangle t_s)$. We plot the values of $\frac{|R_{s_i}(t) - N_{s_i}|}{N_{s_i}}$ within the intersection range. If the value of progress for $R_{s_i}(t) = N_{s_i}$ falls to the right (left) of the intersection box, then the values of $\frac{|R_{s_i}(t) - N_{s_i}|}{N_{s_i}}$ is decreasing (increasing) as progress increases. On the other hand, if the value of progress for $R_{s_i}(t) = N_{s_i}$ falls within the intersection box, the rate change function defined in (7) decreases to zero upto this point and then increases. The rate change function is a linear function, since $R_{s_i}(t)/N_{s_i}$ varies linearly with progress, and $\frac{R_{s_i}(t) - N_{s_i}}{N_{s_i}}$ varies linearly with $R_{s_i}(t)/N_{s_i}$. We can easily calculate the slope of this function from Equation 2. The slope of each function is $1/(\triangle t_s - t_{rem})$ when the function is increasing, and is $-1/(\triangle t_s - t_{rem})$ when the function is decreasing. The slopes may be different for each of the streams.

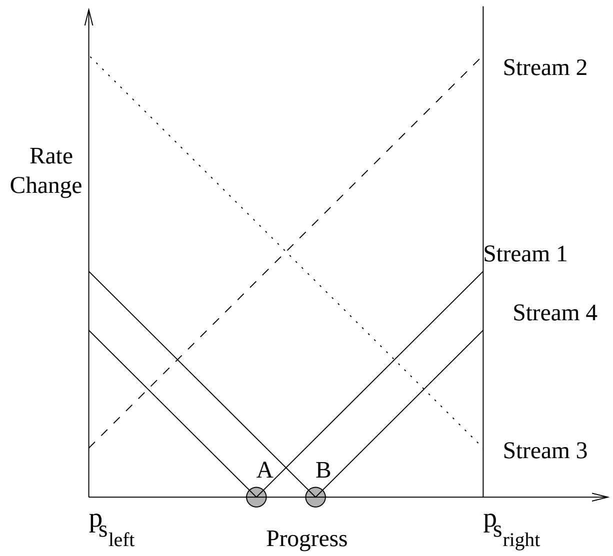


Figure 3: Progress versus Rate Change

A graph of a typical situation is shown in Figure 3. We can calculate presentation rate change rate at each point. The sum of the slopes of all four functions indicates whether the presentation rate change is increasing or decreasing. If the sum of the slopes is negative, the presentation rate change decreases upto the point where the sum of the slopes become greater than or equal to zero. This can only happen at either A or B, where the slopes of the constituent functions change from negative to positive. Either that, or the slopes remain negative, till the end of the intersection box.

If the sum of the slopes is positive or zero right from the start, then it will always remain positive, because there is never a change of slope from positive to negative or zero for any of the constituent functions. Hence the minimum presentation rate change occurs at the beginning of the intersection box.

We can conclude that the presentation rate change is minimized either at the start or end of the intersection box, or at the point where the slope of a constituent function changes. So we need to calculate the presentation rate change for only these points. The maximum number of points for which we would have to calculate this value is $n+2$ (where n is the number of streams), that is when the progress corresponding to nominal rendition rate for all streams fall within the intersection box. Hence we can determine the appropriate $p_{s_i}(t)$ in linear time.

- Jitter $j \neq 0$ When no intersection occurs, then the jitter cannot be zero. This happens when $p_{s_{left}} \geq p_{s_{right}}$. The presentation can only proceed if we allow a jitter of $p_{s_{left}} - p_{s_{right}}$. If this value is greater than j, then we must abort the presentation. Otherwise, the range of allowable values is

$$\forall i = 1 \cdot \cdot n, p_{s_{right}}(t + \triangle t_s) \leq p_{s_i}(t + \triangle t_s) \leq p_{s_{left}}(t + \triangle t_s).$$

  For each stream s_i, we choose $R_{s_i}(t + \triangle t_s)$ which results in $p_{s_i}(t + \triangle t_s)$ according to the following criteria:

  - If the progress achieved by nominal rate is less than the progress range, then the minimum rate change occurs for

$$p_{s_i}(t + \triangle t_s) = p_{s_{right}}(t + \triangle t_s)$$

  - If the progress achieved by nominal rate is greater than the progress range, then the minimum rate change occurs for

$$p_{s_i}(t + \triangle t_s) = p_{s_{left}}(t + \triangle t_s)$$

  - If the progress achieved by nominal rate lies between the progress range, then set the rendition rate equal to the nominal rate.

5.2 Minimizing Change in Rate

This algorithm allows the maximum allowable jitter j between the stream, and tries to minimize the presentation rate change. This algorithm is an extension of the previous algorithm.

Comparing with the previous section, the calculation for the intersection is modified. At time t, we first calculate the allowable range $[pl_{s_i}(t+\triangle t_s), pr_{s_i}(t+\triangle t_s)]$ of progress $p_{s_i}(t+\triangle t_s)$ for s_i to meet the intra-stream constraints. This range follows the definition of (4). We then extend this range on either side by j, that is, $[pl_{s_i}(t+\triangle t_s)-j, pr_{s_i}(t+\triangle t_s)+j]$. The extended range offers an allowable progress range for any other stream s_j such that if $p_{s_j}(t+\triangle t_s) \in [pl_{s_i}(t+\triangle t_s - j, pr_{s_i}(t+\triangle t_s)+j]$, there exists $p_{s_i}(t+\triangle t_s) \in [pl_{s_i}(t+\triangle t_s), pr_{s_i}(t+\triangle t_s)]$ satisfying

$$p_{s_i}(t+\triangle t_s) - p_{s_j}(t+\triangle t_s) \leq j.$$

We now take the intersection $[p_{s_{left}}(t+\triangle t_s), p_{s_{right}}(t+\triangle t_s)]$ of the extended regions for all the streams combined, as shown by the dark box in Figure 4.

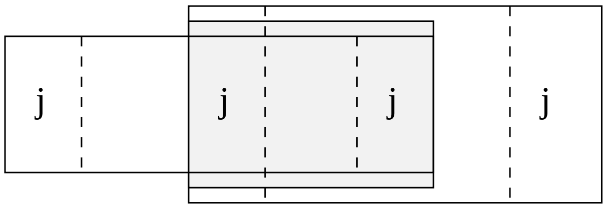


Figure 4: Intersection of stream ranges extended by j

From the previous section, we have seen that when a maximum jitter of j is allowed, $p_{s_{left}}(t+\triangle t_s) - p_{s_{right}}(t+\triangle t_s) \leq j$. When we extend the ranges to either side by j, $p_{s_{left}}(t+\triangle t_s)$ decreases by j and $p_{s_{right}(t+\triangle t_s)}$ increases by j. Hence the intersection range $[p_{s_{left}}(t+\triangle t_s), p_{s_{right}}(t+\triangle t_s)]$ will have width of at least j. If the range is less than j, then no schedule is possible such that the inter-stream synchronization constraint can be met.

To determine the rendition rates such that presentation rate change is minimized, we must scan the intersection range using a window of width j. Within this window, there is some value for progress for which the rate change for each stream is minimum. We call this rate change corresponding to this value of progress minimum rate change. The progress corresponding to minimum rate change is very simple to obtain. If the progress for nominal rate falls to the right (left) of the window, then minimum rate change occurs for progress at the right (left) edge of the window. If this progress falls within the window then the minimum rate change within the window is zero. In the previous section, we have plotted the rendition rate change of the individual streams as a function of progress. We can obtain a similar plot using the window. The left edge of the window moves from $p_{s_{left}}(t+\triangle t_s)$ upto $p_{s_{right}}(t+\triangle t_s)-j$, and we plot the minimum rate change versus progress of the left edge of the window.

We have plotted two graphs for a typical scenario, as shown in Figure 5. The first one shows the rate changes versus progress as in the previous section. We see that for stream 3, there is no valid rate change at $p_{s_{left}}(t+\triangle t_s)$. Valid rate changes only appear after a gap no larger than j. This is because, we have extended the valid ranges for each stream by j on either side. The same effect occurs on the right edge for stream 2. We can also observe from Figure 4 that since the right and left edges of the intersection are formed by the right and left edges of a range of at least

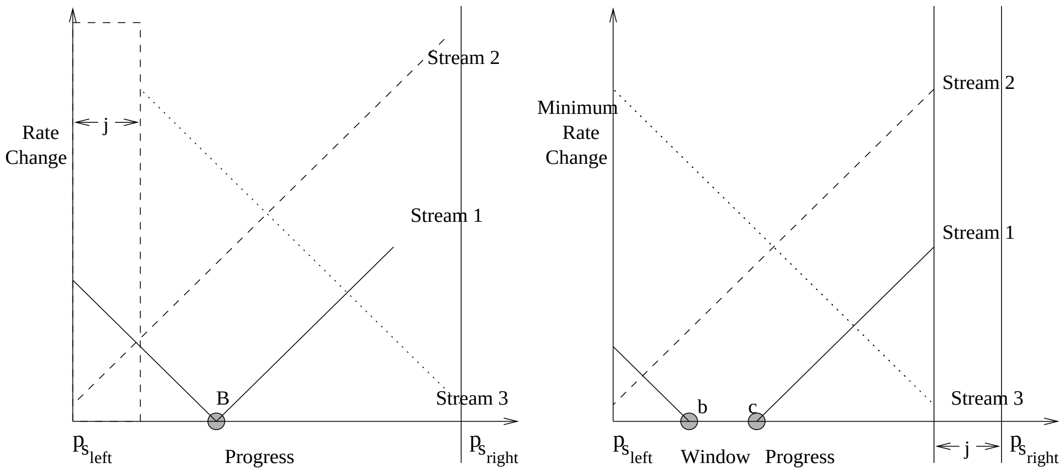


Figure 5: Window Progress versus Minimum Rate Change

one stream, at least one stream will have a gap of j from $p_{s_{left}}$ and $p_{s_{right}}$. Since the maximum gap within which there is no valid progress rate for a stream is j, we are guaranteed that within a window of j, some progress valid progress value for the stream exists.

The second graph shows minimum rate change in the window of width j versus progress of the left edge As we can see, the slope of the minimum rate change functions changes when the left edge of the window falls at points b and c corresponding to progress $B - j$ and B.

As in the previous section, we now only need to check the presentation rate change at the points where the slope changes, and at $p_{s_{left}}(t + \triangle t_s)$ and $p_{s_{right}}(t + \triangle t_s) - j$.

There is a change in slope for streams whose progress corresponding to nominal rate occurs within the intersection windows. This progress is B for stream 1. The slope changes from negative to zero when the right edge of the window reaches B. This means that the left edge is at $B - j$. The slope changes from zero to positive when the left edge of the window reaches B. So zero slope continues only for width j. If $B - j$ lies to the left $p_{s_{left}}(t + \triangle t_s)$, we need not check the point.

If the progress corresponding to nominal rate change for all n streams falls within the intersection, at most $2n + 2$ points need to be checked.

6 Algorithm Performance

6.1 Running Time

We have seen that for both algorithms we need to consider only order of n points where n is the number of streams. Each of these points correspond to some value of progress. We also observe that the slopes of the functions never decrease at any of the points. It only increases. The sum of the slopes at a point indicates whether the presentation rate change is decreasing or increasing at that point.

In order to find the point of minimum presentation rate change, we first find the sum of the slopes at $p_{s_{left}}$. If the sum is positive, then the minimum progress rate change occurs at $p_{s_{left}}$. Otherwise, we sort the points to be checked according to increasing progress. At each point we determine the new slope of the function for which that point was taken. We add the difference from previous slope to the current total slope. As soon as the total slope becomes positive, we can stop at that point of progress (since we know that slope of any function does not decrease). The minimum presentation rate change occurs here.

Since the sorting takes $O(n \log n)$ time, and the addition at each of the n points take constant time, the algorithm runs in $O(n \log n)$ time. The algorithms can be used to synchronize a large number of streams.

6.2 Test Results

The algorithms presented above address different applications. In cases where close synchronization between the streams is of greater importance than the quality of the streams themselves, we would try to minimize jitter. This would be the case for a lecture presentation where lip synchronization must be maintained. In other presentations, such as a narrative of a particular scene, close synchronization is not a priority. Here we can minimize the rate changes of the streams.

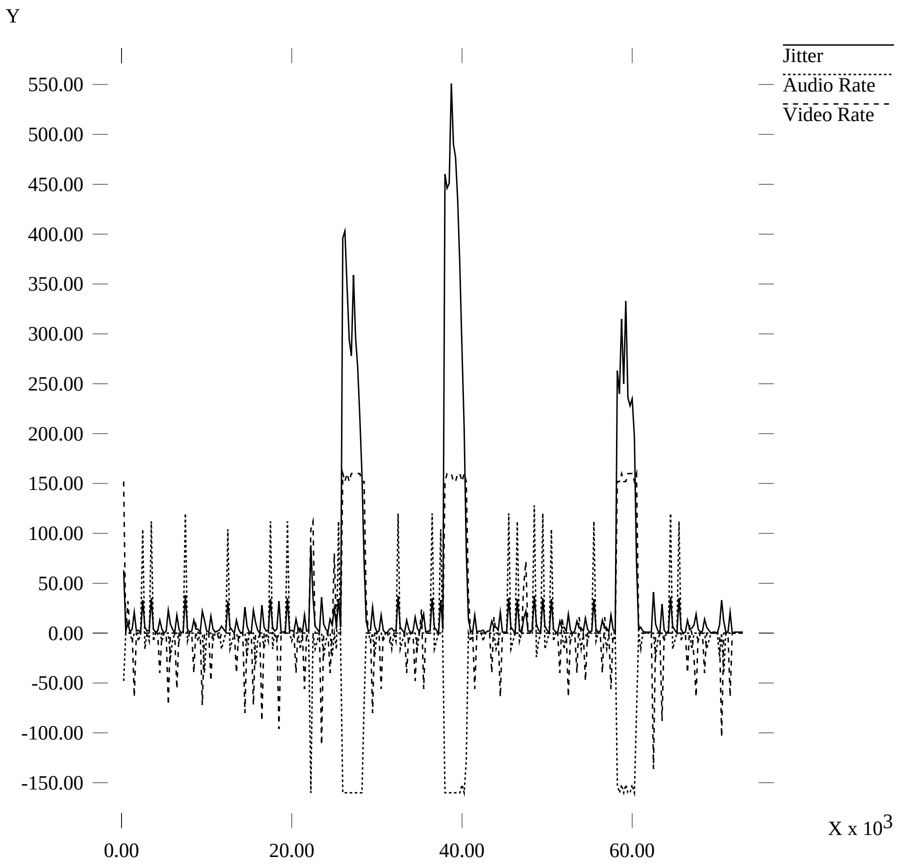


Figure 6: Jitter in a loaded system

The performance is exceptionally good even in conditions of random delays in the system. We present some results for the first algorithm. All the tests have been carried out on a Sun SPARCstation 4 running Solaris. We have plotted jitter between the streams versus time. The synchronization points are set at 250 ms apart. We see that the jitter between the streams is minimal. We have also superimposed the rates of the audio and video streams. They vary between $\pm 20\%$. We heavily load the CPU at three different points. Figure 6 illustrates the situation. The jitter increases momentarily, but the streams quickly get synchronized again. We notice that it is the video stream that is slowing down under load, since the video decoding process requires large CPU time. The algorithm slows down the audio stream and speeds up the video stream to make the jitter zero. By doing so the streams get synchronized in a smooth fashion. The user will not notice any hiccups within the streams, just a momentary loss in synchronization.

Since the algorithm tries to minimize jitter, and then minimize presentation rate change, we see that high synchronization is maintained with low rate change. The streams are better synchronized when we reduce the synchronization granularity, that is when we reduce $\triangle t_s$.

The key to the efficiency of the algorithm is the specification of scheduling information using rates. We need not specify the schedules for each granule of the stream individually. The algorithm does not make any assumptions as to how the rates of the individual streams are actually varied.

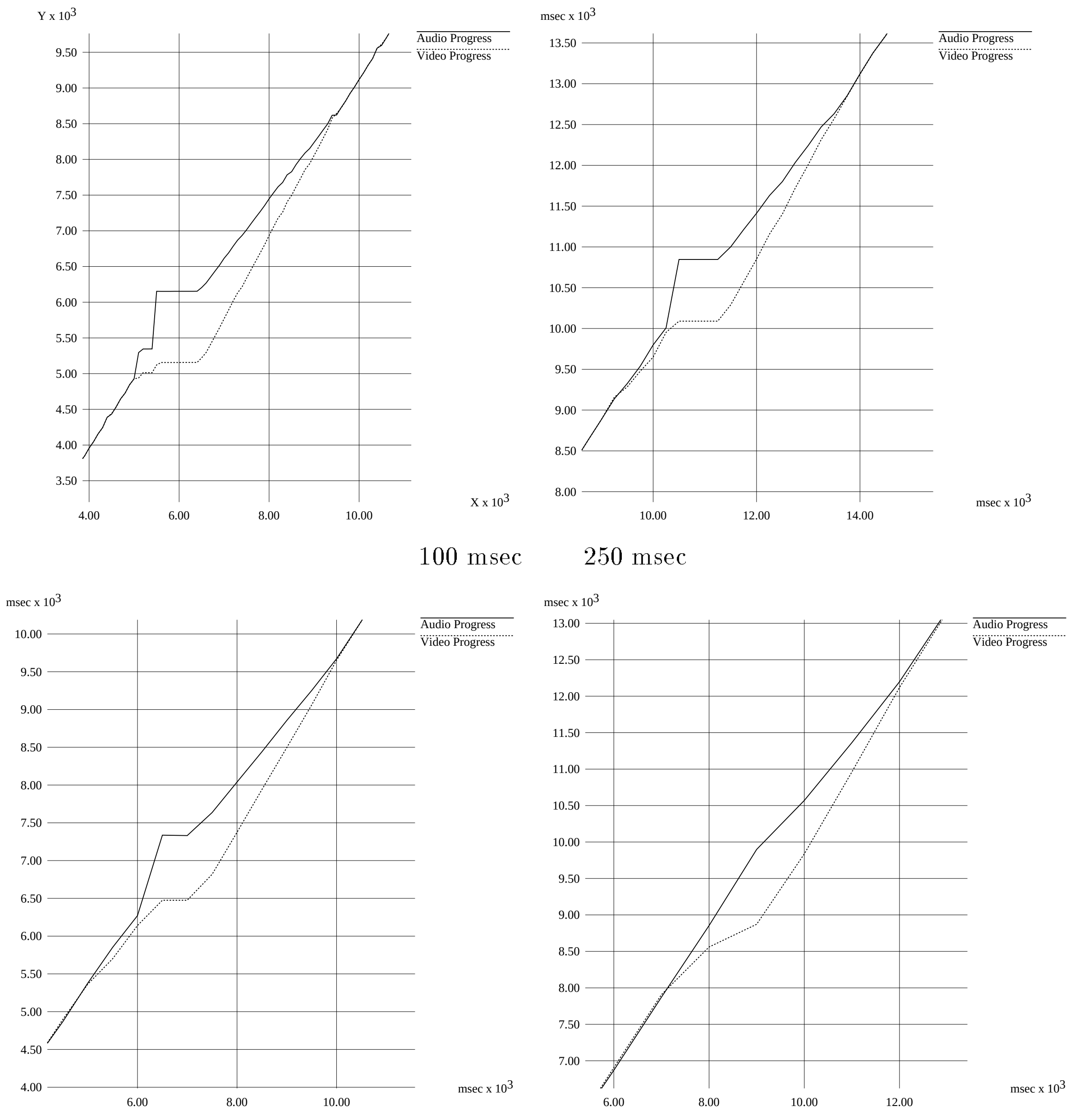
There is a trade-off between better synchronization achieved by increasing the number of synchronization points and overheads caused by these synchronization points. Larger number of synchronization points may lead to loading of the system resources and subsequent degradation in performance. We have observed that having synchronization points every 250 ms gives good performance. The skews (average jitter) between the streams on a lightly loaded system vary between 3ms to 40ms as the synchronization points are changed from every 75ms to 2 secs.

When we have larger number of synchronization points, the recovery time for the system is reduced in the presence of loads. We present the progress of the streams versus time in ms under the same load, using different time between synchronization points. We see that for 100 and 250 ms between synchronization points, recovery is within 4 seconds, for 500 it is around 5 secs and for 1000 around 6 seconds. Figure 7 illustrates these cases.

There is a limit on how small we can make $\triangle t_s$. If the processing time at each synchronization point consumes a substantial portion of $\triangle t_s$ then the prediction that we make will not be accurate.

6.3 Algorithm Enhancements

- Currently, the new rate that is being calculated does not take into account the previous history of the presentation. This does not pose much of a problem when there are only transient changes in the load. However, when there are constant loads on the system, or the system has more resources than expected, we may get consistently different rates than what has been specified. We can take this into account by assigning weights to the comparative rates achieved at the previous synchronization points and obtaining and taking a weighted average of the sum. Using this, we would be able to make more accurate predictions.



100 msec 250 msec

500 msec 1000 msec

Figure 7: Recovery time for synchronization points 100, 250, 500 1000 msec apart

- New QoS parameters such as average drift and average rate change can easily be incorporated. It may be the case that for a certain average value of rate change, the user may be able to view the presentation satisfactorily. However, he or she may be able to tolerate higher rate changes over short periods of time. In such cases it may be overly conservative to just have a value for c_{s_i}. By specifying an average value over a set number of synchronization points, and a maximum rate change, we would have greater flexibility in scheduling the streams. At each synchronization point, we could use larger values of rate change, as long as the average over the last synchronization points can be tolerated. We can similarly handle average drifts.

- It is debatable whether minimizing the sum of the rate changes is the best method of ensuring 'closeness' to the nominal presentation rate. Such a policy may result in several streams with a rate change of zero or close to zero, and a few streams with large rate changes. Such a situation could be avoided if we try to minimize the sum of the squares of the rate changes. Many other functions could be studied. In this case, we would have a similar analysis as for linear equations. We would have to calculate the minima of the function we are trying to minimize, within the intersection range.

7 Conclusions

In this paper, we have developed a framework for supporting quality-based multimedia presentations at client sites in a distributed multimedia system. Upon the assumption that the network transmission and the server provide sufficient support for delivering media objects, important issues regarding the enforcement of smooth presentations of multimedia streams have been addressed at client sites. We have developed several presentation scheduling algorithms that are adaptable to various QoS parameters. In case of delay, the recovery of the presentation from its insynchrony is performed by gradually catching up or slowing down rather than using skipping/pausing in an abrupt fashion. Experimental analysis conducted has demonstrated that our algorithms can avoid hiccups in presentations. These algorithms are important to maintain lip synchronization that must be maintained in many applications including education and training.

As mentioned in Section 6.3, there are still remaining issues to be addressed on this topic. Aspects on enhancing and extending the proposed algorithms have been discussed in the same section.

References

[AFKN95] R.T. Aptekar, J.A. Fisher, V. Kisimov, and H. Nieshlos. Video Acceptability and Frame Rate. *IEEE Multimedia*, 3(3):32–40, 1995.

[AH91] David P. Anderson and George Homsy. A Communication Media I/O Server and Its Synchronization Mechanisms. *IEEE Computer*, 24(10):51–57, 1991.

[All83] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of ACM*, 26(11), 1983.

[Buf94] John F. Koegel Buford. *Multimedia Systems*. Addison-Wesley Publishing Company, 1994.

[CAF+91] S. Christodoulakis, N. Ailamaki, M. Fragonikolakis, Y. Kapetanakis, and L. Koveo. An Object Oriented Architecture for Multimedia Information Systems. *IEEE Data Engineering*, 14(3), September 1991.

[CGS95] S. Chaudhuri, S. Ghandeharizadeh, and C. Shahabi. Avoiding Retrival Contention for Composite Multimedia Objects. In *Proceedings of the 21th VLDB conference*, Zurich, Swizerland, 1995.

[COC94] J. Courtiat, R. Oliverira, and L. Carmo. Toward a New Multimedia Synchronization Mechanism and its formal Specification. In *Proceedings of ACM Multimedia 94*, pages 133–140, San Francisco, California, October 1994.

[DC91] Y. Deng and S.K. Chang. A framework for the Modeling and Prototyping of Distributed Information Systems. *International Journal of Software Engineering and Knowledge Engineering*, 1(3):203–226, 1991.

[EM94] Martha Lucia Escobar-Molano. Management of Resources to Support Continuous Display of Structured Video Objects, 1994. (Technical Report, Department of Computer Science, University of Southern California).

[Fox95a] Edward Fox. Educational Digital Libraries, 1995. Colloquium Talk, SUNY at Buffalo.

[Fox95b] Edward Fox. Panel: Curriculum, Education and Training about Multimedia. In *Proceedings of ACM Multimedia 95*, pages 141–142, San Francisco, California, November 1995.

[Fur96a] Borko Furht. *Multimedia Systems and Techniques*. Boston: Kluwer Academic, 1996.

[Fur96b] Borko Furht. *Multimedia Tools and Applications*. Boston: Kluwer Academic, 1996.

[GBT94] Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. Data Modeling of Time-Based Media. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 91–102, Minneapolis, Minnesota, May 1994.

[Gha95] Arif Ghafoor. Special Issue on Multimedia Database Systems. *Multimedia Systems*, 3(5/6), 1995.

[GR93] Shahram Ghandeharizadeh and Luis Ramos. Continuous Retrievals of Multimedia Data Using Parallelism. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):658–669, 1993.

[GVKR95] D. J. Gemmell, H. M. Vin, D. D. Kandlur, and P. V. Rangan. Multimedia Storage Servers: A Tutorial and Survey. *Computer*, 28(5):40–51, May 1995.

[GZ96a] Sreenivas Gollapudi and Aidong Zhang. Buffer Management in Multimedia Database Systems. In *the Third IEEE International Conference on Multimedia Computing and Systems (ICMCS'96)*, Hiroshima, Japan, June 1996. A full version is under revision for Multimedia Systems.

[GZ96b] Sreenivas Gollapudi and Aidong Zhang. NetMedia: A Client-Server Distributed Multimedia Database Environment. In *Proceedings of the 1996 International Workshop on Multi-media Database Management Systems*, Blue Mountain Lake, New York, August 1996.

[LG90a] Thomas D.C. Little and Arif Ghafoor. Network Considerations for Distributed Multimedia Object Composition and Communication. *IEEE Network Magazine*, pages 32–49, 1990.

[LG90b] Thomas D.C. Little and Arif Ghafoor. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Arears in Communications*, 8(3):413–427, April 1990.

[Mas91] Yoshifumi Masunaga. Design Issues of OMEGA: An Object-Oriented Multimedia Database Management System. *Journal of Information Processing*, 14(1):60–74, 1991.

[NS95] Klara Nahrstedt and Ralf Steinmetz. Resource Management in Networked Multimedia Systems. *IEEE Computer*, 28(5):52–64, May 1995.

[NWS96] In *Proceedings of the 1996 International Workshop on Multi-media Database Management Systems*. Blue Mountain Lake, New York, August 1996.

[RRK93] P. Venkat Rangan, Srinivas Ramanathan, and Thomas Kaeppner. Performance of Inter-Media Synchronization in Distributed and Heterogeneous Multimedia Systems. *Computer Networks and ISDN Systems*, 1993.

[RV93] P. Venkat Rangan and Harrick M. Vin. Efficient Storage Techniques for Digital Continuous Multimedia. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):564–573, 1993.

[SF89] P.D. Stotts and R. Furuta. Petri-net-based Hypertext. *ACM Transaction on Office Automation Systems*, 7(1), 1989.

[SN95] Ralf Steinmetz and Klara Nahrstedt. *Multimedia: Computing, Communications and Applications*. Prentice Hall, 1995.

[Ste90] Ralf Steinmetz. Synchronization Properties in Multimedia Systems. *IEEE Journal on Selected Arears in Communications*, 8(3):401–412, April 1990.

[SUB96] In *2nd International Workshop on Multimedia Information Systems*. West Point, New York, September 1996.

[SWM95] R. Staehli, J. Walpole, and D. Maier. A Quality-of-service Specification for Multimedia Presentations. *Multimedia Systems*, 3(5/6):251–263, 1995.

[TK95a] Heiko Thimm and Wolfgang Klas. Playout Management – An Integrated Service of a Multimedia Database Management System, 1995. (Technical Report, GMD-IPSI).

[TK95b] Heiko Thimm and Wolfgang Klas. Reactive Playout Management: Adapting Multimedia Presentations to Contradictory Constraints, 1995. (Technical Report, GMD-IPSI).

[Tom89] F.W. Tompa. A Data Model for Flexible Hypertext Database Systems. *ACM Transaction on Information Systems*, 7(1), 1989.

[VKvBG95] A. Vogel, B. Kerherve, G. von Bochmann, and Gecsei. Distributed Multimedia and QoS: A Survey. *IEEE Multimedia*, 3(2):10–19, 1995.

[WS96] Duminda Wijesekera and Jaideep Srivastava. Quality of Server (QoS) Metrics for Continuous Media. *The International Journal on Multimedia Tools and Applications*, pages 127–166, September 1996.

[ZF93] T.F. Znati and B. Field. A Network Level Channel Abstraction for Multimedia Communication in Real-time Networks. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):590–599, 1993.

[ZG96] Aidong Zhang and Sreenivas Gollapudi. QoS Management in Educational Digital Library Environments. *The International Journal on Multimedia Tools and Applications*, 1996. (to appear, also a technical report (CS-TR-95-53) in Computer Science, SUNY at Buffalo).