Buffer Management in Multimedia Database Systems

Sreenivas Gollapudi and Aidong Zhang
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260
{golla-s, azhang}@cs.buffalo.edu

Abstract

The delivery of continuous and synchronous multimedia data from a database server to multiple destinations over a network presents new challenges in the area of buffer management. Many factors that were not considered in conventional buffer management must be examined. In this paper, we investigate the principles of buffer management for multimedia data presentations in object-oriented database environments. The primary goal is to minimize the response time of multimedia presentations while ensuring that all continuity and synchronization requirements are satisfied. Minimum buffering requirements to guarantee the continuity and synchrony of the presentation of multimedia data are first proposed. A prefetching technique which satisfies these requirements is then developed. These principles and technique provide users with the full range of information required to develop a database environment for multimedia presentations.

1 Introduction

Advances in multimedia computing technologies have offered new opportunities to store, manage, and present multimedia data in database systems. It has already become evident that object-oriented data representation and management provides a promising approach to the manipulation of multimedia data. To support its flexible retrieval, continuous media data must first be parsed into representable segments, which we term *media objects*. In order to re-present the original data stream to users, synchronization constraints among media objects must be specified and maintained. Such synchronization is usually termed *intra-stream synchronization*. An additional complication arises with the timing relationships that may exist among the different types of media data streams. For example, many multimedia applications, such as recording and playback of motion video and audio, slide presentations, and video conferencing, require the synchronized display of multiple media data streams. However, such media data streams may not be merged prior to storage in a database, and such a merger will vastly compound the difficulties of retrieving component media. Thus, the synchronization of multiple media data streams, termed *inter-stream synchronization*, becomes an essential prerequisite to any successful multimedia database application [LG90b, LG90a].

Buffer management within the multimedia database system is essential to ensure the maintenance of the intra- and inter-stream synchronization requirements of multimedia data presentations.

To facilitate a hiccup-free presentation, we must ensure that an object is present in memory before it is displayed. If the loading rate of a media stream from disk to memory is less than the delivery rate of the media stream, preloading of the stream prior to delivery would be necessary to ensure continuous presentation. If multiple media streams are synchronously presented, the buffer space must be sufficient to simultaneously hold all these streams. Furthermore, an appropriate allocation and replacement strategy must be provided to anticipate the demands of delays and user interactions. Such a strategy must minimize the response time of multimedia presentations while guaranteeing that all continuity and synchronization requirements are satisfied. Thus, the central issues in this context involve the provision of efficient preloading and replacement mechanisms for caching media objects. These mechanisms are intended to ensure that the defined continuity and synchrony of media streams will be preserved to the greatest extent possible, even if there are user interactions and loading delays.

Research involving buffer management in multimedia database systems is still in its infancy [TK95]. Moser et al [MKK95] have proposed a buffer strategy termed "least/most relevant for presentation." This buffer strategy investigates the effects of such user interactions as "rewind" and "fast forward" on buffer design. A mechanism is proposed which reduces the ensuing delay after user interactions. Chaudhuri et al [CGS95] have investigated the problem of continuously displaying composite objects that are dynamically specified at the server level. Techniques based on simple sliding and buffered sliding are proposed which support continuous display by partial prefetching of overlapping media objects. Such an approach is preferable to the naive strategy of prefetching the entirety of overlapped media objects. Ng and Yang [NY94] have attempted to maximize buffer and disk utilization within the constraints of a fixed amount of buffer space and disk bandwidth, both of which are pre-determined in the design phase. Prefetching strategies that are applicable to database systems are proposed which maximize the throughput of the system.

In this paper, we systematically study the principles underlying buffer management for multimedia data presentation in object-oriented database environments. Media streams are stored as sets of media objects coupled with specified synchronization requirements. Complementary to the work proposed by Chaudhuri et al [CGS95], we investigate the minimum buffering requirements that are necessary to guarantee the continuity and synchrony of the presentation of multimedia data. This is accomplished by first determining the minimum consuming times of media objects following their loading times, while guaranteeing the hiccup-free presentation of the media streams. The presentation of both a single media stream and multiple streams with or without the effect of delays are considered. We then study the principles of media objects consumption. Least upper bounds on buffer sharing among media objects are defined. These minimum buffering requirements establish a framework for buffer management in the multimedia database system. A prefetching technique which satisfies the minimum buffering requirements will be developed. These principles and technique provide users with the full range of information required to develop a multimedia data presentation environment.

The remainder of this paper is organized as follows. Section 2 introduces the system and data models to be used in this paper. In Section 3, we present the principles of buffer management for individual media streams, both with or without delays. In Sections 4, we discuss buffer consumption for multiple streams. Section 5 presents a buffer allocation and replacement strategy. Concluding remarks and directions for future research are offered in Section 6.

2 The System and Data Models

In this section, we shall introduce the system and data models that will be used in the rest of the paper.

The system architecture under consideration is illustrated in Figure 1. This architecture includes a centralized multimedia database management system (database server), a multimedia database, and a set of clients which access the server. Within the multimedia database management system, a multimedia playback management function is integrated into an object-oriented database system (OODBS). This system is currently being designed and implemented on top of O2, a well-known OODBS. However, the design strategies can be applied to any OODBS environment that support a C++ interface. The system is designed to handle image, audio, and video media objects.

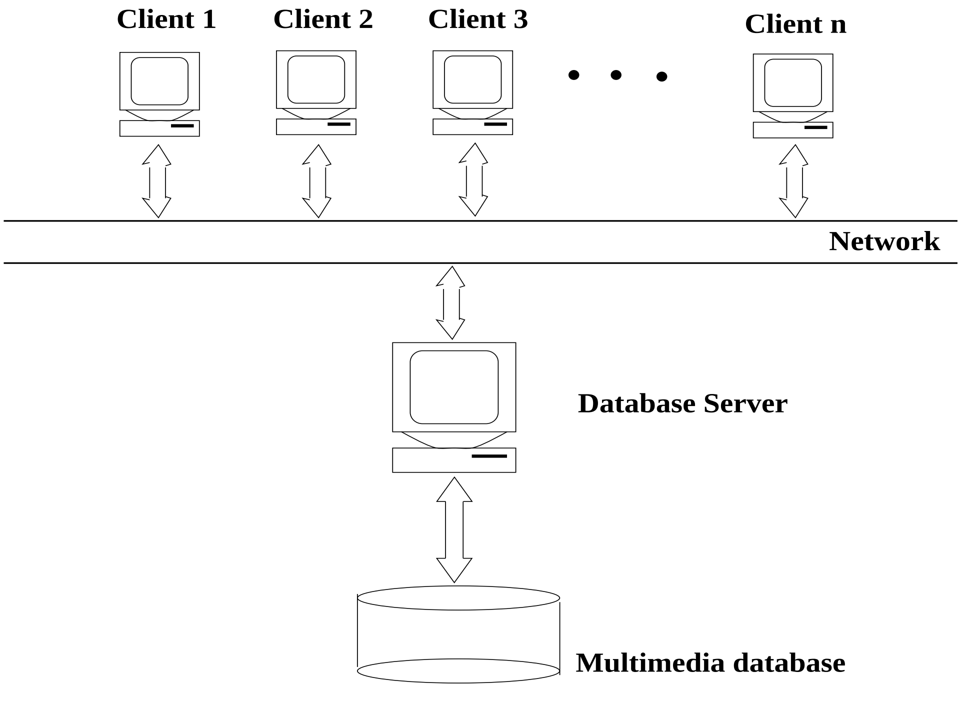

Figure 1: System architecture

In the proposed data model, we assume that each media stream is broken into a set of atomic objects. Higher levels of object classification need not to be considered in this context. 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.

The atomic objects within a media stream are linked together through synchronization time 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 data 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]. In our representation, 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. We use $\langle o, t, \triangle t\rangle$ to denote that object o is to be delivered at time t and will last time period $\triangle t$.

Media objects from different data streams may need to be linked through time constraints to specify their synchronization. For example, in slide presentation applications, an audio object must be played along with a slide object. The temporal relationship between two atomic objects from different media streams may also conform to any of the thirteen temporal relationships described in [All83]. Synchronization requirements may be specified as meta-data or specified in transaction programs. In some cases, the relative time and time interval associated with an atomic object may need to be adjusted to conform with these synchronization requirements.

3 Buffer Model

In this section, we will develop a framework which generates the required start times for media objects. This framework guarantees the continuity of media stream presentation while minimizing buffer utilization. We assume that each media stream is segmented into a set of media objects in the multimedia database. Thus, the proposed principles are developed at the object level rather than at the lower level of media data.

3.1 Start Times of Media Streams

Let $t^m_{l_s}$ be the time at which the loading of data stream m begins and *loading function* $L_m(t, t^m_{l_s})$ be the total number of media objects of m read at time t. Let $t^m_{c_s}$ be the time at which the consumption of data stream m begins and *consuming function* $C_m(t, t^m_{c_s})$ be the total number of media objects consumed at time t. The number of media objects that must be buffered at any given time is then given by

$$B_m(t, t^m_{c_s}) = \begin{cases} 0 & \text{if } t < t^m_{l_s}, \\ L_m(t, t^m_{l_s}) & \text{if } t^m_{l_s} \le t < t^m_{c_s}, \\ L_m(t, t^m_{l_s}) - C_m(t, t^m_{c_s}) & \text{if } t \ge t^m_{c_s}. \end{cases} \tag{1}$$

In (1), $t^m_{l_s}$ is fixed when the loading of data stream m is initiated and $t^m_{c_s} \ge t^m_{l_s}$. We then want to find the smallest $t^m_{c_s}$ such that $B_m(t, t^m_{c_s}) \ge 0$ in the range $t^m_{c_s} \le t \le t^m_{l_f}$, where $t^m_{l_f}$ is the time at which the loading of data stream m is completed. Obviously, if $L_m(t, t^m_{l_s}) \ge C_m(t, t^m_{l_s})$ for any t in the range $t^m_{l_s} \le t \le t^m_{l_f}$, then $t^m_{c_s} = t^m_{l_s}$. Otherwise, let the minimum value for $B_m(t, t^m_{l_s})$, in the range $t^m_{l_s} \le t \le t^m_{l_f}$, be at time t_{min} with $B_m(t_{min}, t^m_{l_s}) = -k$, where $k > 0$.

Suppose that a solution is to begin playback at time x. That is, $B_m(t, x)$ is at least zero for any time $x \le t \le t^m_{l_f}$. If we compare $B_m(t, x)$ with $B_m(t, t^m_{l_s})$ in the range $x \le t \le t^m_{l_f}$, we see that

$$\begin{aligned} B_m(t, x) &= L_m(t, t^m_{l_s}) - C_m(t, x) \\ &= L_m(t, t^m_{l_s}) - C_m(t, t^m_{l_s}) + C_m(t, t^m_{l_s}) - C_m(t, x) \\ &= B_m(t, t^m_{l_s}) + C_m(t, t^m_{l_s}) - C_m(t, x) \end{aligned} \tag{2}$$

This shows us that, in the range $x \le t \le t^m_{l_f}$, $B_m(t, x)$ differs from $B_m(t, t^m_{l_s})$ by $C_m(t, t^m_{l_s}) - C_m(t, x)$. In (2), we know that, in the range $t^m_{l_s} \le t \le t^m_{l_f}$, $B_m(t, t^m_{l_s}) \ge -k$ and $B_m(t, x) \ge 0$. Thus, x must be the minimum start time such that

$$C_m(t, t^m_{l_s}) - C_m(t, x) \ge k. \tag{3}$$

Let us now consider a situation in which the consuming function is linear [GC92]. That is,

$$C_m(t, x) = \begin{cases} 0 & \text{if } t < x, \\ r_c(t - x) & \text{if } t \ge x. \end{cases} \tag{4}$$

where r_c is the consuming rate of stream m. Following (4), in the range $x \le t \le t^m_{l_f}$, we have

$$
\begin{aligned}
r_c(t - t^m_{l_s}) - r_c(t - x) &\geq k, \\
r_c x - r_c t^m_{l_s} &\geq k, \\
x &\geq t^m_{l_s} + \frac{k}{r_c}. \qquad (5)
\end{aligned}
$$

Thus, x should be $t^m_{l_s} + \frac{k}{r_c}$. As demonstrated by Gemmell [GC92], this value of x is also the intersection of $L_m(t, t^m_{l_s})$ with $B_m(t, t^m_{l_s}) + k$. In general, the consuming function may not be a single linear function or even a continuous function. In such a situation, we must ensure that formula (3) always hold true.

We now discuss the effect of presentation delays on the buffering criterion given above. Such delays may be caused by network delays or physical storage delays. In case that the presentation of a media stream is delayed, some data that are expected to be already delivered may still remain in the buffer. If such a delay is recovered by skipping the delayed data, extra buffer space is then needed to store the data that are to be delivered after skip. Thus, the basic buffering criterion given in (2) and (3) must be revised in the event of delays. We assume that two parameters, *maximum delay*, denoted $\triangle d_i^{max}$, and *maximum skip*, denoted $\triangle s_i^{max}$, can be specified for each media stream m_i. Normally, $\triangle d_i^{max}$ is much larger than $\triangle s_i^{max}$. Let $\triangle d_i$ denote the current delay time interval with the presentation of media stream m_i. We then have the following situations:

(1) $\triangle d_i \leq \triangle s_i^{max}$.

(2) $\triangle d_i - \triangle s_i^{max} \leq \triangle d_i^{max}$.

(3) $\triangle d_i - \triangle s_i^{max} > \triangle d_i^{max}$.

In case (1), synchronous presentation can be restored by simply skipping by the interval by which the delayed media streams lag behind. This will lead to an increased buffer requirement as the objects after the skip should also be loaded in the buffer to facilitate a hiccup-free presentation. In case (2), since the delay is larger than the permissible skip, simply skipping the delayed objects may not be applicable. However, this difficulty can be circumvented by a compromise between skipping and pausing. Case (3) does not allow the presentation of the stream as the delay after skip is greater than the allowable delay (or time out) in the presentation.

Since it is not possible to determine the delays in a presentation beforehand, it would be safe to allocate buffers corresponding to the longest delay possible in a presentation. We assume that the recovery from delays takes the optimistic approach discussed above, that is, to skip as much as possible. In order to ensure a hiccup-free presentation, the objects to be displayed after the skip should be pre-loaded in the buffer. Let D_{max} be the maximum size of buffer that is needed to store these extra media objects to compensate for delays. D_{max} is then no larger than $r_c \times (\triangle d_m^{max} - \triangle s_m^{max})$, where r_c is the consuming rate of the stream. Note that D_{max} may vary with the delay recovery approach adopted.

Given D_{max} as the amount of buffer for delay recovery in the presentation of a media stream m, this amount must be added to consumption to determine the start time of delivering the stream. Thus, we can re-define (2) and (3) as

$$\begin{aligned} B_m(t,x) &= L_m(t,t_{l_s}^m) - (C_m(t,x) + D_{max}) \\ &= L_m(t,t_{l_s}^m) - C_m(t,x) - D_{max} \\ &= B_m(t,t_{l_s}^m) + C_m(t,t_{l_s}^m) - C_m(t,x) - D_{max}. \end{aligned} \tag{6}$$

and

$$C_m(t,t_{l_s}^m) - C_m(t,x) - D_{max} \geq k. \tag{7}$$

Comparing formulae (7) with (3), we see that loading delays require additional buffer space. For the linear case, Equation 5 can be re-written as

$$x \geq t_{l_s}^m + \frac{k}{r_c} + \frac{D_{max}}{r_c}. \tag{8}$$

Thus, the start time of delivering stream m is determined by not only the loading time but also the delay and the difference between loading rate and consuming rate. There could be a gap between the loading and display times if delay exists or loading rate is lower than consuming rate.

3.2 Start Times of Media Objects

In the object-oriented database system, we assume that the smallest data unit to be accessed is media object. After the deliver time of stream m is determined, the deliver time of each media object within the stream is also precisely determined. In order to ensure a hiccup-free presentation based on pre-determined time, the loading of each media object must guarantee that there is enough object data to be consumed at the its consuming time. The presentation of each object can thus be divided into two phases: (a) a loading phase and (b) a consumption phase. Let $t_{l_s}^{im}$ be the time at which the loading of object o_{im} of stream m begins and *loading function* $L_{im}(t, t_{l_s}^{im})$ be the amount of object o_{im} of stream m read at time t. Let $t_{c_s}^{im}$ be the time at which the consumption of object o_{im} of stream m begins and *consuming function* $C_{im}(t, t_{c_s}^{im})$ be the amount of object o_{im} consumed at time t. The amount of buffer space that must be allocated for object o_{im} at any given time t is given by

$$B_{im}(t, t_{c_s}^{im}) = \begin{cases} 0 & \text{if } t < t_{l_s}^{im}, \\ L_{im}(t, t_{l_s}^{im}) & \text{if } t_{l_s}^{im} \leq t < t_{c_s}^{im}, \\ L_{im}(t, t_{l_s}^{im}) - C_{im}(t, t_{c_s}^{im}) & \text{if } t \geq t_{c_s}^{im}. \end{cases} \quad (9)$$

In similar manner to the derivation given in Subsection 3.1, we can derive the relationships between the loading and consuming times for each media object similar to Formula 8. Thus, for every object o_{im} in stream m to be successfully presented at time $t_{c_s}^{im}$, it must be loaded into memory at a time given by

$$t_{l_s}^{im} \leq t_{c_s}^{im} - (\frac{k_{im}}{r_l^m} + \frac{D_{max}^m}{r_l^m}). \quad (10)$$

where k_{im} is the amount of object o_{im} that must be loaded before consumption of o_{im} begins, D_{max}^m is the maximum delay in the stream m, and r_c^m is the consumption rate of stream m. Depending on the loading delays and the amount of data that has to be loaded, $t_{l_s}^{im}$ can belong to any of the time periods in which previous media objects are consumed.

4 Buffer Consumption

This section will develop a framework which generates required minimum buffer space for media objects. The framework guarantees both continuity and synchrony in the presentation of one or more media streams. Least upper bounds on media objects buffering for various synchronization situations are presented.

In the analysis above, multiple streams are simultaneously supported by assuming that, for each stream m, enough objects of m are loaded into memory to allow smooth consumption of m. Buffers are needed to satisfy this criterion. It is clear from (1) and (7) that the size of buffer at any point in a time interval $[a, b]$ depends on external factors and cannot be predetermined. This problem may be circumvented by allocating maximum buffer required for each object presented in $[a, b]$.

We will limit the analysis to a case where both the loading function L_{im} and the consuming function C_{im} of object o_{im} of stream m are linear. The underlying assumption in the case of the loading function is that the seek time is much less than the load time. Thus, L_{im} can be approximated as a linear function represented as

$$L_{im}(t, t_{l_s}^{im}) = \begin{cases} 0 & \text{if } t < t_{l_s}^{im}, \\ r_l^{im}(t - t_{l_s}^{im}) & \text{if } t \geq t_{l_s}^{im}. \end{cases} \quad (11)$$

where r_l^{im} is the loading rate of object o_{im} of stream m. A linear case of the consuming function is shown in (4). Without loss of generality, let us assume that the loading rate of all objects in stream m is constant and that the consumption rate of all objects in stream m is also constant;

that is, $\forall i \; 1 \leq i \leq n_m$,

$$\begin{aligned} r_l^{im} &= r_l^m, \\ r_c^{im} &= r_c^m. \end{aligned} \tag{12}$$

However, objects from different streams need not have uniform loading or consumption rates. For a detailed analysis of buffer consumption with streams with identical consumption rates, the reader is referred to [NY94].

Clearly, if the loading rate of a stream m is greater than the consumption rate, all objects in stream m attain their maximum buffer requirement at the end of loading phase, that is, at $t = t_{l_f}^{im}$, provided $t_{l_f}^{im} \in [a, b]$. In contrast, if the consumption rate is higher than the loading rate, then all objects in stream m attain their maximum buffer requirement at the beginning of consumption phase, at time $t = t_{c_s}^{im}$, given $t_{c_s}^{im} \in [a, b]$. Let us denote the time at which object o_{im} in stream m attains its maximum buffer requirement as t_{max}^{im}. If $t_{max}^{im} \notin [a, b]$, the maximum buffer requirement could occur at one of the bounds of the interval. For the case $r_l^m > r_c^m$, we define the time at which object o_{im} of stream m attains its maximum buffer requirement as

$$t_{max}^{im} = \begin{cases} t_{l_f}^{im} & \text{if } a \leq t_{l_f}^{im} \leq b, \\ a & \text{if } t_{l_f}^{im} < a, \\ b & \text{if } t_{l_f}^{im} > b. \end{cases} \tag{13}$$

On the other hand, if $r_c^m > r_l^m$,

$$t_{max}^{im} = \begin{cases} t_{c_s}^{im} & \text{if } a \leq t_{c_s}^{im} \leq b, \\ a & \text{if } t_{c_s}^{im} < a, \\ b & \text{if } t_{c_s}^{im} > b. \end{cases} \tag{14}$$

Given that each object o_{im} in stream m needs the maximum buffer space at $t = t_{max}^{im}$ in the interval $[a, b]$, the maximum buffer requirement for all objects presented in the interval is simply the sum of all maximum buffers over all the objects presented in interval $[a, b]$. Let B_{tot}^m denote the maximum buffer requirement over all the objects in stream m that are presented in the interval $[a, b]$. It is given by

$$B_{tot}^m = \sum_{i=1}^{n_m} B_{im}(t_{max}^{im}, t_{c_s}^{im}), \tag{15}$$

where n_m is the total number of objects in stream m in the interval. Therefore, the total buffer requirement over all the streams is

$$\begin{aligned} B_{tot} &= \sum_{j=1}^{n} B_{tot}^j, \\ &= \sum_{j=1}^{n} \sum_{i=1}^{n_j} B_{ij}(t_{max}^{ij}, t_{c_s}^{ij}). \end{aligned} \tag{16}$$

where n is the total number of streams and n_j is the total number of objects in stream j in the interval.

Thus far, we have analyzed buffer requirements for the presentation of multiple streams in the presence of delays. We will now describe buffer sharing and show its advantages vis-a-vis buffer consumption and buffer utilization. As defined in (16), the total buffer requirement of n media streams is based on the assumption that each stream i occupies the sum of maximum buffers required for all objects o_{im} presented in each interval.

Figure 2 shows the buffer requirement in two intervals $[t_0, t_1]$ and $[t_1, t_2]$ of a presentation involving two streams m_1 and m_2. Assume that display of objects in m_1 can be overlapped (as in images) while delivery of objects in m_2 is sequential (as in audio). Within the interval $[t_0, t_1]$, two objects from stream m_1 (o_{11} and o_{21}) and one object from stream m_2 (o_{12}) are presented, while in the interval $[t_1, t_2]$, one object from each stream is presented (o_{31} and o_{22}). From Figure 2, it is clear that stream m_1 does not require $B_{11}(t^{11}_{max}, t^{11}_{c_s}) + B_{21}(t^{21}_{max}, t^{21}_{c_s})$ buffer size at all times in the interval $[t_0, t_1]$, while stream m_2 does not require $B_{21}(t^{21}_{max}, t^{21}_{c_s})$ buffer size at all times in the interval $[t_0, t_1]$. Thus, allowing buffer sharing between the streams would be a simple way to enhance buffer utilization and lower buffer consumption.

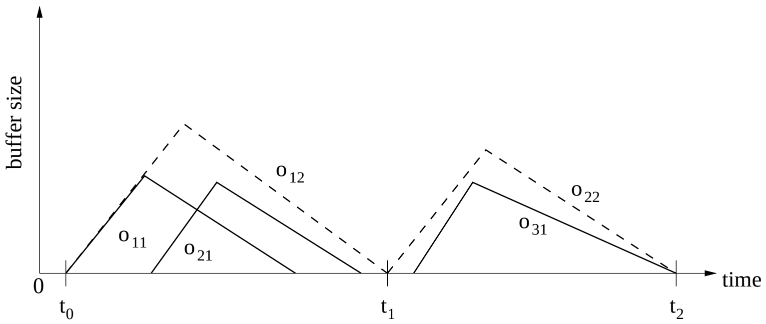

Figure 2: Buffer sharing with 2 streams.

A full analysis of buffer sharing would involve finding the maximum buffer requirement applicable to all the objects presented in an interval. This maximum is not necessarily the sum of the maximum buffers over all the objects given by (16). We begin by defining a piecewise linear, non-negative function F as

$$F(t) = \begin{cases} 0 & \text{if } t < t_0, \\ c_1(t - t_0) & \text{if } t_0 \leq t < t_1, \\ c_2(t - t_2) & \text{if } t_1 \leq t \leq t_2, \\ 0 & \text{if } t > t_2 \end{cases} \tag{17}$$

where c_1 is a positive constant and c_2 is a negative constant. Note that the definition of F is similar to the definition of the buffer function B_{im} in (1) with $t_0 = t^{im}_{l_s}$, $t_1 = t^{im}_{l_f} = t^{im}_{c_s}$ and $t_2 = t^{im}_{c_f}$. The following theorem is useful in determining the maximum buffer requirement for all objects in each interval.

Theorem 1 *Let $F_1(t)$, $F_2(t)$, ..., $F_n(t)$ be n functions of the form defined in (17). Furthermore, let $f_i(t) = F_i(t)$ for $a \le t \le b$. Assume that $f_1(t)$, $f_2(t)$, ..., $f_n(t)$ attain their maximum at $t^1_{\max}$, $t^2_{\max}$, ..., $t^n_{\max}$ respectively. Let*

$$g(t) = f_1(t) + f_2(t) + \cdots + f_n(t).$$

Then, $\max(g)$ *occurs at one of* $\{t^i_{\max} \mid 1 \le i \le n\}$*. More precisely,* $\max(g) = \max\{g(t^i_{\max}) \mid 1 \le i \le n\}$

Proof. Let t_0 be a point in $[a, b]$ where no function $f_i(t)$ has a maximum (see Figure 3). Let S denote the sum of the slopes of functions that have a non-zero value at t_0, that is, $S = \sum_{i, f_i(t_0) \ne 0} \left.\frac{df_i}{dt}\right|_{t=t_0}$. If $S \ge 0$, then there exists at least one function f_i whose slope is positive at t_0. Let t_1 be the first point to the *right* of t_0 at which one of the functions with positive slope or one of those functions that were zero at t_0 attains its maximum. We have

$$\begin{aligned} g(t_1) - g(t_0) &= \sum_i f_i(t_1) - \sum_i f_i(t_0) \\ &= \sum_i f_i(t_1) - f_i(t_0) \end{aligned}$$

Using $\left.\frac{dh}{dx}\right|_{x=x_i} = \frac{h(x_1)-h(x_2)}{x_1-x_2}$ for a linear function $h(x)$ and for any $x_i \in [x_1, x_2]$, we have

$$\begin{aligned} \frac{g(t_1) - g(t_0)}{t_1 - t_0} &= \sum_i \frac{f_i(t_1) - f_i(t_0)}{t_1 - t_0} \\ &= \sum_{j, f_j(t_0) \ne 0} \left.\frac{df_j}{dt}\right|_{t=t_0} + \sum_{k, f_k(t_0) = 0} \frac{f_k(t_1) - f_k(t_0)}{t_1 - t_0} \end{aligned}$$

The first term equals $S(\ge 0)$ and by choice of t_1 the second term is positive. Therefore,

$$g(t_1) - g(t_0) \ge 0$$

or,

$$g(t_1) \ge g(t_0)$$

If $S < 0$, by the same argument, it can be shown that $g(t_1) \ge g(t_0)$ for the first point t_1 to the *left* of t_0 at which one of the functions with a negative slope at t_0 or one of those functions that were zero at t_0 attains its maximum. $\square$

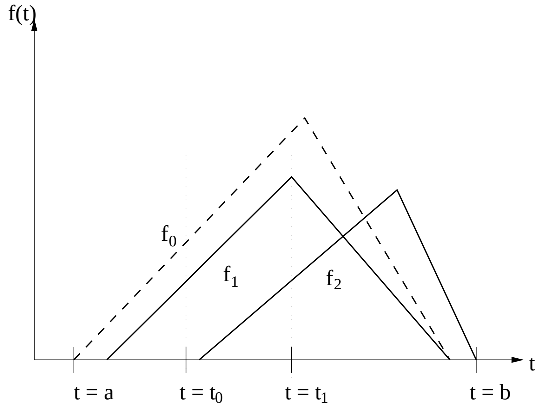

Figure 3: Finding the maximum value of a set of functions.

Corollary 1 *The maximum buffer requirement over all the objects in interval [a, b] occurs at a time when at least one of the objects attains its maximum buffer requirement in that interval.*

We know that the maximum buffer consumption for any object o_{im} in media stream m occurs at $t = t^{im}_{max}$. By Corollary 1, the maximum buffer space in the interval $[a, b]$ would be

$$B_{shar} = \max\{\sum_{j=1}^{n}\sum_{i=1}^{n_j} B_{ij}(t^{kl}_{max}, t^{ij}_{c_s}) \mid 1 \leq l \leq n,\ 1 \leq k \leq n_l\} \tag{18}$$

Theorem 2 *Let the total buffer space required in the interval $[a, b]$ without buffer sharing be B_{tot} given by (16). Then, $B_{shar} \leq B_{tot}$. Moreover, buffer size B_{shar} is necessary.*

Proof. Let the maximum in (18) occur at $l = q, (1 \leq q \leq n)$ and $k = p, (1 \leq p \leq n_q)$. Obviously, the buffer space required at $t = t^{pq}_{max}$ cannot be less than B_{shar}. Hence, B_{shar} is necessary. We have

$$B_{shar} = \sum_{j=1}^{n}\sum_{i=1}^{n_j} B_{ij}(t^{pq}_{max}, t^{ij}_{c_s}).$$

Consider the objects in stream m. Let us denote the total buffer requirement for stream m as B^m_{shar}. We have,

$$\begin{aligned} B^m_{shar} &= B_{1m}(t^{pq}_{max}, t^{1m}_{c_s}) + B_{2m}(t^{pq}_{max}, t^{2m}_{c_s}) + \cdots + B_{n_m m}(t^{pq}_{max}, t^{n_m m}_{c_s}), \\ &\leq B_{1m}(t^{1m}_{max}, t^{1m}_{c_s}) + B_{2m}(t^{2m}_{max}, t^{2m}_{c_s}) + \cdots + B_{n_m m}(t^{n_m m}_{max}, t^{n_m m}_{c_s}), \\ &= B^m_{tot} \end{aligned}$$

Similarly, we can show that

$$B^i_{shar} \leq B^i_{tot},\ 1 \leq i \leq n$$

or,

$$\sum_{i=1}^{n} B^i_{shar} \leq \sum_{i=1}^{n} B^i_{tot},$$
$$B_{shar} \leq B_{tot}.$$

□

Example 1 *Consider a presentation involving two media streams m_1 and m_2. Let the buffer consumption profile for both streams be as shown in Figure 2. Table 1 lists the specifications for both streams. Since the loading rate is greater than the consumption rate for both streams, $t_{l_s} = t_{c_s}$*

Streams	m_1			m_2	
Objects	o_{11}	o_{21}	o_{31}	o_{12}	o_{22}
Loading rate, r_l (kB/s)	1000	1000	1000	1000	1000
Consumption rate, r_c (kB/s)	240	240	240	120	120
Loading start, t_{l_s} (s)	0.2	1.2	4.8	0.2	4.5
Loading end, t_{l_f} (s)	1.2	2.8	6.0	2.0	6.4
Consumption start, t_{c_s} (s)	0.2	1.2	4.8	0.2	4.5

Table 1: Loading and consumption parameters

for all objects and the maximum buffer requirement for each object occurs at the end of the loading phase; that is , at $t = t^{im}_{l_f}$ for object o_{im} of stream m. By (16), the maximum buffer requirement for stream m_1 in interval $[t_0, t_1]$ is $B_{11} + B_{21}$ and is equal to $(760 \times 1.0 + 760 \times 1.6) = 1976$ kB. The maximum buffer requirement for stream m_2 in $[t_0, t_1]$ is $B_{12} = 880 \times 1.8 = 1584$ kB. The total buffer requirement is 3560 kB. Similarly, in $[t_1, t_2]$, m_1 requires 912 kB while m_2 requires 1672 kB and the total buffer requirement is 2584 kB.

In contrast, by (18), the maximum buffer requirement with buffer sharing for streams m_1 and m_2 in $[t_0, t_1]$ occurs precisely at one of $t = \{1.2s, 2.0s, 2.8s\}$. The total buffer requirement for objects o_{11}, o_{21}, and o_{12} in $[t_0, t_1]$ at $t = 1.2s$ is $(760 + 880 + 0) = 1640$ kB. Similarly, at $t = 2.0s$, the total buffer requirement is $(1584 + 608 + 368) = 2560$ kB, while at $t = 2.8s$, the total buffer requirement is $(176 + 1488 + 1216) = 2880$ kB. The maximum buffer requirement in the interval is $\max\{1640, 2560, 2880\} = 2880$ kB. In $[t_1, t_2]$, the maximum buffer requirement with buffer sharing is be found to be 2232 kB. Thus, buffering sharing reduced buffer consumption by 19% in $[t_0, t_1]$ and by 14% in $[t_1, t_2]$.

4.1 Buffer Consumption with Synchronization

We will now apply the basic principles of buffer consumption discussed above to synchronized presentation of multiple media streams in the event of delays.

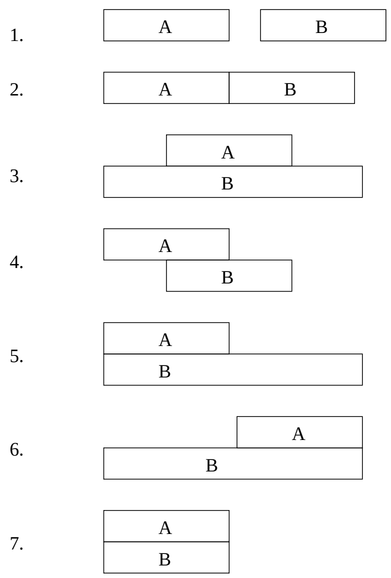

Figure 4: Typical temporal relationships between objects.

We define a *synchronization point* to be a point held in common by all participating media objects in synchronized streams. Thus, synchronization points specify nodes in media streams at which the synchrony of presentation must be checked and maintained. In general, the temporal relationship between two atomic objects from different media streams may conform to any of the thirteen temporal relationships described in [All83], which is demonstrated in Figure 4. We specifically introduce three types of synchronization points among media objects; these are *intra*-synchronization point, *inter*-synchronization point, and *middle*-synchronization point. Intra-synchronization points specify sequential time-constraints among two media objects. Inter-synchronization points specify exact parallel start and end relationships among objects. Middle-synchronization points specify the situations that one object must start in the middle of another object. For example, in Figure 4, cases 1 and 2 have intra-synchronization points between objects A and B; cases 5, 6, and 7 have inter-synchronization points between objects A and B; and cases 3, 4, 5, and 6 have middle-synchronization points between objects A and B. The detailed discussion on synchronization can be found in [ZG95a].

All objects that exist at an inter-synchronization point must either start or finish consumption at that point, that is, $t_{c_s}^{jm} = t_{c_s}^{kn}$ or $t_{c_f}^{jm} = t_{c_f}^{kn}$ for all objects o_{jm} and o_{kn}, $m \neq n$, that exist at the synchronization point. These constraints could lead to one of the following presentation in an interval $[t_i, t_{i+1}]$:

1. All objects begin their consumption at the same time t_i but complete consumption at different times. In this instance, the inter-synchronization point is set at $t_s = t_i$ in the interval $[t_i, t_{i+1}]$.

2. All objects are constrained to finish consumption at the same time but do not start consump-

tion concurrently. In such a case, the synchronization time t_s would equal the upper bound of the interval t_{i+1} and there would be no constraint on the lower bound.

3. All objects start consumption at t_i and end consumption at t_{i+1}. In this instance, two inter-synchronization points are fixed in the interval $[t_i, t_{i+1}]$, one at t_i and the other at t_{i+1}.

The above three cases can occur under one of the following conditions: (a) $r_c^m > r_l^m, \forall m(1 \leq m \leq n)$; (b) $r_l^m > r_c^m, \forall m(1 \leq m \leq n)$; and (c) $r_c^m > r_l^m, \exists m(1 \leq m \leq n)$. Figure 5 shows the above three cases under condition (a). Under condition (a), $t_{l_s}^m < t_{c_s}^m, \forall m(1 \leq m \leq n)$. Therefore, the maximum buffer requirement occurs at time $t = t_{c_s}^m$. Condition (b) implies $t_{l_s}^m = t_{c_s}^m, \forall m(1 \leq m \leq n)$ provided there are no delays in the presentation of objects. Under such a condition, the maximum buffer requirement would occur at $t = t_{l_f}^m$, $t_i \leq t \leq t_{i+1}$. Under condition (c), since all the objects do not start their consumption at their maximum or minimum buffer requirement, the maximum could occur at any point inside the interval. If all objects in the interval $[t_i, t_{i+1}]$ are participating at a inter-synchronization point, we have the following result:

Corollary 2 *Given that the bounds of the interval $[t_i, t_{i+1}]$ are inter-synchronization points involving all objects presented in the interval,*

1. *If $r_c^m > r_l^m, \forall m, 1 \leq m \leq n$, the maximum buffer requirement in the interval occurs at one of the bounds and is equal to $B_{inter} = \sum_j \sum_i B_{ij}(t_{max}^{ij}, t_{c_s}^{ij})$. In contrast, if all streams in the interval do not participate at the inter-synchronization points, the maximum buffer requirement in $[t_i, t_{i+1}]$ is B_{shar}.*

2. *If $r_c^m < r_l^m, \forall m, 1 \leq m \leq n$, the maximum buffer requirement in the interval occurs at one of the points inside the interval and is equal to B_{shar}.*

3. *If $r_c^m < r_l^m, \exists m, 1 \leq m \leq n$, then the maximum buffer requirement in the interval occurs at one of the points inside the interval and is equal to B_{shar}.*

Two types of intra-synchronization points occur within a stream. In the first type, the consumption of an object o_{im} begins immediately after the preceding object o_{i-1m} is consumed. In other words, $t_{c_f}^{i-1m}$ of object o_{i-1m} equals $t_{c_s}^{im}$ of object o_{im}. With intra-synchronization points of the second type, the consumption of an object o_{im} follows after a time interval $\triangle t$ after the preceding object o_{i-1m} is consumed. As before, we have the following conditions: (a) $r_c^m > r_l^m$ for stream m; and $r_c^m < r_l^m$ for stream m. Figure 6 illustrates intra-synchronization under condition (a). Under condition (a), the maximum buffer requirement occurs at the one of the bounds of the interval while under condition (b) the maximum occurs at a point inside the interval and is given by the following result:

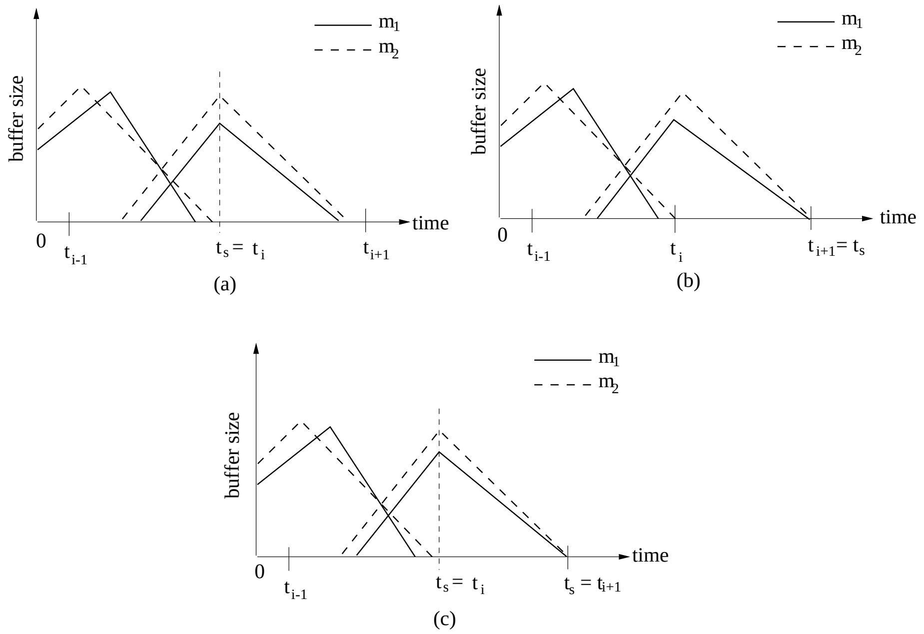

Figure 5: Buffer consumption in the presence of inter-stream synchronization. (a) concurrent start of consumption; (b) concurrent end of consumption; and (c) concurrent start and finish of consumption.

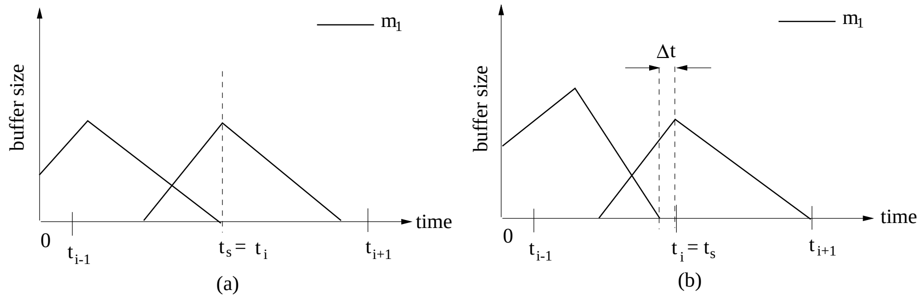

Figure 6: Buffer consumption in the presence of intra-stream synchronization. (a) concurrent start and finish of consumption of two consecutive objects; (b) delay between the start and finish of consumption of two consecutive objects.

Corollary 3 *Given that the bounds of the interval* $[t_i, t_{i+1}]$ *are intra-synchronization points involving all objects* $\in$ *stream* m *that are presented in the interval,*

1. *If* $r_c^m > r_l^m$ *the maximum buffer requirement over all the objects occurs at one of the bounds of the interval and is equal to* $B_{intra} = \max\{B_{im}(t_{max}^{jm}, t_i), B_{im}(t_{max}^{km}, t_{i+1}\}$ *for some objects* o_{jm} *and* o_{km} *participating at the intra-synchronization points* t_i *and* t_{i+1}*, respectively.*

2. *If* $r_c^m < r_l^m$ *the maximum buffer requirement occurs at a point inside the interval excluding the bounds and is equal to* B_{shar}*.*

Middle-synchronization points can involve more than one media stream. A media stream m with objects $o_{im}, (1 \leq i \leq n_m)$ has a middle-synchronization point if there exists another media stream l with objects $o_{jl}, (1 \leq j \leq n_l)$ such that $t_{l_s}^{km} < t_{l_s}^{pl} < t_{l_f}^{km}$ or $t_{l_s}^{pl} < t_{l_s}^{km} < t_{l_f}^{pl}$ for some objects $o_{km} \in m$ and $o_{pl} \in l$. In Figure 7, t_s is a middle-synchronization point where the second object of stream m_1 starts consumption after the first object of stream m_1 starts consumption but before the first object finishes consumption. Note that Figure 7 illustrates middle-synchronization under the condition $r_l^m > r_c^m, \forall m (1 \leq m \leq n)$.

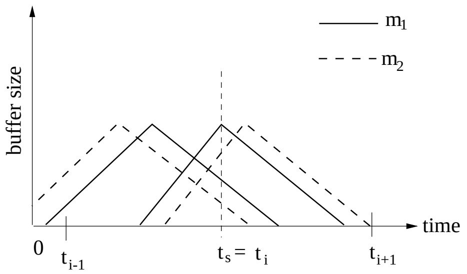

Figure 7: Buffer consumption in the presence of middle-synchronization.

Corollary 4 *Given that the bounds of the interval* $[t_i, t_{i+1}]$ *are middle-synchronization points, the maximum buffer requirement for the interval is* B_{shar} *for both conditions,* $r_c^m > r_l^m, \forall m (1 \leq m \leq n)$ *and* $r_c^m < r_l^m, \forall m (1 \leq m \leq n)$*.*

5 A Prefetching Strategy

In this section, we propose a prefetching technique which is based on the principles discussed above and satisfies the requirement of continuity and synchrony in the presentation of multimedia objects while minimizing the number of buffer faults.

In order to provide support for buffering based on the principles discussed in Sections 3 and 4, a multimedia database system must provide efficient buffer *allocation* and *replacement* strategies. One of the aims of an allocation strategy is to reduce the number of *buffer faults* and thereby reduce delays in the presentation. A buffer fault occurs when the object needed for display is not in the buffer. The main goal of a replacement strategy is to replace those buffers that are least unlikely to be referenced during a certain time interval. Certainly, an easy way to circumvent the problem of buffer faults is to always allocate sufficient buffer space, but this could lead to very low buffer utilization. On the other hand, reducing the buffer space could lead to a large number of buffer faults.

In our strategy, we make use of information gathered at the presentation level to efficiently prefetch objects prior to their presentation. A presentation is characterized by a list of tuples $\langle o_{im}, t_{c_s}^{im}, \triangle t_{im} \rangle$ that represent the objects over all streams in the presentation. Each media object o_{im} is associated with a START event, corresponding to $t_{c_s}^{im}$, and an END event corresponding to $t_{c_f}^{im} = t_{c_s}^{im} + \triangle t_{im}$. All START and END events are classified into layered GROUPs based on the time constraints pertaining to the events. Moreover, all START events in GROUP$_i$ $(1 \leq i \leq n)$ are invoked simultaneously; all END events in GROUP$_i$ $(1 \leq i \leq n)$ are invoked simultaneously; all START events in GROUP$_i$ $(1 \leq i \leq n)$ can be invoked only after all END events in the same GROUP have terminated; and events in GROUP$_i$ always have a higher priority over events in GROUP$_{i+1}$. Thus, each GROUP includes all events occurring at the same time in the presentation and corresponds to a synchronization point in the presentation. Enforcing synchronous delivery of objects at every synchronization point will generate a *synchronous* presentation. See [ZG95b] for more details.

The presentation is segmented into time intervals corresponding to the synchronization points. Thus, interval $[t_0, t_1]$ is constructed from the first two synchronization points p_0 and p_1, occurring at $t = t_0$ and $t = t_1$, respectively. The lower bound of the first interval $[t_0, t_1]$ is determined dynamically by calculating the actual start of consumption of the object o_{im} with the lowest $t_{c_s}^{im}$, using (5). The upper bound of $[t_0, t_1]$ and the bounds of the rest of the intervals are adjusted by an amount equal to $t_{c_s}^{im} - t_{l_s}^{im}$. The display of objects and their retrieval are interleaved to reduce memory requirements and latency in the presentation. Depending on the maximum delay and the loading and consumption rates of stream m, an object o_{im}, that will be consumed in interval $[t_i, t_{i+1}]$, has to be prefetched in an earlier interval or the same time interval to ensure continuous presentation of the object. To minimize buffer requirements, all objects are not prefetched at the same time. Given an object $\langle o_{im}, t_{c_s}^{im}, \triangle t_{im} \rangle$, we calculate the prefetching time of object o_{im} by the relation,

$$t_{l_s}^{im} = \begin{cases} t_{c_s}^{im} - \frac{r_c^m \triangle t_{im} + D_{max}^m}{r_l^m} & \text{if } r_c^m > r_l^m, \\ t_{c_s}^{im} - \frac{D_{max}^m}{r_l^m} & \text{if } r_c^m < r_l^m. \end{cases} \tag{19}$$

For every object o_{im} prefetching at $t_{l_s}^{im}$ is necessary and sufficient to ensure a *continuous* presentation of object o_{im} beginning at $t_{c_s}^{im}$. These prefetching times are sorted and the objects are placed in the appropriate intervals. Thus, each interval has a list of objects whose prefetching times $\in [t_i, t_{i+1}]$.

In each interval, objects in this list are prefetched at the appropriate times to ensure smooth presentation of the objects. Dealing with a finite buffer size of B_{shar} in each interval necessitates an efficient replacement strategy to minimize buffer faults and ensure a continuous presentation. We will use a variation of the LRU strategy called the Least Recently Displayed (LRD) strategy that replaces the object with the smallest t_{c_f} in the buffer with the new object. In other words, the object that finished consumption the earliest is removed first from the buffer. Using an LRD replacement strategy helps in dealing with delays more gracefully. In the event of delays, the last presented object in the faster stream is most likely to be presented over and over again(object duplication) to correct the skew in presentation. Thus, an LRD strategy guarantees that the most recently displayed objects are always in the buffer.

Algorithm 1
Input: T, the multimedia transaction; B_{max}, the maximum buffer
Segment presentation into n time intervals coinciding with
    the synchronization points in the presentation.
Sort the starting time(t_{c_s}) and ending time(t_{c_f}) for all the
    objects and place them in the appropriate time interval.
Calculate the starting load times(t_{l_s}) for all objects in the
    presentation and place them in the appropriate time interval.
Calculate the maximum buffer requirement in each interval.
buffer_size $\leftarrow 0$
BUFFER $\leftarrow \{\}$
clock $\leftarrow t_0$
for $i = 0$ **to** $n-1$ **do**
    if $B_{shar} > B_{max}$ **then return**
    $S \leftarrow \{o_{kj} \mid t_i \leq t_{l_s}^{kj} \leq t_{i+1} \text{ or } t_i \leq t_{l_f}^{kj} \leq t_{i+1}\}$
        // S is sorted based on time
    foreach $obj \in S$ **do**
        while clock $< t_{l_s}^{obj}$ **do**
            increment clock
        if $t_{l_f}^{obj} > t_{i+1}$ **then**
            $obj' =$ partial obj
        else if $t_{l_s}^{obj} < t_i$ **then**
            $obj' =$ rest of obj
        else $obj' = obj$
        if buffer_size$+|obj'| \leq B_{shar}$
        then BUFFER $\leftarrow$ BUFFER $\cup \{obj'\}$
            buffer_size $\leftarrow$ buffer_size$+|obj'|$
        else replace object obj'' with smallest t_{c_f} in BUFFER with obj'
            buffer_size $\leftarrow$ buffer_size $+|obj'| - |obj''|$
            BUFFER $\leftarrow$ (BUFFER$-\{obj''\}) \cup \{obj'\}$
return

Note that if the prefetching time $t_{l_f}^{im}$ of an object o_{im} is larger than the upper bound of the interval t_{i+1}, partial amount of object o_{im} equal to $r_l^m(t_{i+1} - t_{l_s}^{im})$ is prefetched and the rest of the object $r_l^m(t_{l_f}^{im} - t_{i+1})$ is loaded in the following interval $i+1$. Algorithm 1 summarizes the above scheme. This scheme ensures that the buffer requirement in each interval $[t_i, t_{i+1}]$ is always $\leq B_{shar}$ given in (18). By Theorem 2, B_{shar} size of buffer is necessary and sufficient in each time interval $[t_i, t_{i+1}]$. We have thus proven:

Theorem 3 *Let B^i denote the total buffer size over all the objects presented in interval $[t_i, t_{i+1}]$. Algorithm 1 minimizes the total buffer requirement by minimizing $\max_{0 \leq i \leq n} B^i$ where n is the total number of intervals in the presentation and ensures both continuity and synchronization in the presentation.*

We have shown a buffer prefetching algorithm that maintains synchronization constraints thereby minimizing skew and jitter [LG90a] in a presentation while minimizing the buffer requirements. Furthermore, the algorithm handle delays in the presentation more gracefully by using an LRD replacement strategy for buffer replacement. This allows us to minimize the buffer faults and hence average delays in the presentation, thereby decreasing the additional buffer size required to store the delayed objects and make the presentation continuous.

We have implemented a playout management function in our multimedia database system prototype. This system allows the user to define an application specific presentation script. The service greatly enhances the user's ability to generate acceptable presentations that minimize buffer requirements while maintaining synchrony and continuity in the presentation.

6 Conclusions and Future Work

In this paper, we have introduced a framework for buffer management in multimedia database systems. This framework includes the principles of minimum consuming start times of media objects following their loading times, while guaranteeing the continuity and synchrony of the presentation of media objects, and the optimal buffer space requirements for media objects based on the least upper bounds on media objects consumption. A prefetching technique which satisfies the proposed buffer management principles is presented. The proposed principles and technique have been implemented in a prototype that integrates the functionality of multimedia playback management into an object-oriented database system. This implementation is conducted using O_2, a well-known object-oriented database system.

This paper has not addressed issues relevant to buffer management in event of user interactions. In such a context, synchronous presentations can also be restored by skipping/pausing individual media streams similar to the approaches that handle delays. Given both maximum permitted delivery delay and skipping intervals for the the participating media streams, a buffer management strategy is needed to provide the most efficient buffer consumption and allocation while supporting user interaction such as fast forward and backward, skip, and pause. A precise formulation of such a framework is to be investigated.

References

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

[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.

[GC92] Jim Gemmell and Stavros Christodoulakis. Principles of delay-sensitive multimedia data storage and retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, January 1992.

[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.

[MKK95] Frank Moser, Achim Kraib, and Wolfgang Klas. L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flow in a Multimedia DBMS. In *Proceedings of the 21th VLDB conference*, Zurich, Swizerland, 1995.

[NY94] R. T. Ng and J. Yang. Maximizing Buffer and Disk Utlilizations for News On-Demand. In *Proceedings of the 20th VLDB Conference*, pages 451–462, Santiago, Chile, 1994.

[TK95] Heiko Thimm and Wolfgang Klas. Playout Management – An Integrated Service of a Multimedia Database Management System, 1995. (Technical Report, GMD-IPSI).

[ZG95a] Aidong Zhang and Sreenivas Gollapudi. Multimedia Transaction Management in Database Systems. Technical Report CS-TR-95-44, Dept. of Computer Science, State University of New York at Buffalo, October 1995.

[ZG95b] Aidong Zhang and Sreenivas Gollapudi. Multimedia Transaction Management in Multimedia Database Systems. Technical Report CS-TR-95-44, State University of New York at Buffalo, October 1995.