# Retrieval Schedules Based on Resource Availibility and Flexible Presentation Specifications [*]

**K. Selçuk Candan**[†] **B. Prabhakaran**[‡] **V.S. Subrahmanian**[§]

**Abstract**

A distributed multimedia document presentation involves retrieval of objects from the document server(s) and their presentation at the client system. The presentation of the multimedia objects have to be carried out in accordance with the specification of temporal relationships among the objects. The retrieval of multimedia objects from the document server(s) is influenced by the factors such as: temporal specification of objects presentations, throughput offered by the network service provider, and the buffer resources on the client system. Flexibility in the temporal specification of the multimedia document can help in deriving an object retrieval schedule that can handle variations in the network throughput and buffer resources availability. In this paper, we develop techniques for deriving a flexible object retrieval schedule for a distributed multimedia document presentation. The schedule is based on flexible temporal specification of the multimedia document using the difference constraints approach [5]. We show how the derived retrieval schedule can be validated and modified to ensure that it can work with the offered network throughput and the available buffer resources.

## 1 Introduction

A multimedia document consists of different types of media objects that are to be presented at different instants of time for different durations. The time instances and durations of presentations of the objects are specified as either hard or flexible temporal specifications. In the case of hard temporal specification, the time instants and durations of presentations of objects are fixed. In a flexible temporal specification, however, the time instants and durations of presentations of objects are allowed to vary as long as they preserve certain specified relationships. To see this, consider two temporal constraint specifications of the form

[*]This research was supported by the Army Research Office under grant DAAH-04-95-10174, by the Air Force Office of Scientific Research under grant F49620-93-1-0065, by ARPA/Rome Labs contract Nr. F30602-93-C-0241 (Order Nr. A716), and by an NSF Young Investigator award IRI-93-57756. Proofs of all results are contained in the appendix.

[†]Department of Computer Science, University of Maryland, College Park, Maryland 20742. Email: candan@cs.umd.edu.

[‡]Department of Computer Science, University of Maryland, College Park, Maryland 20742. Email: prabha@cs.umd.edu.

[§]Department of Computer Science, Institute for Advanced Computer Studies & Institute for Systems Research, University of Maryland, College Park, Maryland 20742. Email: vs@cs.umd.edu.

- (a) Start showing the image *at* 10am *for* 10 minutes.
- (b) Start showing the image *sometime between* 9.58am and 10.03am and show it till the audio is played out.

The first statement is a *hard* temporal specification, with the time instant and duration of presentation of the image fixed (at 10am and for 10 minutes, respectively). In contrast, the second specification is more flexible in that it allows the start time instant to vary within a range of 5 minutes. A similar flexibility is allowed for the duration of presentation of the object also, by showing the image till the audio is played out. The temporal constraint specification, in other words, helps in the derivation of a *presentation schedule* that describes the starting times and durations of the presentations of the objects composing the multimedia document. Note that if the specifications are hard, the presentation schedule would be the same as the temporal constraint specification. Flexible temporal specifications imply that the presentation of the multimedia document can be flexible, i.e., one can have a *set* of presentation schedules that satisfy the given temporal constraints. Each member of this presentation schedule set describes one possible *view* of the multimedia document. In this work, we deal with flexible temporal constraints.

In a distributed multimedia presentation, the objects composing the document can be dispersed over a computer network. These objects have to be retrieved from their storage places and presented to the user. With the storage place acting as a *server* and the retrieving system as a *client*, the retrieval process is initiated by the client (as opposed to the server just *delivering* the objects following some schedule of its own). Hence, the retrieval process is composed of the following phases:

- Identify a *presentation schedule* that satisfies the (flexible) temporal specification associated with the multimedia document.
- Identify a *retrieval schedule* that specifies the time instants at which the client should make a request to the server(s) for delivering the objects that compose the multimedia document.

Specification of the time instants for retrieving objects from the server as part of the retrieval schedule is carried out by determining the time taken to transfer the object from the server to the client. Consider the temporal constraint specification (b). We can derive a presentation schedule that specifies the start time of presentation of object A as 9.58am. If we know that the delay involved in retrieving the object A from its server is 3 minutes, then the retrieval schedule can be fixed at 9.55am. This retrieval schedule is constrained by the following factors:

1. Throughput (or the bandwidth) of the communication channel between the server and the client.
2. Buffer availability for the retrieved objects.
3. Size of the object(s) that is (are) to be retrieved from the server.
4. Time duration available for retrieval.

Here, the throughput of the communication channel, and the buffer resources are system dependent. The available throughput can vary depending on the type of network and the load on the network. The buffer resources are dependent on their availability in the client system. The last two constraints: size of the objects and the time available for retrieval, are application dependent. The size of the object depends on the type of media as well as the desired *quality* of presentation (section 2.2). For example, an image object may be retrieved as a thumbnail sketch or as a full image. The time available for presentation depends on the derived presentation schedule from the (flexible) temporal constraints specification. The retrieval schedule for a multimedia document presentation has to be derived based on the above four constraints.

**Related Work:** Multimedia authoring and presentation schedule creation are studied by many researchers, such as [2, 5, 15, 16, 18, 19, 23]. Similarly, deriving retrieval schedules for distributed multimedia presentation has also been studied in many works, such as [18, 16, 25, 26, 27, 28, 32]. In [18], the presentation schedule is based on Petri nets description of the temporal specification. This presentation schedule is fixed before the generation of the retrieval schedule. The retrieval schedule is derived by assuming a certain throughput to be provided by the network service provider. Based on the derived retrieval schedule and the assumed network throughput, estimates for the buffer resource requirements on the client system are made. However, the proposed algorithm does not check whether the estimated buffer resources are available or not. Also, it does not handle the variations in the throughput offered by the network service provider. In [16], however, Li et.al. use time-flow graphs to capture interval-based *fuzzy* presentation schedules, and synchronization of independent sources. Their algorithms guarantee that there will be no gaps in the source's schedules. However, they do not address to the issue of constraints on resources such as throughout and buffer. As in [18], in [25, 26, 27, 28], authors use petri net model to describe temporal specifications, and they base the retrieval schedules on the fixed presentation schedules. In [32], Thimm et.al. describe a method which adapts the presentation schedule to the changes in the resource availability by modifying the overall quality of the presentation.

**Our Approach:** In this paper, we developed techniques for deriving flexible object presentation and retrieval schedules for a distributed multimedia document presentation. Our approach is to use flexible temporal constraint specification for deriving a possible presentation schedule [5]. Based on this presentation schedule, we suggest techniques for deriving the retrieval schedule. The derived retrieval schedule is validated by checking whether it satisfies all the system availability constraints. If it does not, we do the following:

1. Modify the retrieval schedule. We try to find a different retrieval schedule which fits into the presnetation schedule at hand.
2. If no such change in the retrieval schedule is possible, then modify the presentation schedule. We identify the portions of the retrieval schedule which do not satisfy the system availabilities. Based on the unsatisfied retrieval schedule, we suggest a feedback for modifying the presentation schedule appropriately.

3. If everything else fails, then modify the quality of presentation. Such a modification is possible in the case of certain objects like *gif* formatted images, etc. Also, the reduced quality of presentation should be acceptable to the viewer.

## 2 Multimedia Document Presentation

A multimedia document is composed of objects that are to be presented at different time instances and for different time durations. Based on the way the objects are to be retrieved from the server(s), we classify them as:

- **Atomic Objects:** These objects need to be received at the client side as a whole before the presentation starts. For example, still image files are atomic objects.
- **Stream Objects:** These objects can be presented to the viewer as soon as some portion of them is received. The rest of the object is then continuously fed to the viewer. Video and audio objects are generally considered to be stream objects. However, in systems which cannot handle display-while-retrieving operation, video can be retrieved as a whole, and be displayed afterwards. In such systems, video objects must be considered as atomic objects.

Atomic objects must be present in the buffers of the client as a whole at the start of their presentations. Then, they can be consumed from the buffers during the presentation (as in the case of atomic-video objects) or can be kept in the buffers as a whole till the end of the presentation (as in the case of the still images). We use $\mathrm{c}_a(o)$ to denote the consumption rate of an atomic object $o$ from the buffers. Note that for objects like still images $\mathrm{c}_a(o) = 0$.

Stream objects do not need to be delivered as a whole before their presentations. However, in order to reduce jitter and in to smooth their display, such objects usually require some fraction to arrive at the display site before the start of their presentations. In this paper, we use $b_{init}(o)$ to denote the size of this fraction for a stream object $o$, and we use $\mathrm{c}_s(o)$ to denote the consumption rate of a stream object $o$ from the buffers. Note that in order to prevent the underflow of the buffer $B(o)$, the consumption rate $\mathrm{c}_s(o)$ must be equal to the average delivery rate (throughput) of the object $o$ (throughput of an object $o$ is denoted as $th(o)$). However, if the network is not capable of providing the consumption rate, then we can reduce the throughput requirement by buffering a larger portion of the object at the client site. This process will be explained in more details in section 7.1.

### 2.1 Flexible Multimedia Presentation

In practical circumstances, one may encounter a situation where the derived retrieval schedule cannot be satisfied. For example, the network service provider might offer a very low throughput for the application. The retrieval schedule based on the throughput offered by the network service provider may overshoot the buffer availability on the client. Hence, the derived retrieval schedule cannot be used for the multimedia presentation. However, we may be able to modify the retrieval schedule by relaxing (one or both of) the application dependent constraints. For example, one can reduce the size of the multimedia objects to be retrieved by reducing the quality of the presentation (if the reduced quality is acceptable to the viewer). If the reduction in the quality of the presentation is not acceptable, then we can modify the time duration for retrieving the objects. This can be performed by selecting a different presentation schedule satisfying the given set of (flexible) temporal constraints.

As an example, consider the (previously described) temporal constraint specification (b): start showing the image sometime between 9.58am and 10.03am and show it for a duration of 10 minutes. Let the chosen presentation schedule be such that the image presentation starts at 9.58am. If we find that the retrieval schedule based on this presentation schedule does not satisfy all the system constraints, then we can try another presentation schedule, say: image presentation starts at 10.03am. This change in the presentation schedule gives more time for the retrieval of the object (in this example, it gives 5 more minutes for retrieval). We can derive a new retrieval schedule based on the modified presentation schedule and check whether the new retrieval schedule satisfies all the constraints.

In the above example, we (seem to have) arbitrarily picked up another value for the start of presentation of the image. Instead, we can use feedback from the retrieval schedule to give us an idea of which temporal values can be changed by what factor. In other words, *we can use the flexibility of the multimedia presentation to find other solutions within the application dependent constraints, such as the quality of presentation and the presentation schedule. This new solution for the retrieval schedule is selected in such a way that it can handle the changes in the system dependent constraints such as throughput offered by the network and the buffer resources available on the client.*

## 2.2 Quality of the Presentation

In this paper, we suggest the use of the quality reduction as a method for satisfying the system resources. Note that this method is used only when there is no other way to satisfy the constraints.

The quality of a presentation has two main aspects:

- **Response time of the presentation:** The response time is defined as the amount of time that elapses between the time at which the first object request is issued by the system and the time at which the presentation starts. The smaller this value is, the higher the quality of the presentation.
- **Quality of the objects:** In this paper we assume that a quality is associated with each object, and a reduction in the size of the object is accompanied by a reduction in its quality. Different media types observe different quality reductions when their size is reduced. A table of the following form provides a means by which the system relates quality with reduction in object size:

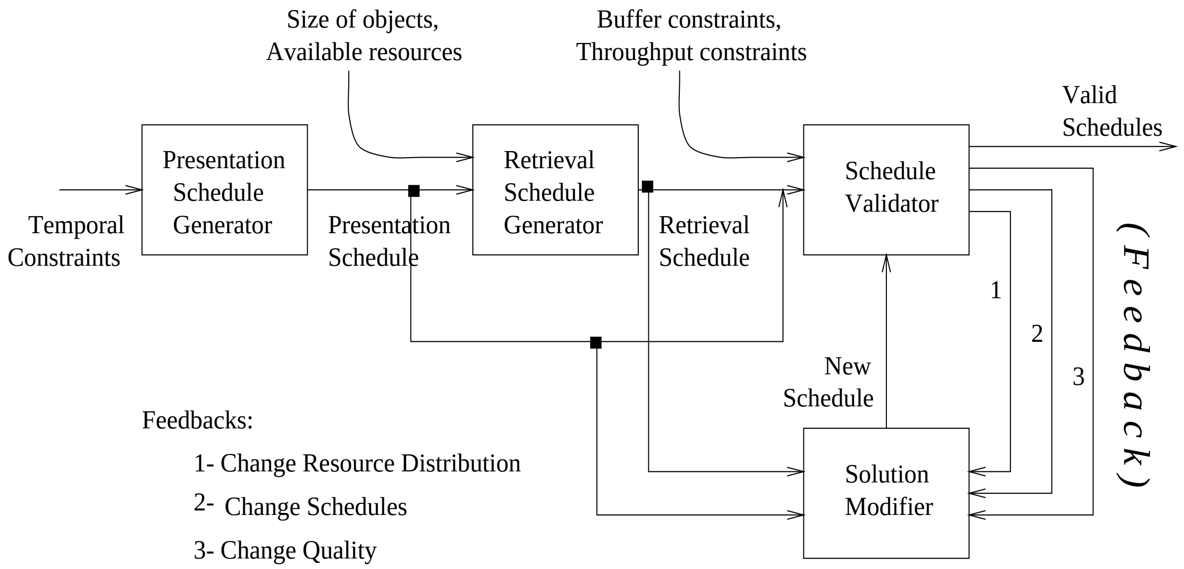


Figure 1: Flexible Multimedia Presentation Architecture

| **Media Type** | **Reduction in size** | **Reduction in quality** |
|---|---|---|
| A | 0.2 | 0.1 |
| A | 0.5 | 0.7 |
| B | 0.1 | 0.8 |
| ... | ... | ... |

Note that a reduction in the quality of an object causes a reduction in the quality of the overall presentation. Hence, it is better to use higher quality objects as long as their sizes do not violate system constraints.

### 2.3 Object Priorities

When there are two or more objects competing for the same system resource, it may be necessary to change the schedules or reduce the qualities of their presentations. Note that if objects $o_1$ and $o_2$ are competing for a resource, then $o_1$, $o_2$, or both may be affected if there is not enough resources to serve them both. We assign priorities to the objects in the presentation to minimize the number objects that will be affected from a resource shortage. The priority of an object $o$ is calculated using the user preferences, the number of other objects whose presentations depend on $o$, and cost of quality change of $o$.

## 3 Flexible Multimedia Presentation Architecture

Figure 1 shows the architecture we propose for a flexible multimedia presentation. The temporal specifications associated with a given multimedia document are used by the Presentation Schedule Generator module to generate a possible presentation schedule. Based on this schedule and other constraints such as available throughput and the size of the objects, the retrieval schedule generator determines a possible retrieval schedule. The schedules are checked by the schedule validator to determine whether all the associated constraints (buffer availability, throughput) are satisfied. If some of the constraints are not satisfied, one or more of the following modifications are made in order to satisfy all the constraints.

- Change the buffer resource distribution.
- Pick a different presentation or retrieval schedule.
- Change the quality of the presentation

In the above modifications, modifying the buffer resource distribution is most desirable while changing the quality of presentation is the least desirable one. Based on this discussion, we can say that the input and output of the flexible multimedia presentation system are as follows.

**Input:** The input to the system consists of

- a set of temporal specifications.
- a list of available system resources: throughput and buffer.
- a list of object sizes and object locations.
- a list of presentation quality requirements.
- a list of object priorities.

**Output:** The output is a presentation and a retrieval schedule which satisfy the input specifications, and requirements.

**Flexible Multimedia Presentation System Components:**

1. **Presentation Schedule Generator:** This component of the system provides a solution to the given temporal specifications. It picks a schedule which satisfies the temporal specifications. In [5], we have described the details of such a temporal constraint solver. We provide a brief overview of the temporal constraint solver in Section 4.
2. **Retrieval Schedule Generator:** The *Retrieval Schedule Generator* takes the presentation schedule, the list of available system resources, and the object sizes, and it outputs a retrieval schedule. Section 5 describes how the retrieval schedule is generated from the listed inputs.
3. **Schedule Validator:** Given a temporal schedule, system constraints, and a retrieval schedule, this module checks the validity of the generated retrieval schedule based on the input constraints. If the schedules are valid with respect to the specified constraints, then the *validator* returns them as the final solution. However, if the schedules do not satisfy the system constraints, this module suggests modifications that can be made to the current solution in

order to satisfy the system constraints. These suggestions are used by the *Schedule Modifier* module to find a modified solution that can satisfy the constraints. Section 6 describes in detail the functionality of the schedule validator module.

4. **Schedule Modifier:** This module modifies the current solution for retrieval and presentation schedules, based on the suggestions made by the Schedule Validator module. The modified solution is given back to the Validator module to check the solution against the system constraints. We discuss the details of the Schedule Modifier in Section 7.2.

   This process of solution-feedback and validation is repeated till a valid schedule is generated. In case a valid schedule cannot be arrived at, then the best schedule found so far can be used as the solution. Objects whose schedules do not satisfy the system constraints can be dropped from the presentation, provided the viewer agrees to it.

## 3.1 Segmented Validation of Schedules

In our approach, we first generate a presentation schedule based on the specified temporal constraints. Then we generate a retrieval schedule for a *segment of time* (the duration of the time segment is chosen based on the implementation requirements). The process of segmented retrieval schedule generation (and validation) is done for the following reasons.

- In the case of a long presentation (say, a 1 hour presentation), system constraints such as throughput and buffer can vary considerably. Hence, an initial schedule generated for the entire presentation may become invalid at a later point in time.
- Creating a complete schedule for the whole presentation can be time consuming. It might lead to a long wait time for the user before the presentation can start. (And after all that, the schedules might become invalid!)

The retrieval schedule for the time segment is then validated with respect to the system constraints. If the retrieval schedule is found valid, then the multimedia document presentation for the validated time segment is started. (Otherwise, we go through the process of solution-feedback to generate another schedule, as discussed above). Then, the retrieval schedule for the next time segment is generated and validated. Hence, we follow a *segmented validation* of the generated schedules.

It should be noted here the segmentation of the presentation is done with respect to the presentation schedules. The retrieval schedule for a segment might fall into the previous segment. This can modify the throughput and buffer requirements. In our approach, we allow these segments to overlap by a chosen time duration, say $t$. For example, if $SE_{i-1}$ denotes the end of segment $i-1$, then $SS_i$, the segment start time of the $i^{th}$ segment will be: $SS_i = SE_{i-1} - t$. While validating one segment, its overlap with the previous segment gets *revalidated*. Hence, it takes care of the overlap of the retrieval schedules.

Figure 2 shows an example of this segmented validation of a one hour presentation. The document is divided into 5 segments of size 12 minutes each, and each of the segments are handled

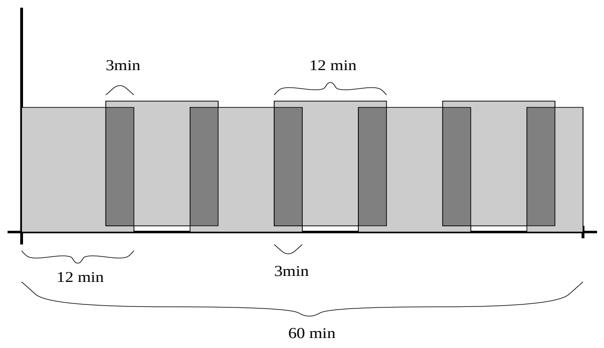


Figure 2: Segmented Validation: An Example

| **Symbol** | **Meaning** |
|---|---|
| $\mathrm{Th}_{max}$ | Maximum throughput available at a communication line. |
| $\mathrm{Th}_{tot}$ | The total throughput required by the objects sharing a communication line. |
| $th(\mathrm{o})$ | The amount of throughput used by object $o$. |
| $\mathrm{Buf}_{max}$ | Maximum buffer available for the objects sharing a communication line. |
| $\mathrm{Buf}_{tot}(t)$ | The total buffer required at time $t$ by the objects sharing a communication line. |
| $buf(o,t)$ | The amount of buffer used by object $o$ at time $t$. |
| $b_{init}(o)$ | The size of the buffer required by object $o$ before the start of its presentation. |
| $st(o)$ | The time at which the display of object $o$ starts. |
| $et(o)$ | The time at which the display of object $o$ ends. |
| $req(o)$ | The time at which the request for the object $o$ is issued by the client. |
| $rec(o)$ | The time at which the first bit of the object $o$ is received at the client. |
| $sz(o)$ | The size of the object $o$. |

Figure 3: Notations and terminology

separately. The consecutive segments have an overlap of three minutes: the schedule of the last three minutes of segments are reprocessed at the beginning of the following segments, as discussed above.

### 3.2 Notation Used in the Paper

The major symbols we use in this paper are explained in figure 3.

## 4 Presentation Schedule Generation

In [5], we developed a framework that supports the creation and incremental modification of multimedia documents. We showed that spatial and temporal specifications can be uniformly described within a small class of the language of real valued linear constraints, called *difference constraints*.

While generalized linear constraints [12] have the form

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \quad \leq \quad b \tag{1}$$

where $a_1, \ldots, a_n, b$ are *rational numbers* (positive and negative), and $x_1, \ldots, x_n$ range over the *real numbers* (positive and negative), difference constraints have the form

$$x_1 - x_2 \quad \leq \quad b. \tag{2}$$

Thus, difference constraints are a special case of linear constraints where:

1. There are only two variables (i.e. $n = 2$ in Equation 1), and
2. One variable has coefficient 1 (i.e. $a_1 = 1$) while the other has coefficient $-1$ (i.e. $a_2 = -1$).

Due to the fact that difference constraints have a very tightly restricted syntactic form, it turns out that they are very easy to solve. Using difference constraints, we showed [5] how it is possible to determine if a given set of media objects can be scheduled in a way that satisfies the desired specification– if no such schedule exists, then this means that the specification demanded by the authors of the document are inconsistent. Our algorithms checked for such inconsistencies. We also showed how an inconsistent set of constraints may be *relaxed* so as to restore consistency [5].

Associated with each object $O$ in a multimedia document $D$, we associate a set, $T_O$, of temporal constraints. As is customary in operations research[12], constraints are constructed from *variables*. In the case of multimedia documents, we associate, with each multimedia object $O$ in the document, the following *temporal variables*:

- $st(O)$: Denotes the start time of the display of the object $O$
- $et(O)$: Denotes the end time of the display of the object $O$
- $req(O)$: Denotes the time when the request for the object $O$ is issued

The last variable ($req(O)$) did not exist in the original framework, but it is easy to enlarge the framework to include this new variable. The framework is also capable of specifying constraints in subobject level (for instance, authors can specify synchronization of two video clips on frame by frame level). However, here, for the sake of simplicity, we will consider only the synchronization of whole objects.

There are four types of temporal constraints:

| | |
|---|---|
| • $\mathcal{T}(o) - t \leq \delta t$ | • $\mathcal{T}(o) - t \geq \delta t$ |
| • $t - \mathcal{T}(o) \leq \delta t$ | • $t - \mathcal{T}(o) \geq \delta t$ |

where:

1. $\mathcal{T}(o) \in \{st(o), st(o)\}$ and
2. $t \in \bigcup_j \{et(o_j), et(o_j)\} \bigcup \{st_p, et_p\}$ and
3. $st_p$ and $et_p$ denote the start and end of the presentation respectively.

**Example 4.1** Let us assume that there exist two objects $o_1$ and $o_2$ that we want to display simultaneously, i.e. we want them to start and finish simultaneously. This requirement can be described using the following constraints:

$$\begin{array}{l} st(o_1) - st(o_2) \leq 0 \\ st(o_2) - st(o_1) \leq 0 \\ et(o_1) - et(o_2) \leq 0 \\ et(o_2) - et(o_1) \leq 0 \end{array} \qquad \square$$

Note that using these constraints, not only can we specify Allen's 13 temporal relationships[1] between events, but also specify more complex quantitative relationships that cannot be expressed in Allen's framework. Suppose $D$ is any document and $T_D$ is the set of temporal constraints associated with $D$. With this set of difference constraints, we may associate a graph $G = (V, E)$ defined as follows:

1. **Vertices:** For each constraint variable $\tau_i$ occurring in the set of difference constraints $T_D$ , $V$ contains a vertex $v_i$ representing that variable. In addition, $V$ contains two special vertices $v_s$ (document "start" node) and $v_e$ (document "end" node).
2. **Edges:** If $\tau_j - \tau_i \leq \delta t$ is a constraint in the set of difference constraints being considered, then $E$ contains an edge from $v_i$ to $v_j$ and the weight associated with this edge is $\delta t$. Furthermore, for each node $v_i$, there is an edge from $v_i$ to $v_s$ with weight 0 and ........ (!!)

Thus, given any document $D$, we have one graph associated with its temporal specifications. In [], we showed that the shortest path solution of this graph results in a schedule that satisfies the temporal specification. Example 4.2 shows how the solution works.

**Example 4.2** Let us assume that there exist two objects $o_1$ (50 seconds) and $o_2$ (40 seconds). We want to display one after the other (i.e. $o_2$ after $o_1$). But, we also want that the display of $o_2$ start within 10 seconds after $o_1$ finishes. This requirement can be described using the following constraints:

$$\begin{array}{l} st(o_1) - et(o_1) \leq -50 \\ et(o_1) - st(o_1) \leq 50 \\ st(o_2) - et(o_2) \leq -40 \\ et(o_2) - st(o_2) \leq 40 \\ et(o_1) - st(o_2) \leq 0 \\ st(o_2) - et(o_1) \leq 10 \end{array}$$

Figure 4 shows the corresponding graph, and the corresponding shortest path solution. At the end, the vertices of the graph has the following values:

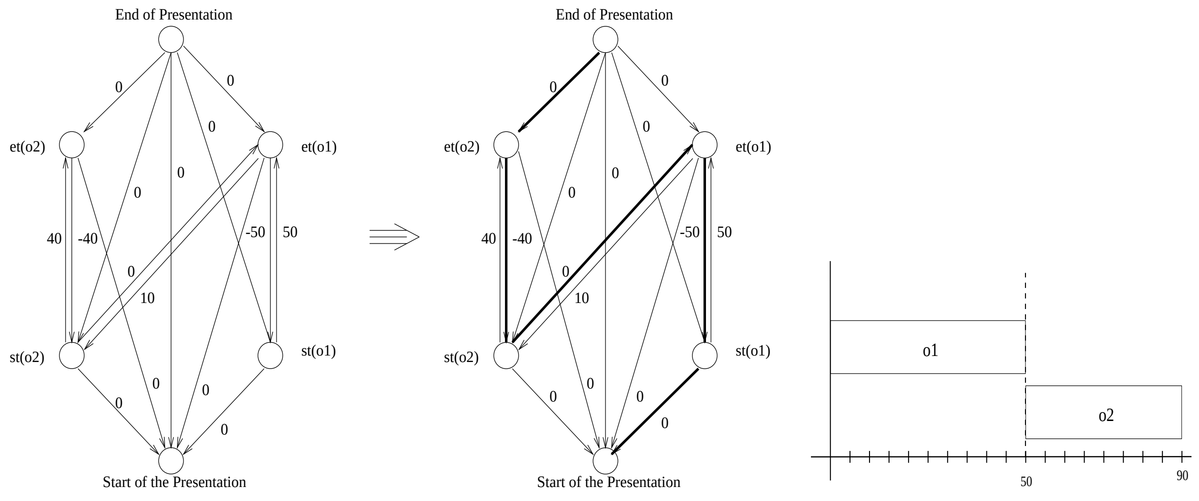


Figure 4: The constraint graph, and the corresponding solution

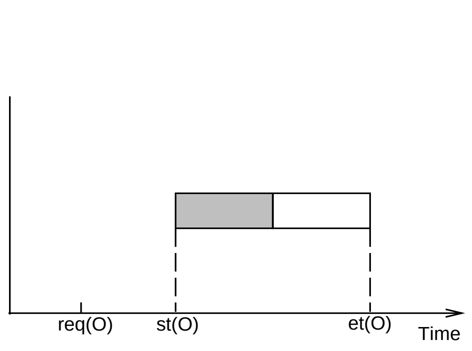
| | |
|---|---|
| End of the presentation = 0 | Start of the presentation = −90 |
| $et(o_2) = 0$ | $st(o_2) = -40$ |
| $et(o_1) = -40$ | $st(o_1) = -90$ |

The corresponding presentation schedule is also shown in figure 4. Note that the time values in the schedule are shifted by 90 to make them positive. □

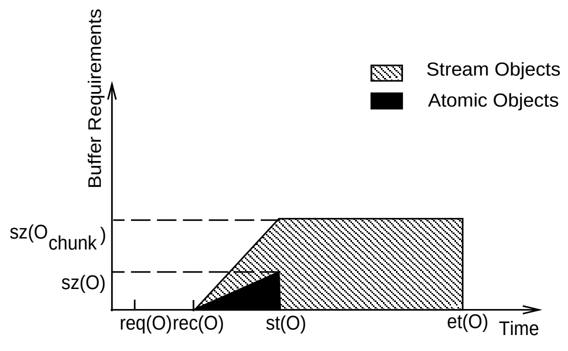
## 5 Retrieval Schedule Generation

Retrieval schedule of a multimedia presentation specifies the time instants at which the client should make requests to the server(s) for delivering the objects that compose the presentation. As discussed earlier, the retrieval schedule is constrained by system dependent factors, such as the available throughput and available buffer resources, as well as by application dependent factors, such as the time duration available for retrieval and the size of the objects. While deriving the retrieval schedule, we make the following assumptions.

- Multiple objects can be retrieved over the same network connection.
- The network provides a maximum throughput $Th_{max}$ for each connection. Hence, this available throughput has to be shared by different objects in case their retrieval from the server has to be done in parallel. This throughput offered by the network service provider can vary with time, depending on the network load.
- The client provides a maximum buffer $Buf_{max}$ for each connection for storing the retrieved objects before their presentation.



(a) Retrieval Schedule for a Single Object



(b) Buffer Requirements for a Single Object Presentation

Figure 5: Single Object Retrieval

- The release of the buffer resources associated with the object presentation depends on the application as well as on the media type to which the object belongs. The resources can be released, in the earliest case, once the object presentation is started (for media type such as still images).

Based on these assumptions, we now discuss the idea behind our approach for determining the retrieval schedule.

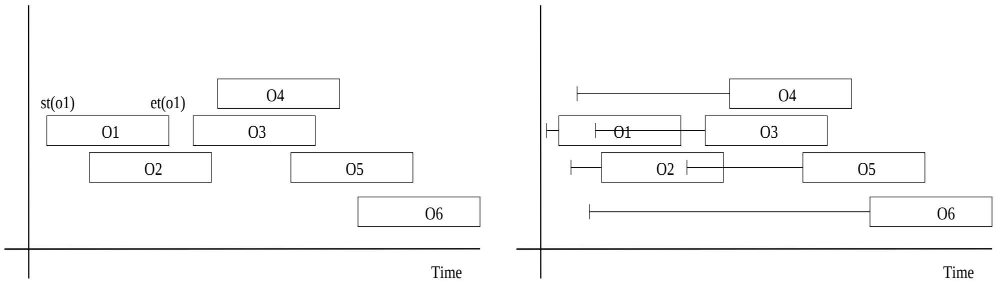
**Single Object Retrieval:** As the simplest case, let us consider the retrieval of single object as shown in Figure 5 (a). The object $O$ has to be presented by the client at time $st(O)$. If the object is atomic, then the retrieval of the object has to be completed before $st(O)$. The client makes a request at time $req(O)$ to the server for the transfer of the object ($req(O)$ must be *before* $st(O)$). Here, $req(O)$ depends on the time required for transferring the object from the server to the client and on the round trip time required for sending the request to the server and receiving a response. Hence, for the case of single atomic object retrieval, $req(O)$ can be defined as:

$$\begin{aligned} &req(O) = st(O) - \{\tfrac{sz(O)}{Th_{max}} + \Delta t\}, \text{ and} \\ &rec(O) = req(O) + \tfrac{\Delta t}{2}, \text{ where,} \end{aligned}$$

$\Delta t$ is the round trip propagation time for sending the request and receiving a response. Based on the above discussion, the system dependent factors such as the throughput and the buffer requirements during a time interval for single atomic object retrieval is simple, and is as follows.

$$\left.\begin{array}{ll} th(O) = Th_{max} & \\ buf(O,t) = 0 & \{t < rec(O)\} \\ buf(O,t) = Th_{max} \times (t - rec(O)) & \{rec(O) \le t \le (st(o)\} \end{array}\right\} \quad \textit{in interval } [req(O), st(O)]$$

In the above equations $t$ varies between $req(O)$ and $st(O)$. When $t$ is equal to $req(O)$, the buffer requirement is 0, whereas when $t$ becomes equal to $st(O)$, the amount of buffer required for the object $O$ rises to $sz(O)$.



(a) Presentation Schedule (b) Initial Retrieval Schedule

Figure 6: Parallel, Multiple Object Retrieval

In the case of $O$ being a stream object, a chunk of frames (shown by the shaded portion in Figure 5(a)) is to be retrieved before the start of the presentation. The size of this chunk depends on the type of the media, the jitter requirement, and the display hardware at the client. The time $req(O)$ at which the client has to make a request to the server for transferring the object follows the same argument as in the case of an atomic object, except that the size of the object $sz(O)$ has to be replaced with the size of the chunk of frames to be retrieved $sz(O_{chunk})$. Hence,

$$req(O) = st(O) - \{\frac{sz(O_{chunk})}{Th_{max}} + \Delta t\}$$

However, the throughput and the buffer requirements are different from that for an atomic object:

$$\left.\begin{array}{ll} th(O) = Th_{max} & \\ buf(O,t) = 0 & \{t < rec(O)\} \\ buf(O,t) = Th_{max} \times (t - rec(O)) & \{rec(O) \leq t \leq (st(o)\} \end{array}\right\} \quad in\ interval\ [req(O), st(O)]$$

$$\left.\begin{array}{l} th(O) = c(O) \\ buf(O,t) = sz(O_{chunk}) \end{array}\right\} \quad in\ interval\ [st(O), et(O)]$$

Here again, $t$ varies between $req(O)$ and $st(O)$.

Figure 5 (b) shows the buffer requirements for a single object (atomic and stream) retrieval. Buffer requirements of an atomic object is basically in the time interval: [req(O), st(O)], whereas the requirements of stream objects are distributed in the time interval: [req(O), et(O)].

**Parallel, Multiple Object Retrieval:** In many cases, the presentations of multiple objects that are to be retrieved over the same network connection can overlap, as shown in figure 6. In this case, the available throughput and the buffer resources have to be shared among the objects to be retrieved. For instance, the stream objects in figure 6(a) are initially assigned the following throughputs:

$$\begin{aligned}
th(O1) &= Th_{max}/2 \\
th(O2) &= Th_{max}/2 \\
th(O3) &= Th_{max}/3 \\
th(O4) &= Th_{max}/3 \\
th(O5) &= Th_{max}/3 \\
th(O6) &= Th_{max}/2
\end{aligned}$$

Each object $O$ is assigned a throughput of $Th_{max}/n$ where $n$ is the maximum number of objects that simultaneously overlap during the presentation time of $O$ (i.e. from $st(O)$ to $et(O)$). The corresponding buffer requirements, $req(O)$, and $rec(O)$, then, can be calculated using these throughput values. However, the throughput values used for the above calculation are only estimates. These (heuristic and initial) estimates are made on the basis of the overlap of the presentation times of the objects. When the values for $req(O)$ are determined, one might find that the object retrieval time overlaps in a different manner from their presentation times. Figure 6(b) shows the overlap of the retrieval schedules of the objects in Figure 6(a). Hence, the summation of the throughput estimates in the retrieval schedules has to be checked to ensure that the maximum offered throughput $Th_{max}$ (by the network service provider) is not exceeded. A similar discussion applies to buffer estimates also. Checking the throughput and buffer estimates can be done for each *time interval.* We can define a time interval $(a, b)$ as the time period between the occurrence of two successive events $a$, $b$ occur. The events may be:

- Request time of an object $O$ ($req(O)$).
- Presentation start time of an objects $O$ ($st(O)$).
- Presentation end time of an object $O$ ($et(O)$).

For each time interval, the constraints that must be obeyed by the schedules of all the objects sharing the same communication path are the following:

- Throughput: $th(o_1) + \ldots + th(o_n) \leq \mathrm{Th}_{max}$
- Buffer: $buf(o_1, t) + \ldots + buf(o_n, t) \leq \mathrm{Buf}_{max}$

**Example 5.1** Consider the two stream objects $o1$ and $o2$ that are scheduled as in figure 7.

The throughput requirement of the system can be calculated as follows:

- **interval (0-1):** No information is being transmitted on the connection line. Hence, the total throughput $\mathrm{Th}_{tot}$ is 0.
- **interval (1-2):** The initial fraction of the stream object $o1$ is being retrieved. Hence, assuming that $b_{init}(o) \leq \mathrm{Buf}_{max}$, the total throughput requirement is

  $$\mathrm{Th}_{tot} = \frac{b_{init}(o1)}{(st(o1) - rec(o1))}$$

  The buffer requirement, on the other hand, is

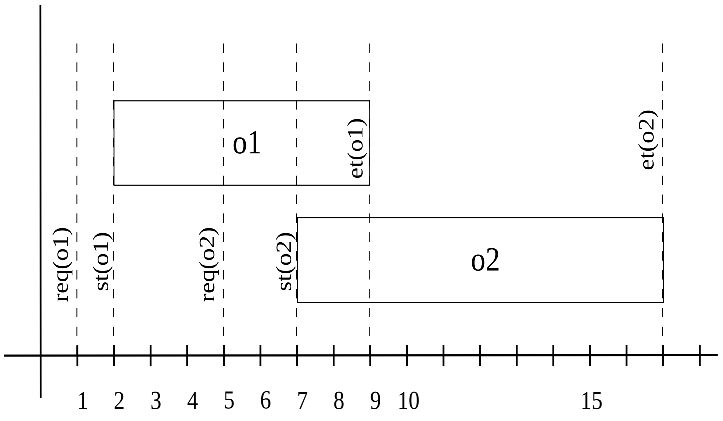


Figure 7: Example 3

Buf$_{tot}(t)$ =$buf(o1,t) = 0$ (when $t < rec(o1)$), and
Buf$_{tot}(t)$ =$buf(o1,t) = (\frac{b_{init}(o1)}{(st(o1)-rec(o1))}) \times (t - rec(o1))$ (when $t \geq rec(o1)$).

- **interval (2-5):** The throughput on the communication path is equal to the consumption rate of $o1$. Hence,

  Th$_{tot}$=c$_s(o1)$, and
  Buf$_{tot}$= $b_{init}(o1)$.

- **interval (5-7):** Both $o1$ and $o2$ use the communication line: $o1$ receives the remaining portion of its stream information, and $o2$ receives its initial fraction.

  Th$_{tot}$= $\frac{b_{init}(o2)}{(st(o2)-rec(o2))}$+c$_s(o1)$

  Note that during this interval, the followings also hold:

  $buf(o1,t)$ =$b_{init}(o1)$,

  $buf(o2,t) = 0$ (when $t < rec(o2)$), and
  $buf(o2,t) = (\frac{b_{init}(o2)}{(st(o2)-rec(o2))}) \times (t - rec(o2))$ (when $t \geq rec(o2)$). .

  Hence, $buf(o1,t)$+$buf(o2,t)$ must be less than or equal to Buf$_{max}$. If this relation does not hold, then the schedule is not feasible. We will later show how to modify non-feasible schedules to make them conform to the constraints imposed by the system. For now we only state what needs to hold during the presentation.

- **interval (7-9):** Both $o1$ and $o2$ use the communication line to receive the remaining portions of their stream information:

  Th$_{tot}$=c$_s(o1)$+c$_s(o2)$

  and

  $b_{init}(o1)$+$b_{init}(o2)$ ≤Buf$_{max}$.

- **interval (9-17):** Only $o2$ is active and receiving the remaining portion of the stream information:

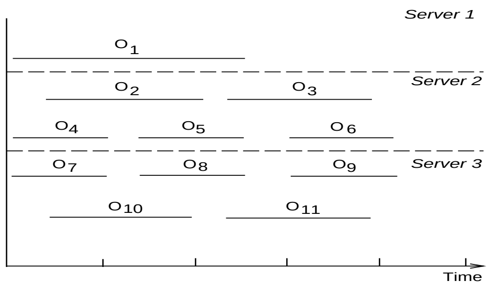


Figure 8: Parallel, Multiple Object Retrieval From Multiple Servers

$$\mathrm{Th}_{tot} = \mathrm{c}_s(o2)$$

and

$$b_{init}(o2) \leq \mathrm{Buf}_{max}. \qquad \square$$

**Parallel, Multiple Object Retrieval From Multiple Servers:** Figure 8 shows how objects composing a multimedia document presentation have to be retrieved from different servers. Here, separate network connections will be used for retrieving the objects from different servers. The throughput constraints for multiple object retrieval has to be satisfied for each network connection separately. However, the buffer constraint is the same because the entire retrieval is handled by the client.

**Interaction Between Throughput and Buffer Constraints:** Both throughput and buffer are resources provided by the system: the first is provided by the network service provider and the latter by the client system. When a certain throughput is offered by the network service provider, the required buffer resources become an estimate based on this offered throughput. The estimated buffer requirement, then, has to be checked to ensure that it can be provided by the client system. In order to achieve this, we go through a process of validating the retrieval schedule that was generated based on the above discussion.

## 6 Retrieval Schedule Validation

As discussed above, the generated retrieval schedule has to be checked to see whether it satisfies the system constraints such as throughput and buffer. This validity is checked for every time interval in the generated retrieval schedule. It should be noted here that the retrieval schedule generation and validation process is done for every time segment in the entire multimedia presentation, as discussed in Section 3.1. In case, modifications to the presentation or the retrieval schedules are necessary, it is easier to start from the end of the segment and work backwards. Working backwards from the segment end time helps in redoing already validated schedules. The following algorithm shows how the presentation and retrieval schedules for a given segment $s$ is validated.

**Input:** A segment $s$, the tentative presentation and retrieval schedules for $s$, the throughput and buffer availability constraints that the system must obey.

**Output:** Validity or otherwise of the presentation and retrieval schedules for the segment $s$.

The algorithm starts from the last interval of the segment, and it proceeds towards the earlier intervals. This enables the system to first fix the *starting times* and then the *retrieval times*. As a result, the system tries to change the presentation schedule only if it can not change the retrieval schedule. We chose this order because, as mentioned earlier, sticking to the presentation schedule is more desirable than sticking to the retrieval schedule.

**Algorithm:**

```
1. Sort the events (time instances at which changes in requirements occur), and identify the
   number of intervals ($num_{int}$).
2. Set the borders of the intervals as **unmarked**. (When a border is **marked**, the event that
   corresponds to the border can not be changed).
3. $Satisfied = True$;
4. $ThisInterval = last\,interval$;
5. **while** $Satisfied$ and ($ThisInterval \geq FirtInterval$) **do**
   (a) Check if the interval $ThisInterval$ is *valid*
   (b) **while** $ThisInterval$ is not *valid* and **do**
       i. Create a feedback (section 7). Note that the values that are already **marked** must
          be kept constant in the feedback.
       ii. Send the feedback to the *schedule modifier*.
       iii. **while** *schedule modifier* returns no schedule and ($ThisInterval < last\,interval$) **do**
            A. $ThisInterval = ThisInterval + 1$;
            B. Set the borders of the interval $ThisInterval$ as **notmarked**
            C. Create a feedback (section 7). Note that the values that are already **marked**
               must be kept constant in the feedback.
            D. Send the feedback to the *schedule modifier*.
       iv. Check if the interval $ThisInterval$ is *valid*
   (c) **if** interval $ThisInterval$ is *valid* **then**
       i. Set the borders of the interval as **marked**
       ii. $ThisInterval = ThisInterval - 1$
   (d) **else**
       i. $Satisfied = False$
6. **if** $Satisfied$ **then**
   (a) return the schedules
```

7. **else**
    (a) return empty schedule

**Example 6.1** Consider the two stream objects $o1$ and $o2$ that are scheduled as in figure 7.

Assume that the segment in example 5.1 is fed into the schedule validator. Let us also assume that the throughput and the buffer constraints of the system is also as specified in Section 5.

The algorithm will start from the last interval, i.e. (9-17). It will check the throughput and the buffer constraints as specified in example 5.1. Let us assume that this interval does not violate any cosntraints. Then, the algorithm marks the variables $et(o2)$ and $et(o1)$, that is it declares that the values of these variables should not change with subsequent operations.

The algorithm, then, will try to validate interval (7-9). Let us assume that the throughput required for this operation is more than the available throughput. One solution to this problem is to reduce the stream throughput of the object $o2$ and to move its request time to an earlier point in time (the details of this operation is described in gerater detail in section 7). Let us assume that the system decides to apply this solution, and it moves the req(o2) from 5 to 4. Hence, as a result, the interval (5-7) changes to interval (4-7), and similarly the interval (2-5) changes to interval (2-4). At the end of this step the variable $st(o2)$, which denotes the start of the interval (7-9), is marked by the system.

In the subsequent iterations, the intervals (4-7), (2-4), (1-2), and (0-1) are going to be validated in a similar fashion. □

In the next section we show how the feedback is generated, and how it is used by the system.

## 7 Feedback Generation and Schedule Modification

As discussed above, the schedule validator checks for the satisfiability of the two system constraints: throughput and buffer. If the throughput or buffer required by the schedules exceed the upper bounds, then the schedule validator declares the generated retrieval schedule invalid. In such cases, the schedules (retrieval or presentation or both) have to be modified such that the system constraints are satisfied. If schedule modifications are not possible, then the quality of the presentation can be reduced.

The schedule validator generates feedback for modifying the schedules, depending on how the system constraints were violated. In this section, we discuss how the schedule validator generates appropriate feedbacks.

## 7.1 Throughput Violation

In Section 5, we showed that the total throughput needed for multimedia objects retrieval in a time interval can be expressed as

$$\mathrm{Th}_{tot} = \underbrace{\frac{c_1}{(st(o_1) - rec(o_1))} + \ldots + \frac{c_n}{(st(o_n) - rec(o_n))}}_{non-stream} + \underbrace{d_1 + \ldots + d_m}_{stream},$$

where $c_1$ through $c_n$ are constants denoting the sizes of the non-stream information (atomic objects, or the initial buffer requirements of the stream objects), and $d_1$ through $d_m$ are constants denoting the throughput requirements of the objects with constant consumption rate (stream objects).

If the total required throughput ($\mathrm{Th}_{tot}$) exceeds the available throughput($\mathrm{Th}_{max}$), then we need to reduce the throughput requirement by

$$\delta_{thru} = \mathrm{Th}_{tot} - \mathrm{Th}_{max} = \underbrace{{\delta_{thru}}^1 + \ldots + {\delta_{thru}}^n}_{non-stream} + \underbrace{{\theta_{thru}}^1 + \ldots + {\theta_{thru}}^m}_{stream}.$$

Here $\delta$s come from the atomic components, and $\theta$s come from stream components of the above equation. The amount of reduction required ($\delta_{thru}$) is distributed on $\delta$s and $\theta$s using their priorities.

### 7.1.1 Handling Non-Stream Retrievals

For reducing the throughput utilized by the non-stream information (i.e., the throughput of the form $\frac{c}{(st(o)-rec(o))}$, we need either to increase the time of retrieval $(st(o) - rec(o))$ or decrease the size of the object $c$. In other words, we have the following options:

- Modify the retrieval schedule by changing $rec(o)$.
- Modify the presentation schedule by changing $st(o)$.
- Modify the quality of the presentation by reducing the size $c$.

Changing the retrieval schedule or the presentation schedule involves modification of the value of $rec(o)$ (time at which the object has to arrive in the client side) and $st(o)$ (the presentation start time). The modification of retrieval schedule is the most desirable option since it does not involve any change in the presentation schedule or the quality of presentation.

The desired reduction in the throughput for non-stream information of size $c_j$, then, can be expressed as:

$$\frac{c_j}{(st(o_j)-rec(o_j))} - \frac{c_j}{(st'(o_j)-rec'(o_j))} \geq {\delta_{thru}}^j \ ,$$

where $st$ and $rec$ denote the current values of the presentation start time and the receive time, $st'$ and $rec'$ denote the corresponding new values, and ${\delta_{thru}}^j$ is a positive real number. The above equation can also be rewritten as

$$st'(o_j) - rec'(o_j) \geq \frac{c_j}{-\delta_{thru}{}^j + \frac{c_j}{(st(o_j) - rec(o_j))}}.$$

**Modifying the retrieval schedule:** The retrieval schedule $rec(o)$ can be modified by keeping the presentation start time unchanged (i.e., $st'(o) = st(o)$). Hence, the new value for $rec'(o)$ is

$$rec'(o_j) \leq st(o) - \frac{c_j}{-\delta_{thru}{}^j + \frac{c_j}{(st(o_j) - rec(o_j))}}.$$

If this change in the retrieval schedule is not acceptable (for example, if it leads to a longer wait time before the presentation can be started), then the presentation schedule $st(o)$ can be modified as follows.

**Modifying the presentation schedule:** To modify the presentation start time of an object, we need to keep the retrieval schedule $rec(o)$ unchanged (i.e., $rec'(o) = rec(o)$). When we substitute $rec'(o_j) = rec(o_j)$ in the equation, we get

$$st'(o_j) \geq rec(o_j) + \frac{c_j}{-\delta_{thru}{}^j + \frac{c_j}{(st(o_j) - rec(o_j))}}.$$

The range for the new presentation start time of the object $o_j$ can be fed into a presentation schedule modifier (discussed in section 7.2) in order to find another feasible value for $st(o_j)$. The modifier generates a new presentation schedule with the new presentation start time for object $o_j$, such that the values of the already validated presentation schedule variables are kept unchanged. Section 7.2 discusses this issue in more detail.

In both the above cases (modifying the presentation schedule and modifying the retrieval schedule), there will be a change in the intervals of the segment. Since the validation process is carried out backwards starting from the segment end time, and since, the values of the already validated intervals are kept unchanged, there is no need for backtracking in the validation process. The interval changes will affect only the non-validated parts of the segment.

**Modifying the quality of presentation:** If the above modifications of retrieval and presentation schedules are not feasible, then the quality of the presentation must be modified as a last resort. Reduction in the quality of an object implies a reduction in the size of the information to be retrieved. A smaller sized object than be retrieved with a smaller throughput. As described in section 2.2, this operation can be performed by using a look-up table to see what the smallest feasible reduction in the object quality is.

### 7.1.2 Handling Stream Retrievals

The stream retrieval part deals with the throughput requirements of the stream objects *after* the object presentation has started. For reducing the throughput required for stream object retrieval, we have the following options:

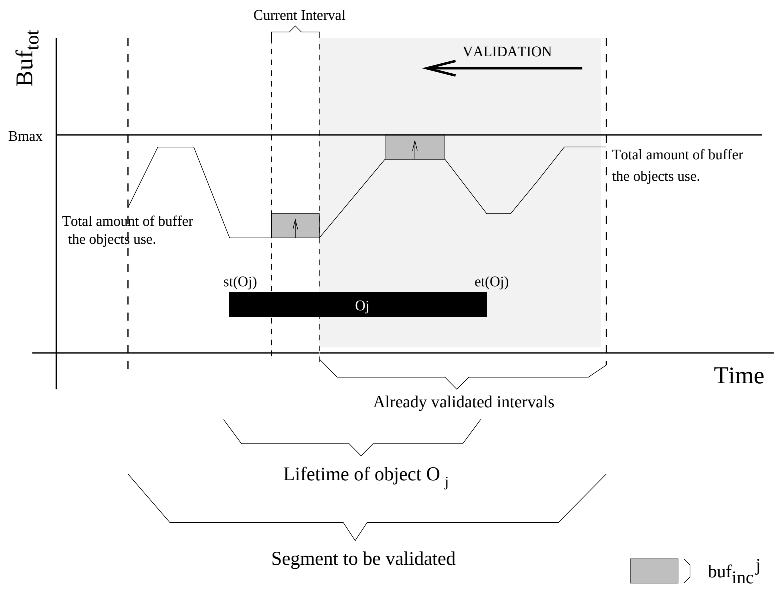


Figure 9: The increase in the buffer size is limited with the minimum available buffer

- Buffering a larger initial chunk of the object *before* the presentation.
- Reducing the quality of the presentation by reducing the size of the objects to be retrieved.

**Increased Initial Buffering:** Let us assume that for the stream object $o_j$, the required reduction in throughput is $\theta^j_{thru}$. This reduction is for the time interval $< st(o_j), et(o_j) >$, corresponding to the start and end times of presentation of the objects $o_j$. Hence, the required increase in the size of the initial buffer can be calculated as

$$buf_{inc}{}^j = duration_j \times \theta_{thru}{}^j, \text{ where } duration_j = et(o_j) - st(o_j).$$

Increasing the initial buffer size may, however, may cause buffer violations at the already validated intervals. Hence, an increase can be allowed only if there is enough available buffer space to accomodate the suggested increase: $buf_{inc}{}^j$ must be limited by the minimum buffer size available during the already validated portion of $o_j$'s presentation (figure 9).

Note that an increase in the size of the initial buffer requires that the retrieval schedule for the initial chunk is suitably modified. The change in $rec(o)$ can be calculated as follows:

$$\delta_{rec(o_j)} = buf_{inc}{}^j \times th(o_j),$$

where $th(o_j)$ denotes the throughput assigned to object $o_j$ for the initial chunk retrieval.

### 7.1.3 Buffer Violation

The other system resource that may be inadequate, is the buffer. In section 5, we showed that the total buffer requirement at a time instant $t$ is

$$\text{Buf}_{tot} = \underbrace{\frac{c_1}{(st(o_1) - rec(o_1))} \times (t - rec(o_1)) + \ldots + \frac{c_n}{(st(o_n) - rec(o_n))} \times (t - rec(o_n))}_{non-stream} + \underbrace{e_1 + \ldots + e_m}_{stream},$$

where $c_1$ through $c_n$ are constants denoting the sizes of the non-stream information being retrieved, and $e_1$ through $e_m$ are constants denoting the buffer requirements of the stream objects.

If we consider an interval of the form $< t_{start}, t_{end} >$, the total buffer requirement at the end of the time interval is

$$\text{Buf}_{tot} = \underbrace{\frac{c_1}{(st(o_1) - rec(o_1))} \times (t_{end} - rec(o_1)) + \ldots + \frac{c_n}{(st(o_n) - rec(o_n))} \times (t_{end} - rec(o_n))}_{non-stream} + \underbrace{e_1 + \ldots + e_m}_{stream}.$$

Note that the maximum buffer requirement within an interval occurs at the end of the interval. Hence, the above equation also gives the maximum buffer requirement within the interval $< t_{start}, t_{end} >$. If Buf$_{tot}$ calculated as above exceeds the available buffer space (Buf$_{max}$) of the system, then we must reduce the buffer usage by

$$\delta_{buf} = \text{Buf}_{tot} - \text{Buf}_{max}.$$

Note that $\delta_{buf}$ can also be written as

$$\delta_{buf} = {\delta_{buf}}^1 + \ldots + {\delta_{buf}}^n + {\theta_{buf}}^1 + \ldots + {\theta_{buf}}^m.$$

As in the discussion for throughput violation, the above equation comprises of two components:

- (non-stream) Buffer requirements for atomic objects and initial chunk retrieval of stream objects.
- (stream) Buffer requirements during the presentation of stream objects.

### 7.1.4 Handling Non-stream Retrievals

To reduce the buffer requirements during the retrieval of the atomic objects and the initial chunk retrieval of stream objects (i.e., the component $\frac{c}{(st(o)-rec(o))} \times (t - rec(o))$, we need to either increase the retrieval time $(st(o) - rec(o))$ or reduce the size of the object $c$. This in effect results in one or more of the following:

- Modification of the retrieval schedule $(rec(o))$.

- Modification of the presentation schedule ($st(o)$).
- Modification of the presentation quality ($c$).

Changing the retrieval or presentation schedule involves modification of the value of $rec(o)$ and $st(o)$, i.e., the time at which the object has to arrive in the client side and the presentation start time. The modification of retrieval schedule is most desirable since it does not involve any change in the presentation schedule or the quality of presentation. The desired reduction in the buffer requirement for an object $o_j$ can be expressed as

$$\frac{c_j}{(st(o_j)-rec(o_j))} \times (t_{end} - rec(o_j)) - \frac{c_j}{(st'(o_j)-rec'(o_j))} \times (t_{end} - rec'(o_j)) \geq \delta_{buf}{}^j$$

where $st$ and $rec$ denote the current values of these variables, $st'$ and $rec'$ denote the values that we are searching for, and $\delta_{buf}{}^j$ is a positive real number. The above equation can be rewritten as

$$\frac{c_j}{(st'(o_j)-rec'(o_j))} \times (t_{end} - rec'(o_j)) \leq -\delta_{buf}{}^j + \frac{c_j}{(st(o_j)-rec(o_j))} \times (t_{end} - rec(o_j)).$$

**Modifying the retrieval schedule:** This involves a change in the value of $rec(o)$. To change $rec(o)$, we should keep the presentation start time unchanged (i.e., $st'(o) = st(o)$). Substituting $st'(o_j) = st(o_j)$ in the above equation, we get

$$\frac{c_j}{(st(o_j)-rec'(o_j))} \times (t_{end} - rec'(o_j)) \leq \underbrace{-\delta_{buf}{}^j + \frac{c_j}{(st(o_j) - rec(o_j))} \times (t_{end} - rec(o_j))}_{\gamma}$$

$$(\gamma - c_j) \times rec'(o_j) \leq (\gamma \times st(o_j)) - (c_j \times t_{end})$$

$$\begin{cases} rec'(o_j) \leq \frac{(\gamma \times st(o_j))-(c_j \times t_{end})}{(\gamma - c_j)} & for(\gamma - c_j) > 0 \\ rec'(o_j) \geq \frac{(\gamma \times st(o_j))-(c_j \times t_{end})}{(\gamma - c_j)} & for(\gamma - c_j) < 0 \end{cases}$$

$$\begin{cases} rec'(o_j) - rec(o_j) \leq \frac{(\gamma \times st(o_j))-(c_j \times t_{end})}{(\gamma - c_j)} - rec(o_j) & for(\gamma - c_j) > 0 \\ rec'(o_j) - rec(o_j) \geq \frac{(\gamma \times st(o_j))-(c_j \times t_{end})}{(\gamma - c_j)} - rec(o_j) & for(\gamma - c_j) < 0 \end{cases}$$

The above set of equations gives a range $(rec(o_j - rec'(o_j))$ in which the retrieval schedule can be suitably modified.

**Modifying the presentation schedule:** This involves a change in the value of $st(o)$. To do this, we should keep the object retrieval time unchanged (i.e., $rec(o_j) = rec'(o_j)$). Substituting $rec(o_j) = rec'(o_j)$ in the above equation, we get

$$\frac{c_j}{(st('o_j)-rec(o_j))} \times (t_{end} - rec(o_j)) \leq \underbrace{-\delta_{buf}{}^j - \frac{c_j}{(st(o_j) - rec(o_j))} \times (t_{end} - rec(o_j))}_{\gamma}$$

$$\begin{cases} st'(o_j) \geq \frac{c_i \times (t_{end} - rec(o_j))}{\gamma} + rec(o_j) & for(\gamma > 0) \\ st'(o_j) \leq \frac{c_i \times (t_{end} - rec(o_j))}{\gamma} + rec(o_j) & for(\gamma < 0) \end{cases}$$

$$\begin{cases} st'(o_j) - st(o_j) \geq \frac{c_i \times (t_{end} - rec(o_j))}{\gamma} + rec(o_j) & for(\gamma > 0) - st(o_j) \\ st'(o_j) - st(o_j) \leq \frac{c_i \times (t_{end} - rec(o_j))}{\gamma} + rec(o_j) & for(\gamma < 0) - st(o_j) \end{cases}$$

The above set of equations gives us a range $(st'(o_j) - st(o_j))$ in which the presentation start time can be suitably modified. This range has to be given to the presentation schedule modifier (discussed in section 7.2) to generate a new schedule with the presentation start time for the object $o_j$ in the suggested range. While generating this new presentation schedule, the schedule modifier keeps the values of the other presentation variables (start and end times of presentations of other objects) constant.

**Modifying the presentation quality:** This involves reducing the size of the objects to be retrieved. The procedure is similar to the one discussed for handling throughput violation in section 7.1.

### 7.1.5 Handling Stream Retrievals

The reduction in the stream components of the buffer usage, can only be made by reducing the size of the objects: A reduction in the object size would reduce the amount of buffers needed for its storage. Reduction in presentation quality can be made as a last resort. The procedure is the same as the one discussed for handling throughput violation in section 7.1.

## 7.2 Presentation Schedule Modifier

The schedule modifier module takes as input the range of values for the presentation start times of the objects, as suggested by the feedback generator. The schedule modifier then generates a new schedule in which only the start times of the objects suggested by the feedback generator are modified and other presentation variables (i.e., the start and end presentation times of other objects) are left unchanged.

**Definition 7.1 (Presentation Schedule Modifier)** Let $G$ be a weighted, directed constraint graph which represents the temporal specifications of the multimedia document. Let $v_i$ denote the temporal variables of the document, and let $\phi$ denote a schedule for the document, i.e. a mapping from the variables into reals.

Let $\delta$ be a mapping from the vertex variables into $(R \times R) \bigcup \perp$, where $\delta(v_i)$ specifies the range of the change required in the value of $v_i$. If $\delta(v_i) = \perp$, then the value of $v_i$ can be changed freely. Note that $\delta(st_p)$ must always be $< 0, 0 >$.

$\phi$

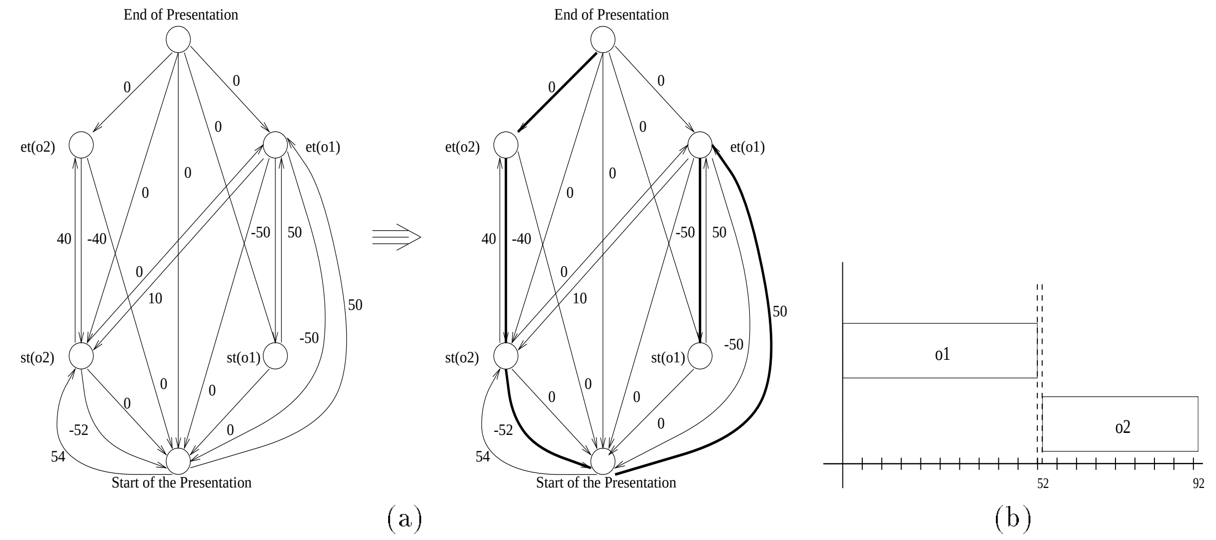


Figure 10: The modified constraint graph, and the corresponding solution

The *presentation schedule modifier* takes $G$, $\phi$ (the initial solution) and $\delta$ (the feedback) as inputs, and it returns a new solution $\phi_{new}$. If, however, there is no solution satisfying the feedback, then the algorithm returns false. $\diamond$

In order to see how the schedule modifier works, observe that if $\delta(v_i) =< a, b >$, then

$$\phi(v_i) + a \leq \phi_{new}(v_i), \text{ and}$$
$$\phi_{new}(v_i) \leq \phi(v_i) + b.$$

Since the values of the temporal variables are with respect to the start of the presentation, i.e. $\phi_{new}(v_i) = v_i - st_p$, we can rewrite the above inequalities as

$$st_p - v_i \leq -(\phi(v_i) + a), \text{ and}$$
$$v_i - st_p \leq \phi(v_i) + b.$$

Hence, for each variable $v_i$ such that $\delta(v_i) \neq \perp$, we need to introduce two new constraints to the document. This insertions can be represented as the additions of two new edges to the graph $G$.

**Example 7.1** Let us consider the multimedia document described in example 4.1. Let us assume that the presentation schedule found in that example does not lead into a suitable retrieval schedule, and the validator asks the schedule modifier to postpone the start of the object $o_2$ by 2 to 4 seconds while keeping the end of the object $o_1$ as it is (we show in section 7) how the feedbacks are generated).

The inputs to the schedule modifier, that is $\phi$ and $\delta$, are as follows:

$$\phi(st_p) = 0 \qquad\qquad \delta(st_p) =< 0, 0 >$$

| | |
|---|---|
| $\phi(st(o_1)) = 0$ | $\delta(st(o_1)) = \perp$ |
| $\phi(et(o_1)) = 50$ | $\delta(et(o_1)) =< 0, 0 >$ |
| $\phi(st(o_2)) = 50$ | $\delta(st(o_2)) =< 2, 4 >$ |
| $\phi(et(o_2)) = 90$ | $\delta(et(o_2)) = \perp$ |
| $\phi(et_p) = 90$ | $\delta(et_p) = \perp$ |

Hence, the corresponding new constraints are
$st_p - st(o_1) \leq -50$, and
$st(o_1) - st_p \leq 50$.
$st_p - st(o_2) \leq -52$, and
$st(o_2) - st_p \leq 54$.
The corresponding new graph can be seen in figure 10(a). At the end, the vertices of the graph and the $\phi_{new}$ have the following values:

| | |
|---|---|
| $et_p = 0$ | $\phi_{new}(st_p) = 92$ |
| $st_p = -92$ | $\phi_{new}(et_p) = 0$ |
| $et(o_2) = 0$ | $\phi_{new}(et(o_2) = 92$ |
| $st(o_2) = -40$ | $\phi_{new}(st(o_2) = 52$ |
| $et(o_1) = -42$ | $\phi_{new}(et(o_1) = 50$ |
| $st(o_1) = -92$ | $\phi_{new}(st(o_1) = 0$ |

Figure 10(b) shows the corresponding schedule. □

In [5], we showed how the additions of new constraints can be handled efficiently. Hence, in this paper we are not going into the details of this process.

## 8 Conclusion

A distributed multimedia document involves retrieval of objects from server(s) and their presentation at the client systems. The presentation of the multimedia objects have to be carried out in accordance with the specified temporal relationships among the objects composing the presentation. Flexibility in the specification of the temporal relationships helps in deriving a set of possible presentation schedules, with each schedule representing one possible *view* of the document. The retrieval of multimedia objects from the server(s) is influenced by factors such as:

- Presentation schedule of the multimedia objects.
- Maximum throughput offered by the network service provider.
- Maximum buffer resources available on the client system.

In the previous approaches [16, 18, 26, 27], the multimedia presentation schedule is fixed before the generation of the retrieval schedule. Based on an assumed network throughput availability, the retrieval schedules are derived to generate a retrieval schedule. The generated schedules do not handle variations in the offered system resources such as network throughput and available buffer resources.

In this paper, we have developed techniques for deriving flexible object presentation and retrieval schedules for a distributed multimedia document presentation. The main advantage in the proposed methodology is that, instead of choosing a presentation schedule and trying to find a matching retrieval schedule, the proposed algorithms modify both the presentation and retrieval schedules in such a way that system resources (network throughput and buffer resources in the client system) are used in a very efficient manner.

## References

[1] J.F. Allen (1984) *Towards a General Theory of Time and Action*, Artificial Intelligence, 23, pps 123–154.

[2] M.C. Buchanan and P.T. Zellweger (1993) *Automatic Temporal Layout Mechanisms* ACM Multimedia 93, pp. 341-350.

[3] K. Selçuk Candan, V.S. Subrahmanian, and P. Venkat Rangan (1995) *Collaborative Multimedia Systems: Synthesis of Media Objects*, Submitted for publication, Nov. 1995. Also available as Technical Report: CS-TR-3595, UMIACS-TR-96-8, University of Maryland, College Park, Computer Science Technical Report Series, January 1996.

[4] K. Selçuk Candan, V.S. Subrahmanian, and P. Venkat Rangan (1996) *Towards a Theory of Collaborative Multimedia*, IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan.

[5] K. Selçuk Candan, B. Prabhakaran, and V.S. Subrahmanian (1996) *Collaborative Multimedia Documents: Authoring and Presentation.* Submitted for publication. Also available as Technical Report: CS-TR-3596, UMIACS-TR-96-9, University of Maryland, College Park, Computer Science Technical Report Series, January 1996.

[6] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, McGraw Hill Publishers.

[7] D. Ferrari (1990) *Client Requirements For Real-Time Communication Services*, IEEE Communication Magazine, Vol. 28, No. 11, Nov. '90, pp. 65-72.

[8] D. Ferrari, J. Ramaekers and G. Ventre (1992) *Client-Network Interactions in Quality of Service Communication Environments*, Proc. of High Performance Networking, 4th IFIP Conf. on High Performance Networking, Liege, Belgium, 14-18, December '92.

[9] H. Gajewska (1994) *Argo: A System for Distributed Collaboration*, ACM Multimedia 94, Pages 433-440.

[10] A. Ginsberg and S. Ahuja (1995) *Automating envisionment of virtual meeting room histories*, ACM Multimedia 95, Pages 65-76.

[11] F. Gong (1994) *Multipoint Audio and Video Control for Packet-Based Multimedia Conferencing*, ACM Multimedia 94, Pages 425-432.

[12] F. Hillier and G. Lieberman. (1974) *Operations Research*, Holden-Day.

[13] T. Imai, K. Yamaguchi, T. Muranaga (1994) *Hypermedia Conversation Recording to Preserve Informal Artifacts in Realtime*, ACM Multimedia 94, Pages 417-424.

[14] H. Korth and A. Silberschatz. (1986) *Database System Concepts*, McGraw Hill.

[15] M.Y. Kim and J. Song. (1995) *Multimedia Documents with Elastic Time*, ACM Multimedia 95.

[16] L. Li, A. Karmouch and N.D. Georganas (1994) *Multimedia Teleorchestra With Independent Sources: Part 1 and Part 2*, ACM/Springer-Verlag Journal of Multimedia Systems, vol. 1, no. 4, February 1994, pp.143-165.

[17] T.D.C. Little and A. Ghafoor (1990) *Synchronization and Storage Models for Multimedia Objects*, IEEE J. on Selected Areas of Communications, vol. 8, no. 3, April 1990, pp. 413-427.

[18] T.D.C. Little and A. Ghafoor (1991) *Multimedia Synchronization Protocols for Broadband Integrated services*, IEEE J. on Selected Areas of Communications, vol. 9, no. 9, Dec. 1991, pp. 1368-1382.

[19] T.D.C. Little and A. Ghafoor (1993) *Interval-Based Conceptual Models for Time-Dependent Multimedia Data* IEEE Transactions on Knowledge and Data Engineering, vol. 5, no. 4, Aug 1993, pp. 551-563.

[20] N.R. Manohar and A. Prakash (1995) *Dealing With Synchronization and Timing Variability in the Playback of Interactive Session Recordings*, ACM Multimedia 95, Pages 45-56.

[21] N. Pahuja, B.N. Jain and G.M. Shroff (1996) *Multimedia Information Objects: A Conceptual Model for Representing Synchronization*, Proceedings of International Conference on Computer Networks, Networks'96, Bombay, India, January 1996.

[22] M.J. Perez-Luque and T.D.C. Little (1995) *A Temporal Reference Framework for Multimedia Synchronziation*, IEEE Workshop on Multimedia Synchronization, May 1995.

[23] B. Prabhakaran and S.V. Raghavan (1994) *Synchronization Models For Multimedia Presentation With User Participation*, ACM/Springer-Verlag Journal of Multimedia Systems, vol.2, no. 2, August 1994, pp. 53-62. Also in the Proceedings of the First ACM Conference on MultiMedia Systems, Anaheim, California, August 1993, pp.157-166.

[24] B. Prabhakaran (1996) *Multimedia Synchronization*, Chapter in the book *Multimedia Systems and Techniques* published by Kluwer Academic Publishers, 1996, pp.177-214.

[25] N.U.Qazi, M. Woo and A. Ghafoor (1993) *A Synchronization and Communication Model for Distributed Multimedia Objects*, Proceedings of the First ACM Conference on MultiMedia Systems, Anaheim, California, August 1993, pp.147-155.

[26] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi (1994) *Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation*, Special issue on Multimedia Synchronization, IEEE Journal on Selected Areas in Communication, January 1995.

[27] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi (1994) *Quality of Service Considerations For Distributed, Orchestrated Multimedia Presentation*, Proceedings of High Performance Networking 94 (HPN'94), Paris, France, July 1994, pp. 217-238. Also available as Technical Report: CS-TR-3167, UMIACS-TR-93-113, University of Maryland, College Park, Computer Science Technical Report Series, October 1993.

[28] S.V. Raghavan, B. Prabhakaran and Satish K. Tripathi (1995) *Handling QoS Negotiations in Distributed Orchestrated Presentation*, to be published in Journal of High Speed Networking.

[29] S. Rajan, P.V. Rangan, and H.M. Vin (1995) *A Formal Basis for Structured Multimedia Collaborations*, IEEE Intl. Conf. on Multimedia Computing and Systems, 1995.

[30] R. Steinmetz (1990) *Synchronization Properties in Multimedia Systems*, IEEE J. on Selected Areas of Communication, vol. 8, no. 3, April 1990, pp. 401-412.

[31] P.D. Stotts and R. Furuta (1990) *Temporal Hyperprogramming*, Journal of Visual Languages and Computing, Sept. 1990, pp. 237-253.

[32] H. Thimm and W. Klas (1996) *$\delta$-Sets for Optimal Reactive Adaptive Playout Management in Distributed Multimedia Database Systems*, 12'th International Conference on Data Engineering, Feb. 1996, pp. 584-592.

[33] P. van Beek (1990) *Exact and Approximate Reasoning about Qualitative Temporal Relations*, PhD. Thesis, University of Waterloo, Canada, 1990.

[34] M. Vilain and H. Kautz (1986) *Constraint Propogation Algorithms for Temporal Reasoning*, Proceedings of AAAI-86, Artificial Intelligence, Aug. 1986, pp. 377-382.

[35] T.M. Wittenburg and T.D.C. Little (1994) *An Adaptive Document Management System for Shared Multimedia Data*, IEEE Intl. Conf. on Multimedia Computing and Systems, 1994, Pages 245-254.

[36] K.H. Wolf, K. Froitzheim and P. Schulthess (1995) *Multimedia Application Sharing in a Heterogeneous Environment*, ACM Multimedia 95, Pages 57-64.

[37] S. Wray, T. Glauert, and A. Hopper (1994) *The Medusa Applications Environment*, IEEE Intl. Conf. on Multimedia Computing and Systems, 1994, Pages 265-274.