CHIMP: A Framework for Supporting Distributed Multimedia Document Authoring and Presentation

K. Selçuk Candan, B. Prabhakaran, and V.S. Subrahmanian
Department of Computer Science,
Institute for Advanced Computer Studies
Institute for Systems Research,
University of Maryland,
College Park, Maryland 20742.
Email: {vs,candan,prabha}@cs.umd.edu.

ABSTRACT

A multimedia document consists of different media objects that are to be sequenced and presented according to temporal and spatial specifications. Collaborative authoring helps in simultaneous editing and viewing of a multimedia document by multiple authors. However, it may cause the objects composing a multimedia document to be distributed over a computer network. In this paper, we propose a framework for distributed multimedia document authoring and presentation. The salient features of this framework are: flexible temporal specification based on difference constraints, system and user defined access filters, local editing, format conversions of media objects, and flexible object retrieval schedules for handling variations in system parameters such as network throughput and buffer resources. We propose shortest-path based algorithms for solving difference constraints. We show how the proposed algorithms can handle local editing and access filtering of multimedia documents. We also describe how the difference constraints based temporal specifications can help in deriving a flexible object retrieval schedule.

KEYWORDS: Multimedia authoring, multimedia presentation, multimedia object retrieval, presentation schedules, temporal constraints, buffer and throughput requirements.

This research was supported by the Army Research Office under grants DAAH-04-95-10174 and DAAH-04-96-10297,, 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), NSF Young Investigator award IRI-93-57756, and the Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002 Federated Laboratory ATIRP Consortium.

ACM Multimedia 96, Boston MA USA
© 1996 ACM 0-89791-871-1/96/11 ..$3.50

INTRODUCTION

A multimedia document consists of different types of media objects that are to be presented at different instants of time and for different durations. The media objects may be stored in computer systems connected by a network, thereby rendering the document distributed. Authors may wish to collaborate in a distributed manner to edit a multimedia document. They may have different views of the multimedia document depending on their interests and the access rights as allowed by the system. The authors can modify portions of the document on their local system. Such local editing can be realized later on the system document. The systems used for multimedia authoring can have different capabilities for object presentation. For example, a system may not have the tools needed for presenting image or text objects of a particular format. In such a case, these objects might have to be converted from their original formats to suitable formats that can be handled by the authoring system. This format conversion can either be done by the system that stores the object or by a set of intermediate nodes. If the conversion needs to be done by a set of intermediate nodes, the objects have to be *logically routed* via the intermediate nodes. Hence, an authoring system needs to identify a retrieval schedule that describes the time instants at which the objects have to be retrieved as well as the logical paths to be followed by the objects.

In this paper, we consider a distributed multimedia document authoring system with the above mentioned features. The main issues that we focus on in such distributed document authoring systems are the following:

- Temporal specification of the objects composing a multimedia document. Such specifications describe the time instances and durations of presentations of the objects.
- System and user defined filters for accessing the multimedia document.
- Local editing of the document.
- Conversion of objects to appropriate formats suitable for local presentation.
- Retrieving the objects over the computer network for presenting the document.

Our Approach: We propose a difference constraints based temporal specification for multimedia documents [6]. This approach allows us to specify ranges for the temporal values. Such a range specification implies that the values for the start time and duration of an object's delivery can be determined independently for each presentation. In other words, we can generate a flexible schedule for retrieving objects over the computer network. This flexible retrieval schedule can handle variations in system parameters such as available network throughput and buffer resources.

We develop algorithms based on the shortest path approach for solving the temporal specifications. We also propose a methodology for generating flexible retrieval schedules based on temporal specifications. The retrieval schedule may also include provisions to convert objects from one format to another, prior to delivering the object to the client in question. This may be required in situations where the client is unable to view the document in its original form.

Organization of the Paper: In the next section, we discuss the architecture of our multimedia document authoring system. We outline our approach to the above mentioned issues. We also describe the salient features of this system. In the section titled "Temporal specification and presentation schedule generation", we propose a difference constraints based approach for temporal specification of the multimedia document. In the section titled "Multimedia Document Presentation", we describe the steps required for a successful multimedia presentation. In the section titled "Related Work" we compare the features of our framework with the proposed methodologies in the literature. The major symbols we use in this paper are explained in figure 1.

Symbol	Meaning
Th_{max}	Maximum throughput available at a communication line.
Th_{tot}	The total throughput required by the objects sharing a communication line.
$th(\text{o})$	The amount of throughput used by object o.
Buf_{max}	Maximum buffer available for the objects sharing a communication line.
$\text{Buf}_{tot}(t)$	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 1: Notations and terminology

MULTIMEDIA DOCUMENT AUTHORING AND PRESENTATION ARCHITECTURE

Figure 5 at the end of the paper shows the overall architecture of the CHIMP multimedia document authoring and presentation system. In this section we briefly describe the components of this architecture.

(a) Filters for Accessing the Multimedia Document: Depending on their interests and their needs, authors editing a document may wish to *view* a document in different ways, through the use of *filters*. These filters describe the specific portions of the multimedia document that can be viewed or edited by an author/user. The filters that can possibly be applied to a document depend on

- User access rights.
- Local system capabilities.
- User's projection of the document.

Here, the user access rights and local system capabilities are system defined filters. Access rights describe the portions of the multimedia document accessible to the user. Local system capabilities, on the other hand, describe the facilities the user has for multimedia object handling. For example, if the local system cannot handle MPEG video streams, the author might filter out MPEG video objects and view the rest of the document. The user's projection of the document is a user defined filter that helps the author to access the portion of the document that s/he is interested in. The presentation must adapt itself to the changes, such as to the omissions or additions of new objects. Authors can perform certain edit operations that modifies only the local view of the multimedia document. When the author is satisfied with the edit operations s/he has performed, the changes can be realized on the system's document view.

(b) Presentation Schedule Generator: This component picks a presentation schedule that satisfies the given temporal specification. Temporal specifications may be either hard or flexible. In the case of hard temporal specification, the start times 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. For example, consider two temporal constraint specifications of the form

- (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 associated audio is played out.

Description (a) is a hard temporal specification whereas (b) is a flexible one. These temporal specifications, in other words, help in the derivation of a *presentation schedule* that describes the starting times and durations of the presentations of the objects composing a multimedia document. Flexible temporal specifications imply that a multimedia document may have a

set of different presentation schedules each of which satisfies the given temporal constraints. Each member of this presentation schedule set describes one possible *view* of the multimedia document. In our work, we deal with flexible temporal constraints. The section titled "Temporal specification and presentation schedule generation" provides an overview of the presentation schedule generator.

(c) Retrieval Schedule Generator: Depending on the access filters and the temporal specification, a schedule for retrieving objects has to be identified. The retrieval schedule specifies the time instants at which the author's system should make a request for retrieving the objects that compose the multimedia document. This retrieval schedule depends on the following factors:

- Presentation schedule (time instant and duration of presentation of an object).
- Size of the object(s) that is (are) to be retrieved from the server.
- Throughput (or the bandwidth), and delay of the communication channels.
- Buffer availability for the retrieved objects.

The temporal specification and the size of the objects, are application dependent whereas the last two factors, i.e. communication characteristics and buffer resources, are system dependent. The flexibility in the temporal specification of the multimedia document helps in deriving a flexible retrieval schedule that can adapt to the system dependent factors such as available throughput and buffer resources.

In addition, a multimedia document may involve objects of different formats. The local system may or may not have the capability to present an object using the same format in which it is stored. In this case, objects might have to be converted from the stored format to the desired format. This conversion can be done in one of two ways:

- The system that stores the object may accomplish the required conversion before transmitting the object.
- Alternatively, the system that stores the object may "route" the object to intermediate nodes that convert the object to the desired target format (or to an intermediate format which is later convert, by another node on the path, to the final target format).

The section titled "Retrieval Schedule Generation" describes how the retrieval schedule is generated from the listed inputs.

(d) Schedule Validator: Given a presentation schedule, system constraints, and a retrieval schedule, this module checks the validity of the generated retrieval schedule based on the application dependent and system dependent constraints. If the schedule is valid with respect to the specified constraints, then the *validator* returns them as the final solution. However, if the schedule does 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. The section titled "Retrieval Schedule Validation" describes in detail the functionality of the schedule validator module.

(e) 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 against the system constraints (throughput and buffer constraints).

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. There are several ways of defining what constitutes a "best" schedule. One possibility is to eliminate from the presentation, in consultation with the user, those objects whose schedules do not satisfy the system constraints. In this paper, we do not discuss ways of optimal relaxations of systems constraints.

TEMPORAL SPECIFICATION AND PRESENTATION SCHEDULE GENERATION

Our approach for temporal specification is to use a small class of the language of real valued linear constraints called *difference constraints*. Difference constraints have the form

$$x_1 - x_2 \leq b. \tag{1}$$

By using a variable to denote a multimedia presentation event (start of an object presentation or end of an object presentation), difference constraints may be used to create a flexible temporal specification, describing a *possible* range of values between two events. In addition, we will introduce the notion of priorities on temporal constraints – these priorities may then be used to resolve conflicts between constraints in a priority-optimal manner. Each object O in a multimedia document D, has an associated set of temporal constraints, T_O. With each multimedia object O in the document, we associate the following *temporal variables*:

- $st(O)$: Denotes the start time of the display of O.
- $et(O)$: Denotes the end time of the display of the O.
- $req(O)$: Denotes the time when the request for the object O is issued. This variable is not part of the user specification, it is used by the retrieval schedule generator.
- st_p and et_p denote the start and end of the presentation respectively.

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:

- $\mathcal{T}(o) \in \{st(o), et(o)\}$ and

- $t \in \bigcup_j \{st(o_j), et(o_j)\} \bigcup \{st_p, et_p\}$ where

st(o_j) and et(o_j) denote the start and end times, respectively, of the j'th "chunk" of object o. For instance, we may logically divide a video into a set of chunks (each of some number of frames) and refer the the start/end times of these chunks directly in the above specification.

Any set of temporal constraints obeying the following definition constitutes a temporal presentation specification. For instance, if the user chooses to set the duration of the display of an object o to exactly len_{des}, he can do so by the following constraints:

(1) $\quad et(o) - st(o) \leq len_{des}$
(2) $\quad st(o) - et(o) \leq -len_{des}$

Alternatively, the user might choose to give a desirable duration len_{des}, and a maximum duration $len_{alt} > len_{des}$ by adding the following constraint:

(3) $\quad et(o) - st(o) \leq len_{alt}$

The scheduler will first try to set the duration of the presentation of object o to len_{des} (since it is more restrictive), however if it can not accomplish this task, it will remove the constraint (2), and will use constraint (3) to set the duration.

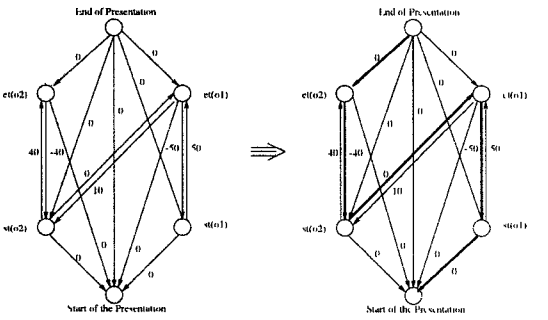
Suppose D is any document and T_D is the set of temporal constraints associated with D. This set of difference constraints may be translated into a graph $G = (V, E)$ defined as follows:

- **Vertices:** For each constraint variable τ_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).
- **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 δt. Furthermore, for each node v_i, there is an edge from v_i to v_s with weight 0 and from v_e to v_i with 0 weight.

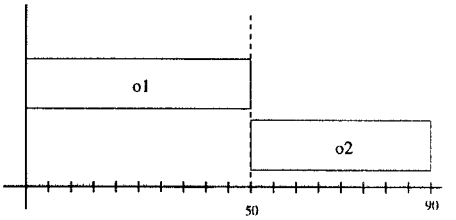
Thus, given any document D, we have a graph associated with its temporal specifications. The shortest path solution of this graph (from the end of presentation node to the start of presentation node) results in a schedule that satisfies the temporal specification. Example 1 shows how the temporal specification and its solution work.

Example 1 Let us assume that there exist two objects o_1 and o_2 with durations 40 and 50 seconds respectively. 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:

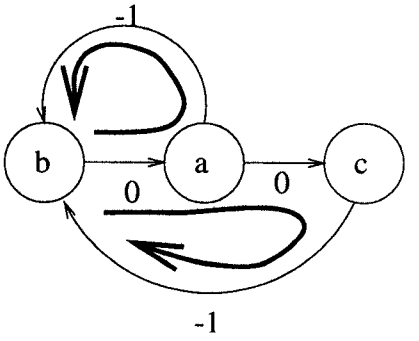
$st(o_1) - et(o_1) \leq -50 \qquad et(o_1) - st(o_1) \leq 50$
$st(o_2) - et(o_2) \leq -40 \qquad et(o_2) - st(o_2) \leq 40$
$st(o_2) - et(o_1) \leq 10 \qquad et(o_1) - st(o_2) \leq 0$



(a)



(b)



(c)

Figure 2: (a) A constraint graph, (b) corresponding solution, and (c) a constraint graph with a negative cycle

Figure 2(a) shows the constraint graph, and the shortest path solution. The corresponding presentation schedule is also shown in figure 2(b). Note that the time values in the schedule are shifted by 90 to make them positive:

End = 0	Start = -90
$et(o_2) = 0$	$st(o_2) = -40$
$et(o_1) = -40$	$st(o_1) = -90$

□

In general, the number of constraints in the graph is linearin the number of temporal specifications provided by the multimedia authors. It is well known that solving a set of difference constraints is equivalent to finding the shortest path in the graph associated with those constraints [8]. The constraint graphs can contain edges with negative weights. As an example, consider the following constraint specification.

Example 2 Assume the following set of specifications:
$(1a)\ a - b \leq 0 \qquad (2a)\ b - a \leq -1$

$(3a)\ c - a \leq 0 \qquad (4a)\ b - c \leq -1$

Here, (1a)-(2a), and (1a)-(3a)-(4a) are in conflict. The best way to handle this problem is to remove (1a), because both conflicts will be resolved by the deletion of a single constraint. However, any other solution would include at least two deletions, such as the removal of (2a) and (3a), which is undesirable. □

Such conflicting constraints are captured by the existence of negative cycles in the constraint graph. For instance, the set of specifications in Example 2 corresponds to the graph in figure 2(c). The elimintation of conflicting constraints requires that some constraints be weakened or eliminated. Three ways of eliminating constraints are presented below, all of which are based on a common (and simple) concept of constraint removal.

Definition 1 (Constraint Removal) Suppose C is a set of difference constraints. A *removal* of C is any subset $C' \subseteq C$ such that C' is solvable. □

The above definition allows any subset of C to be considered a removal. Thus, if c_1, c_2 are constraints in C, and if $(C - \{c_1\})$ is solvable then it must necessarily be the case that $(C - \{c_1, c_2\})$ is solvable. However, the latter removal of C eliminates "more constraints" than strictly needed. Below, we present three alternative definitions of *optimal constraint removal*. The third definition assumes that each constraint has an associated *priority* – a number greater than or equal to 1. The higher the priority, the more important the constraint.

Definition 2 (Optimal Constraint Removal) Suppose C is a set of difference constraints.

- a *card-optimal removal* of C is any subset $C' \subseteq C$ such that C' is solvable, and there is no other removal C'' such that $card(C'') > card(C')$. Here $card(C)$ is the number of constraints in C.
- a *pre-set-optimal removal* of C is any subset $C' \subseteq C$ such that C' is solvable, and there is no other removal C'' such that $C'' \supset C'$. A *set-optimal removal* of C is a *pre-set-optimal removal* C', and there is no other *pre-set-optimal removal* C" such that $card(C'') > card(C')$.
- a *priority-optimal removal* of C is a *set-optimal removal* C' of C such that there is no other *set-optimal removal* C'' which satisfies $\left(\sum_{c \in C''} \wp(c)\right) > \left(\sum_{c \in C'} \wp(c)\right)$ where $\wp(c)$ denotes the priority of constraint C. We assume that $\wp(c) \geq 1$ for all constraints c. □

Finding card-optimal removals of C is an NP-complete problem, while finding a set-optimal removal is solvable in polynomial time. Similarly, finding priority-optimal removals is solvable in polynomial time.

Presentation Schedule Generation Algorithms : Though most graph algorithms for computing shortest paths cannot handle negative edges in the graph, the well known Bellman-Ford shortest path algorithm can deal with negative edges[8]. If there is no negative cycle, the algorithm produces the shortest paths and their weights. The shortest path along with the associated weights specify, in effect, the start times and durations of presentations of the objects composing the multimedia document. If there is such a cycle, the algorithm terminates indicating that there is no solution. The presence of a negative cycle indicates conflicting constraints. However, the Bellman-Ford algorithm cannot remove constraints so as to restore solvability. Also, local editing of a multimedia document necessitates the handling of incremental additions and deletions of constraints. The reason for this is that everytime a document is modified, the set of associated constraints changes. Insertion of a new object into the presentation causes insertion of new constraints, while object deletion causes deletion of existing constraints. Object modification may cause both insertions and deletions of constraints. Hence, we have developed a set of algorithms (see Appendix A) that will do the following :

- Take as input a set of temporal constraints and return as output a presentation schedule that satisfies as many constraints as possible.
- Handle incremental additions and deletions of constraints such that a maximal set of constraints is satisfied at all times.

All these algorithms work in polynomial time, and the incremental addition/deletion algorithms are easily seen to perform significantly better than re-solving the constraints from scratch.

MULTIMEDIA DOCUMENT PRESENTATION

In this section, we describe the basic steps required for delivering a successful multimedia presentation: (1) The routing and conversion of the source objects, (2) the generation and validation of a retrieval schedule, and (3) the creation of feedback for guiding the validation process.

Routing and Conversions

In a collaborative environment, the collaborators and the objects may be located at different locations on the network. Hence, collaborators must know how different objects can be retrieved over the network. Furthermore, the site/machine hosting a given collaborator may have a limited set of display/edit capabilities. Hence, during the checking-out of an object o from an object store by a collaborator c, it may be necessary to convert o into a format usable by c. Of course, such conversions may cause a change in the quality of the object. Similarly, if the collaborator c does not have enough resources to display object c, he may request a lower quality version of the object o (most probably with a lower size) from the source. Hence, there may be more than one way of retrieving an object, each with different characteristics. A table of the following form provides a means by which a client can obtain information about the location of the objects available in the collaboration workspace, as well as their access characteristics:

Name	Type	Quality	Size	Path	Delay
A	gif	0.7	1000000	p_1	Δt_1
A	jpg	0.5	800000	p_2	Δt_2
B	mpeg	0.8	20000000	p_3	Δt_3
B	mpeg	0.6	10000000	p_3	Δt_4
...	...	...	...	...	...

The paths in the above description not only provide the rout-

ing information, but they also provide the conversions/compressions that have to be performed on the objects on the nodes over the paths. The problem for selecting a low cost path given quality constraints can be stated as follows :

MAXIMAL QUALITY AT MINIMAL COST OBJECT SYNTHESIS PROBLEM In this problem, the user wishes to first synthesize object o at the *maximal* quality-level possible and *subsequently* minimize the total cost. In other words, quality is the primary concern, while *cost* is to be minimized only after the optimal quality is achieved.

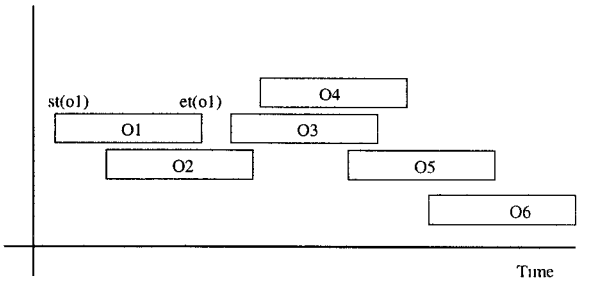
In [5], we have proposed an algorithm that makes a guided search over the possible paths, and returns those providing high quality and requiring low communication cost. In this framework, we benefit from this algorithm in creating the above table. Note, however, that in order to allow the client to choose from a set of possible qualities, we need to make sure that the table contains paths with a variety of different qualities. For each such path, the communication cost must be minimized. The retrieval scheduler then can initially use the high quality forms of the objects, and if it can not create a retrieval schedule for an object o, then it tries to create a schedule using a lower quality version of it. This retrieval process will be described in the next subsection.

Retrieval Schedule Generation

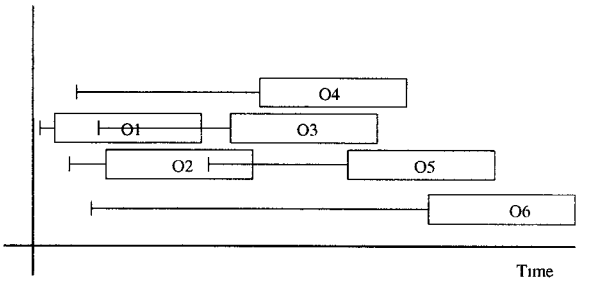
Retrieval schedule of a multimedia presentation specifies the time instants at which the client should make requests to the server(s) for delivering the objects that compose the presentation. The retrieval schedule is constrained by system dependent factors, such as the available throughput and available 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.
- 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).

The presentations of objects that are to be retrieved over the same network connection can overlap, as shown in Figure 3. Hence, the available throughput and buffer resources have to be shared among the objects to be retrieved. For instance, the



(a) Presentation Schedule



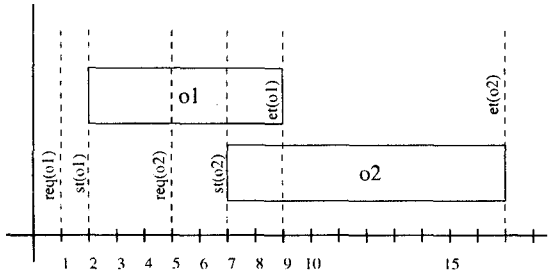
(b) Initial Retrieval Schedule

Figure 3: Parallel, Multiple Object Retrieval

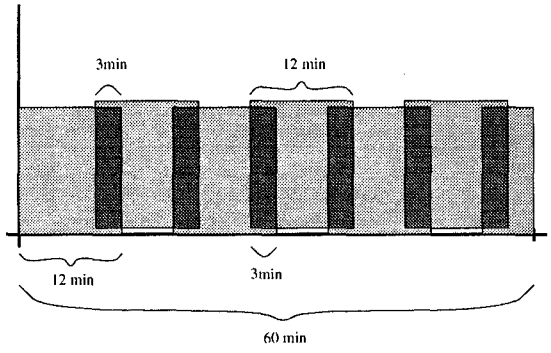
stream objects in figure 3(a) are initially assigned the following throughputs:

$$th(O1) = Th_{max}/2; \quad th(O2) = Th_{max}/2;$$
$$th(O3) = Th_{max}/3; \quad th(O4) = Th_{max}/3;$$
$$th(O5) = Th_{max}/3; \quad th(O6) = Th_{max}/2.$$

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 buffer requirements for an object start from the time, $rec(O)$, at which the first bit of information is received. Here, $rec(O) = req(O) + \Delta t$, where Δt is the round trip propagation time for sending the request and receiving a response. Δt can be found from the table described in the previous subsection. 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. Here, the value for $req(O)$ is: $req(O) = st(O) - b_{init}(O)/th(O) - \Delta t$, where $b_{init}(O)$ is the size (in bits) of the initial fraction of the object needed for its presentation. Figure 3(b) shows the overlap of the retrieval schedules of the objects in Figure 3(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 define a time interval (a, b) as the time period between the occurrence of two successive events a and



(a) Example 3



(b) Segmentation of a multimedia presentation

Figure 4: Schedule validation process

b, where an event is

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

Example 3 Consider two objects $o1$ and $o2$ that are scheduled as in figure 4(a). The throughput and the buffer requirements of the system in the different time intervals can be calculated as follows:

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

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

The buffer requirement, on the other hand, is

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

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

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

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

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

Note that during this interval, the followings also hold:

$$buf(o1, t) = b_{init}(o1),$$
$$buf(o2, t) = 0 \quad (\text{when } t < rec(o2)), \text{ and}$$
$$buf(o2, t) = \left(\frac{b_{init}(o2)}{(st(o2) - rec(o2))}\right) \times (t - rec(o2))$$
(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 their remaining portions:

$$\text{Th}_{tot} = \text{c}_s(o1) + \text{c}_s(o2)$$
$$b_{init}(o1) + b_{init}(o2) \leq \text{Buf}_{max}.$$

- **interval (9-17):** Only $o2$ receives its remaining information:

$$\text{Th}_{tot} = \text{c}_s(o2)$$
$$b_{init}(o2) \leq \text{Buf}_{max}.$$ □

Retrieval Schedule Validation

As discussed above, the generated retrieval schedule has to be checked to see whether it satisfies system constraints, such as throughput and buffer availability. 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 in segments of presentation, i.e. the presentation is divided into fixed-length segments, and each segment is processed only when required (figure 4(b)). 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 minimizing the reprocessing of already validated schedules. The algorithm presented in Appendix B shows how the presentation and retrieval schedules for a given segment s is validated. The input and output of this algorithm are as follows:

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 sched-

ule only if it can not change the retrieval schedule.

Schedule Modification

As described in the previous section, when the schedule validator detects a throughput or buffer violation, it creates a feedback, and it calls the schedule modifier to fix the schedule accordingly. There are three main types of feedback that influence the modification of a schedule:

- Modify the retrieval schedule.
- Modify the presentation schedule.
- Modify the quality of the presentation by reducing the object size.

Note that modification of retrieval schedules is the most desirable option, while the modification of object qualities is the least desirable option. Presentation schedules can be modified by selecting a new value for the presentation start time, $st(O)$, *within* the range specified by the constraints. The schedule modifier then generates a new schedule in which only the start times of the objects in S are modified while the presentation variables (i.e., the start and end presentation times) of other objects in the document are left unchanged. A detailed study of feedbacks is contained in [7].

The CHIMP Project

This paper is part of the Collaborative Heterogeneous Interactive Multimedia Platform (CHIMP) project, which has the goal of studying the technical aspects of collaborative multimedia document authoring and presentation, as well as building a system based upon these results.

Suppose we consider a team of individuals jointly *authoring* a multimedia document. In order to successfully author such a document, the authors must:

- Identify the objects (e.g. audio objects, video objects, text objects, etc.) that will be part of the authored multimedia document. This is studied in detail by Marcus and Subrahmanian[11, 12] who showed that a fragment of Datalog queries may be used to identify objects of interest.
- Specify how these objects should be presented to an end-user wishing to view the final multimedia document. This specification includes, amongst other things, the temporal constraints used to generate the presentation. In contrast to previous work[10, 3, 12] that describes how arbitrary constraints may be used to specify presentations, CHIMP benefits from the use of a small class of constraints called difference constraints that are adequate for specifying very flexible presentations.

In addition to the presentation constraints, when a set of objects are scattered across the network, we need to generate a *Retrieval Schedule* that specifies how the CHIMP retrieval engine will retrieve the desired objects from other locations by interacting with remote servers. This too, constitutes parts of the CHIMP project, and includes the study of resource reservation algorithms, as well as servers that can be used for networked delivery of multimedia objects keeping in mind, the available resources (which include bandwidth resources, buffer resources, available viewing formats at different nodes, etc.). CHIMP is currently implemented on the SUN/Unix platform, and includes some, but not all the services listed above.

RELATED WORK

One of the major differences between multimedia information and text information is the temporal nature of video and audio. Representing time and action has been an important problem in artificial intelligence. In [1, 2], James F. Allen proposes an algorithm based on intervals which, given a set of relationships among the intervals in the database, can infer the relationships among all intervals. Allen's work lacks quantitative aspects of time. The very basic model that addresses the quantitative needs of the multimedia applications is the *timeline model*. In this model, the user places events and actions on a timeline, and each event and action take place when its time arrives. Due to its simplicity, this model formed the basis for many multimedia authoring systems. However, the timeline model is too rigid for many purposes; users must specify exactly when an event occurs. Also, this model does not provide any flexibility which may be required by a retrieval scheduler (essential for distributed document authoring) which must use the available bandwidth provided by the network service provider.

In [10], Little and Ghafoor propose an interval based model based on object composition petri net (OCPN), a modification on timed petri net model. Prabhakaran and Raghavan, then, extend the OCPN model to handle similar user interactions [13]. Due to its higher flexibility and simplicity, this model proved to be very useful in multimedia authoring and multimedia simulating applications. Li, Karmouch, and Georganas, on the other hand, propose a Time Flow Graph (TFG) based model to model temporal presentation scenarios [9]. Their model, again, is based on intervals, and it can represent fuzzy presentation scenarios. In their framework, fuzziness can be due to unknown object presentation durations or unknown relative timing of the events. Buchanan and Zellweger use temporal constraints to specify temporal relationships among the multimedia objects within a document in their system called FireFly [3, 4]. Their work is based on the use of difference constraints for the specification of temporal information, like ours. However, they use the more expensive simplex algorithm (well known to take exponential time in the worst case) to solve for the presentation schedules. In our work, we use highly efficient graph algorithms for this purpose. Buchanan and Zellweger allow the user to specify an optimal duration for a multimedia object along with costs for shrinking and stretching it. We, on the other hand, allow the users to specify a range for the length of an object, along with an optimal value and a set of preferred durations. We can also model user interactions as dynamic local editings on the multimedia document. Marcus and Subrahmanian[12] proposed the use of a fragment of Datalog queries and showed that extending this fragment with arbitrary linear constraints could be used

to specify spatial and temporal characteristics.

Another research effort geared towards increasing the flexibility of multimedia document transfer is due to Thimm et.al. [18, 19]. Their idea is to change the quality of the multimedia objects in order to adapt to the changes in the network resource availabilities. They provide an application level solution which listens to the network, and which finds the best way to decrease/increase the quality of presentation to match the resources available. On the other hand, in our work, we suggest modifying the presentation schedule prior to the modification of the quality of object presentation. Raghavan, Prabhakaran, and Tripathi [16, 15] show how probabilistic attributed context free grammar rules can capture operational semantics of OCPN-based temporal specifications, as well as user and network inputs. In their work, the main concern was the network traffic generated by a multimedia presentation whereas in our work we deal with issues concerning authoring of a multimedia document.

SUMMARY AND CONCLUSION

In this paper, we have presented a unified framework for multimedia document authoring and presentation that facilitates the articulation and use of flexible temporal specification, access filters, local editing, object format conversions and flexible retrieval schedules for presenting objects over a computer network. Our main contributions are the following:

- Difference constraints based flexible temporal specification that allows the following :
  - Generation of flexible presentation schedules to adapt to the system requirements.
  - Handling of inconsistencies in temporal specifications.
  - Incremental modification of temporal constraints to facilitate local editing and access filters.
- Flexible retrieval schedule generation methodology that handles :
  - Variations in system parameters such as available network throughput and buffer resources.
  - Modifications of the presentation schedules due to local editing and access filtering.

The main difference between our work and that of others is that we utilize the flexibility in the temporal presentation specification to generate retrieval schedules which in turn can handle variations in network throughput and buffer resources.

REFERENCES

1. J.F. Allen (1983) *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM, vol. 26, no. 11, pp. 832-843, November 1983.

2. J.F. Allen (1984) *Towards a General Theory of Time and Action*, Artificial Intelligence, 23, pp. 123-154.

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

4. M.C. Buchanan and P.T. Zellweger (1993) *Automatically Generating Consistent Schedules for Multimedia Documents*, ACM/Springer-Verlag Journal of Multimedia Systems, vol. 1, no. 2, 1993.

5. 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.

6. K. Selçuk Candan, B. Prabhakaran, and V.S. Subrahmanian (1996) *Collaborative Multimedia Documents: Authoring and Presentation*, Technical Report: CS-TR-3596, UMIACS-TR-96-9, University of Maryland, College Park, Computer Science Technical Report Series, January 1996.

7. K. Selçuk Candan, B.Prabhakaran, and V.S. Subrahmanian (1996) *Retrieval Schedules Based on Resource Availability and Flexible Presentation Specifications*, Technical Report: CS-TR-3616, UMIACS-TR-96-21, University of Maryland, College Park, Computer Science Technical Report Series, 1996.

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

9. 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.

10. 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.

11. S. Marcus and V.S. Subrahmanian. (1995) Towards a Theory of Multimedia Database Systems. in: *Multimedia Database Systems: Issues and Research Directions* (eds. V.S.Subrahmanian and S. Jajodia), Springer Verlag.

12. S. Marcus and V.S. Subrahmanian. (1996) Foundations of Multimedia Database Systems, JOURNAL OF THE ACM, Vol. 43, 3, pps 474–523.

13. 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.

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

15. 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.

16. 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.

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

18. H. Thimm and W. Klas (1996) *δ-Sets for Optimal Reactive Adaptive Playout Management in Distributed Multimedia Database Systems*, 12'th International Conference on Data Engineering, pp. 584-592, February 1996.

19. H. Thimm, W. Klas, J. Walpole, C. Pu, and C. Cowan (1996) *Managing Adaptive Presentation Executions in Distributed Multimedia Database Systems*, submitted for publication.

Appendix A

A1 : Initialize-single-source (G, v_s)
1. **for** each vertex $v \in V[G]$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow NIL$
4. $d[s] \leftarrow 0$
5. all_cycles = $\emptyset$

A2 : RELAX (u, v, w)
1. **if** $d[v] > d[u] + w(u, v)$
2. then $d[v] \leftarrow d[u] + w(u, v)$
3. $\pi[v] \leftarrow u$

A3 : RELAX_and_MARK_CYCLE (u, v, w)
1. relaxed = 0
2. **if** $d[v] > d[u] + w(u, v)$
3. **then if** NOT_CYCLE (u, v)
4. **then** $d[v] \leftarrow d[u] + w(u, v)$
5. $\pi[v] \leftarrow u$
6. relaxed = 1;
7. return(relaxed)

A4 : NOT_CYCLE (u, v)
1. cycle = PATH (u, v)
2. **if** cycle $\neq \perp$
3. **then** $cycle = cycle \rightarrow v$
4. all_cycles = all_cycles $\cup$ {cycle}
5. return(0)
6. **else** return(1)

A5 : PATH (u, v)
1. **if** $\pi[u] = \perp$
2. **then** return($\perp$)
3. **if** $\pi[u] = v$
4. **then** return($v \rightarrow u$)
5. **else**
6. temp_path = PATH ($\pi[u]$, v)
7. **if** temp_path = $\perp$
8. return($\perp$)
9. **else** return(temp_path$\rightarrow u$)

A6 : SOLVE_and_MARK_CYCLE (G, w, s)
1. Initialize-single-source (G, s)
2. **for** $i = 1$ **to** $|V[G]| - 1$
3. **do for** each edge $(u, v) \in E[G]$
4. **do** RELAX_and_MARK_CYCLE (u, v, w)
5. **if** all_cycles $\neq \emptyset$
6. **then** G' = REMOVE_CYCLES (G, all_cycles)
7. **if** no deleted edge
8. **then** SOLVE_without_CYCLE_CHECK(G', w, s)
9. **else** temp_cycles = all_cycles
10. SOLVE_and_MARK_CYCLE(G', w, s)
11. all_cycles = temp_cycles $\cup$ all_cycles

A7 : SOLVE_without_CYCLE_CHECK (G, w, s)
1. Initialize-single-source (G, s)
2. **for** $i = 1$ **to** $|V[G]| - 1$
3. **do for** each edge $(u, v) \in E[G]$
4. **do** RELAX(u, v, w)

A8 : REMOVE_CYCLES (G, cycles)
1. Let cycles be (auth_cycles $\cup$ sys_cycles)
2. < `deleted_const`, `marked_const` > = CONSULT_AUTHORS (auth_cycles)
3. delete the constraints in `deleted_const` from the graph
4. mark the constraints in `marked_const` as unsatisfiable
{ At this point all the negative cycles in auth_cycles are removed }
5. let E be the set of edges in sys_cycles, and let $p(e)$ be the priority of the edge e
6. sort E with respect to the priorities in ascending order
7. remove duplicate negative cycles from sys_cycles
8. $e_m = 1$
9. **while** sys_cycles $\neq \emptyset$
10. **for** each unmarked edge $e \in E$ (starting from edge 1)
11. **do** $c[e]$ = COUNT_of_CYCLES(sys_cycles, e)
12. **if** $c[e] > c[e_m]$
13. **then** $e_m = e$
13. remove all the negative cycles containing e_m from sys_cycles
14. mark e_m as unsatisfiable

Appendix B

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 $(ThisInterval \geq FirstInterval)$ **do**

(a) Check if the interval $ThisInterval$ is *valid*

(b) **while** $ThisInterval$ is not *valid* and **do**

   i. Create a feedback (as specified in the section titled "Schedule Modification"). 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 (as specified in the section titled "Schedule Modification"). 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** return the schedules
7. **else** return empty schedule

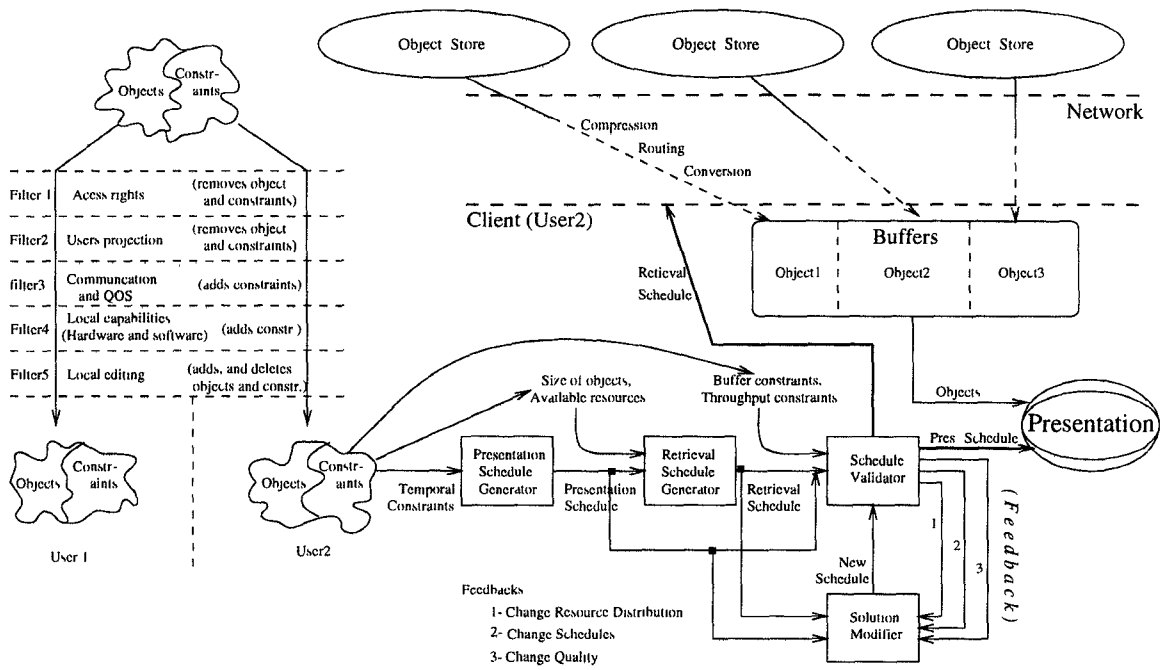


Figure 5: Multimedia Document Authoring and Presentation Architecture