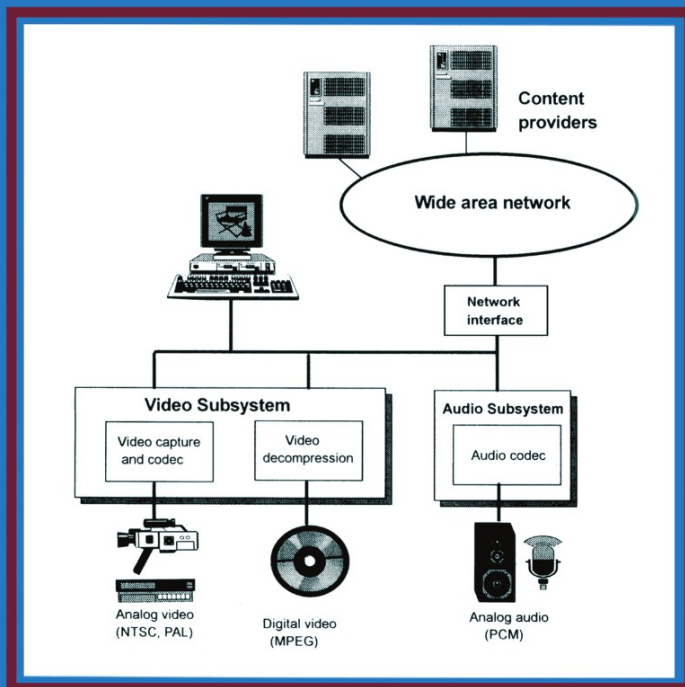


MULTIMEDIA SYSTEMS AND TECHNIQUES

edited by

Borko Furht



Amazon v. Audio Pod
US Patent 10,805,111
Amazon EX-1026

MULTIMEDIA SYSTEMS AND TECHNIQUES

**THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE**

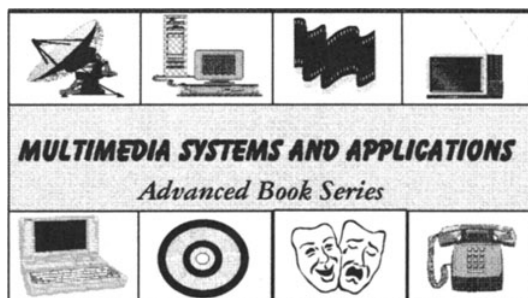
MULTIMEDIA SYSTEMS AND APPLICATIONS

Consulting Editor

Borko Furht
Florida Atlantic University

Recently Published Titles:

VIDEO AND IMAGE PROCESSING IN MULTIMEDIA SYSTEMS, by
Borko Furht, Stephen W. Smoliar, HongJiang Zhang
ISBN: 0-7923-9604-9



MULTIMEDIA SYSTEMS AND TECHNIQUES

edited by

Borko Furht
Florida Atlantic University



KLUWER ACADEMIC PUBLISHERS
Boston / Dordrecht / London

Distributors for North America:

Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061 USA

Distributors for all other countries:

Kluwer Academic Publishers Group
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

ISBN-13: 978-1-4612-8577-9
DOI: 10.1007/978-1-4613-1341-0

e-ISBN-13: 978-1-4613-1341-0

Copyright © 1996 by Kluwer Academic Publishers

Softcover reprint of the hardcover 1st edition 1996

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, mechanical, photo-copying, recording, or otherwise, without the prior written permission of the publisher, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061

Printed on acid-free paper.

4	Applications and New Technologies	107
5	Conclusion	117
	REFERENCES	118
4	MULTIMEDIA STORAGE SYSTEMS	
	<i>Harrick M. Vin and P. Venkat Rangan</i>	123
1	Introduction	123
2	Multimedia Storage Servers	125
3	Managing the Storage Space Requirement of Digital Multimedia	126
4	Efficient Retrieval of Multimedia Objects	132
5	Commercial Video Servers	140
6	Concluding Remarks	141
	REFERENCES	142
5	MULTIMEDIA NETWORKS	
	<i>Borko Furht and Hari Kalva</i>	145
1	Introduction	145
2	Traditional Networks and Multimedia	149
3	Asynchronous Transfer Mode	156
4	Summary of Network Characteristics	165
5	Comparison of Switching Technologies for Multimedia Communications	166
6	Information Superhighways	167
7	Conclusion	171
6	MULTIMEDIA SYNCHRONIZATION	
	<i>B. Prabhakaran</i>	177
1	Introduction	177
2	Language Based Synchronization Model	181
3	Petri Nets Based Models	185
4	Fuzzy Synchronization Models	197
5	Content-based Inter-media Synchronization	200
6	Multimedia Synchronization and Database Aspects	204
7	Multimedia Synchronization and Object Retrieval Schedules	206
8	Multimedia Synchronization and Communication Requirements	209
9	Summary and Conclusion	213

REFERENCES	214
7 INFOSCOPIES: MULTIMEDIA INFORMATION SYSTEMS	
<i>Ramesh Jain</i>	217
1 Introduction	217
2 Information and Data	220
3 Operations in InfoScopes	225
4 InfoScope Architecture	228
5 Knowledge organization	234
6 Interfaces	237
7 Example Systems	239
8 Conclusion and Future Research	250
REFERENCES	251
8 SCHEDULING IN MULTIMEDIA SYSTEMS	
<i>A. L. Narasimha Reddy</i>	255
1 Introduction	255
2 Data Organization	258
3 Disk Scheduling	259
4 Network Scheduling	275
5 General Discussion	289
REFERENCES	290
9 VIDEO INDEXING AND RETRIEVAL	
<i>Stephen W. Smoliar and HongJiang Zhang</i>	293
1 Introduction	293
2 Parsing	296
3 Representation and Classification	307
4 Indexing and Retrieval	311
5 Interactive Tools	315
6 Conclusion	318
INDEX	323

MULTIMEDIA SYNCHRONIZATION

B. Prabhakaran

*Department of Computer Science & Engineering,
Indian Institute of Technology, Madras, India*

1 INTRODUCTION

Multimedia information comprises of different media streams such as text, images, audio, and video. The presentation of multimedia information to the user involves spatial organization, temporal organization, delivery of the components composing the multimedia objects, and allowing the user to interact with the presentation sequence. The presentation of multimedia information can be either *live* or *orchestrated*. In live presentation, multimedia objects are acquired in real-time from devices such as video camera and microphone. In orchestrated presentation, the multimedia objects are typically acquired from stored databases. The presentation of objects in the various media streams have to be ordered in time. *Multimedia synchronization* refers to the task of coordinating this ordering of presentation of various objects in the time domain. In a live multimedia presentation, the ordering of objects in the time domain are implied and are dynamically formulated. In an orchestrated presentation, this ordering is explicitly formulated and stored along with the multimedia objects.

Synchronization can be applied to concurrent or sequential presentation of objects in the various media streams composing the multimedia presentation. Figure 1 shows an example of synchronization during sequential and concurrent presentation. Consider the concurrent presentation of audio and images shown in Figure 1. Presentation of the objects in the individual media streams, audio and image, is sequential. The points of synchronization of the presentation corresponds to the change of an image and the beginning of a new audio clipping. Multimedia presentation applications must ensure that these points of synchronization are preserved.

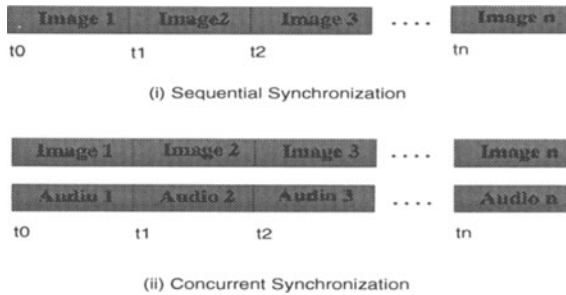


Figure 1 Synchronization in multimedia presentation.

The points of synchronization in a multimedia presentation can be modified by the user going through the presentation. In an orchestrated presentation, for example, a user may interact by giving inputs such as skip event(s), reverse presentation, navigate in time, scale the speed of presentation, scale the spatial requirements, handle spatial clash, freeze and restart of a presentation. User inputs such as skip, reverse presentation and navigate time modify the sequence of objects that are being presented. User inputs such as scaling the speed of presentation modify the presentation duration of the objects. User inputs such as handling spatial clash on the display screen make the corresponding media stream *active* or *passive* depending on whether the window is in the foreground or background. Similarly, while the freeze user input suspends the activities on all streams, restart input resumes the activities on all streams. In the presence of non-sequential storage of multimedia objects, data compression, data distribution, and random communication delays, supporting these user operations can be very difficult.

1.1 Modeling Temporal Relations

The points of synchronization in a multimedia presentation are described by the corresponding temporal information. Temporal information can be represented by *time points* or *time instants* and *time intervals*. Time instants are characterized by specifications such as *AT.9.00 AM* and time intervals by specifications such as *1 hour* or *9.00 AM to 5.00 PM*. Time interval is defined by two time instants : the start and the end. A time instant is a zero-length

moment in time whereas a time interval has a duration associated with it. A time interval can be defined as an ordered pair of instants with the first instant less than the second [5]. Time intervals can be formally defined as follows [13]. Let $[S, \leq]$ be a partially ordered set, and let a, b be any two elements of S such that $a \leq b$. The set $\{x | a \leq x \leq b\}$ is called an *interval* of S denoted by $[a, b]$. *Temporal relations* can then be defined based on the start and end time instants of the involved intervals. Given any two time intervals, there are thirteen ways in which they can relate in time [5], whether they overlap, meet, precede, etc. Figure 2 shows a timeline representation of the temporal relations. Six of the thirteen relations are inverse of the seven relations that are shown in Figure 2.

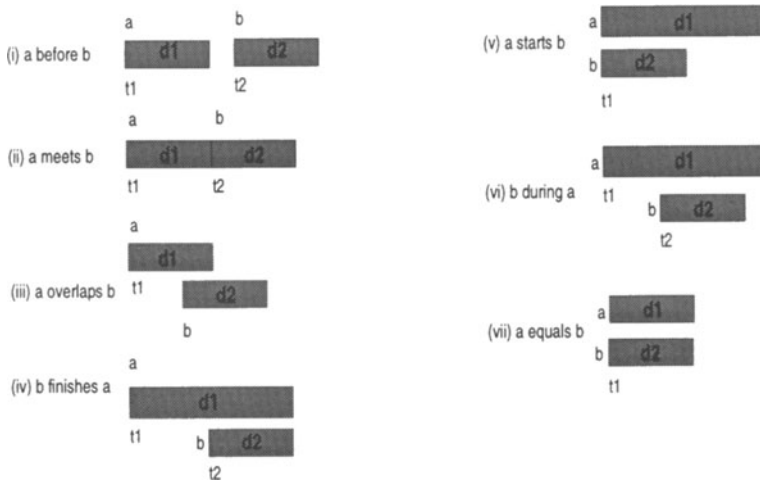


Figure 2 Temporal relations between two events.

Multimedia presentation can be modeled by temporal intervals with the time and duration of presentation of the multimedia objects being represented by individual time intervals. The time dependencies of the multimedia objects composing the presentation are brought out by their temporal relationships. In some cases, it may not be possible to formulate precise temporal relationships among the multimedia objects. However, *relative* temporal relationships can be specified among the objects to be presented. This mode of *fuzzy representation* or the relative temporal specification is to be contrasted with the *absolute* specification. As an example, the specification *present object A at time T* is an absolute temporal relation. Whereas *present object A with (or after) object B* is

a relative temporal specification. Consider a scenario where n objects are to be presented and the exact temporal requirements of the composing objects are not available. Using fuzzy temporal specification, one can formulate a relationship saying that all the $n - 1$ objects are to be presented *during* the presentation of the object that has the longest duration.

1.2 Synchronization Specification Methodologies

Multimedia synchronization specification in terms of the temporal relationships can be absolute or relative. Even in the case of absolute temporal relations, the time deadlines are *soft*. For example, nothing untoward happens if the object is not available for presentation at the scheduled time instant. Methodologies used for synchronization specification should capture all the temporal interdependencies of the objects composing the multimedia presentation and at the same time should have the flexibility for indicating *soft* deadlines. A synchronization methodology should also allow the user to modify the sequence of presentation dynamically. The temporal relationships in orchestrated multimedia presentations have to be captured in the form of a database and a specification methodology must be suitable for developing such a database.

One approach to the specification of multimedia synchronization is to use parallel language paradigms that can be parsed during presentation to the user. Here, the presentation of individual media streams are assigned to executable processes and these processes synchronize via inherent language constructs. Language based schemes for multimedia synchronization have been proposed in [9, 11, 14]. Another approach is to use graphical models. Graphical models have the additional advantage of providing a visual representation of the synchronization specification. Graphical models for synchronization specification have been based on Petri nets [1] and Time-flow Graphs [19]. Another new approach to synchronization specification has been proposed in [21]. This proposal, called the *content-based synchronization*, is based on the semantic structure of the objects composing the multimedia presentation.

Outline of this Chapter : In this chapter, we discuss the various methodologies used for describing multimedia synchronization in an orchestrated multimedia presentation. In Section 2, we examine language based models for multimedia synchronization. In Sections 3 and 4, we discuss graphical models based on Petri nets and Time-flow Graphs. In Section 5, we study the content-based approach for synchronization specification. After discussing the

synchronization methodologies, we present the database storage aspects of multimedia synchronization in Section 6. We then consider distributed orchestrated multimedia presentations where objects have to be retrieved over a computer network before their delivery to the user. In Sections 7 and 8, we describe how a multimedia synchronization specification can be applied to derive schedules for object retrieval over computer networks and to derive the communication requirements of an orchestrated presentation.

2 LANGUAGE BASED SYNCHRONIZATION MODEL

Language based synchronization models view multimedia synchronization in the context of operating systems and concurrent programming languages. Based on an object model for a multimedia presentation, the synchronization characteristics of the presentation are described using concurrent programming languages. As discussed in [9], the basic synchronization characteristics of a multimedia presentation as viewed from the point of view of programming languages are :

- number of multimedia objects involved
- blocking behavior of individual multimedia objects while waiting for synchronization
- allowing more than two multimedia objects combine many synchronization events
- ordering of synchronization events
- coverage of real-time aspects

The above characteristics influence the basic features required from a concurrent programming language in order to describe a multimedia presentation. For example, the *number of involved objects* describes the communication among the multimedia objects that should be described by the language. *Blocking behavior of a multimedia object* describes the nature of synchronization mechanism that is required *viz.* blocked-mode, non-blocked mode and restricted blocked mode. The non-blocked mode would never force synchronization but allows objects to exchange messages at specified states of execution. Blocked

mode forces an object to wait till the cooperating object(s) reach the specified state(s). *Restricted blocked mode* is a concept introduced in [9]. In the context of multimedia presentation, restricted blocking allows an activity to be repeated by an object till the synchronization is achieved with the cooperating object(s). The activity can be presenting the last displayed image for an image object or playing out a pleasant music in the case of an audio object. Restricted blocking basically allows the multimedia presentation to be carried out in a more *user friendly* manner.

More than two objects may have to be allowed to combine many synchronizing events. This is especially true in presentations where many streams of information are presented concurrently. A programming language should be able to describe this scenario. The ordering of synchronization events become necessary when two or more synchronization events are combined by some kind of a relation. The ordering can be *pre-defined*, *prioritized* or *based on certain conditions*. As multimedia presentations deal with data streams conveying information to be passed in real-time, synchronization of the events composing the presentation incorporates real-time features.

2.1 Enhancements For Concurrent Programming Languages

Following the above discussions, additional features are necessary in a concurrent programming language for describing real-time features and for describing restricted blocking. In this section, we discuss these enhancements for a concurrent programming language suggested in [9].

Real-Time Synchronization Semantics : In a multimedia presentation, events have to be synchronized *at* specified times or *within* a time interval. As an example, video images and the corresponding audio have to be synchronized within 150 milliseconds. In [9], following terms of multimedia relations are introduced for a concurrent programming language.

- *Timemin* is the minimum acceptable delay between two synchronization events. The usual parameter is "0".
- *Timeave* is the ideal delay between synchronization events i.e., the synchronization should be as close to timeave as possible, though it might occur before or after timeave. The usual timeave parameter is also "0".

Expression	Meaning
<code>timemin(...)</code>	Minimum acceptable delay between two synchronizing events.
<code>timeave(...)</code>	Ideal delay.
<code>timemax(...)</code>	Maximum acceptable delay.
<code>timemin(...) AND timemax(...)</code>	Synchronization : between <code>timemin</code> and <code>timemax</code> .
<code>timemin(...) AND timeave(...)</code>	Synchronization : closer to <code>timeave</code> , never before <code>timemin</code> .
<code>timemax(...) AND timeave(...)</code>	Synchronization : closer to <code>timeave</code> , never after <code>timemax</code> .
<code>timemin(...) AND timemax(...) AND timeave(...)</code>	Synchronization : closer to <code>timeave</code> , between <code>timemin</code> and <code>timemax</code> .

Table 1 Semantics For Basic Real-Time Synchronization.

- *Timemax* is the *time-out* value and refers to the maximum acceptable delay between synchronization events. The parameter for *timemax* vary depending on the type of media objects involved in the synchronization.

The proposal in [9] also allows the above specified *temporal operands* to be combined. The operands *timemin* and *timemax* are not commutative, thereby allowing specification of a different delay depending on which multimedia object executes its synchronizing operation first. The temporal operands can be combined using logical operator AND. Table 1 describes the possible temporal expressions and their associated meaning.

Restricted Blocking Semantics : Figure 3 shows a possible multimedia presentation scenario with restricted blocking. Here, activity A presents a full motion video while the activity B presents the corresponding audio. Activity A, if it first waits for synchronization with activity B, keeps displaying the last presented image. In a similar manner, activity B, if it first waits for synchronization with activity A, plays out pleasant music. The proposal in [9] describes a *WHILE_WAITING* clause for specifying restricted blocking. The program segment in Table 2 shows the synchronization between the activities A and B (shown in Figure 3) with restricted blocking and real-time restrictions. The synchronization delay between the two objects should be between 1 second of the audio object ahead of the video object and 2 seconds of the video object

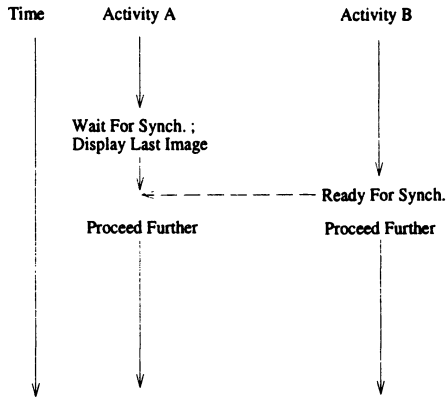


Figure 3 Restricted blocking in multimedia presentation.

program of Object_A :	program of Object_B :
- video	- audio
- from Source A	- from Source B
display (Object_A)	display (Object_B)
SYNCHRONIZE WITH	SYNCHRONIZE WITH
Object_B AT end	Object_A AT end
MODE restricted_blocking	MODE restricted_blocking
WHILE_WAITING	WHILE_WAITING
display(last_image)	play (music)
TIMEMIN 0	TIMEMIN 0
TIMEMAX 1s	TIMEMAX 2s
TIMEAVE 0	TIMEAVE 0
EXCEPTION display(last_image)	EXCEPTION play(music)

Table 2 Program With Restricted Blocking & Time Constraints

ahead of the image object. However, the target is no delay (with timemin being 0).

Summary : Language based specification of the multimedia synchronization characteristics uses concurrent programming language features. Enhancements

to the facilities offered by the programming languages are needed for describing certain features of multimedia presentation such as real-time specification and restricted blocking. However, language based specifications can become complex when a multimedia presentation involves multiple concurrent activities that synchronize at arbitrary time instants. Also, modifying the presentation characteristics might even involve rewriting the program based specification.

3 PETRI NETS BASED MODELS

Petri Nets, as a modeling tool, have the ability to describe real-time process requirements as well as interprocess timing relationships, as required for multimedia presentations [1, 2]. A Petri net is a bipartite graph consisting of 'place' nodes and 'transition' nodes. Places, represented by circles, are used to represent conditions; transitions, drawn as bars, are used to represent events. A *Petri net structure* P is a triple,

$P = \{ T, P, A \}$ where

$T = \{ t_1, t_2, \dots \}$ represents a set of transitions (bars)

$P = \{ p_1, p_2, \dots \}$ represents a set of places (circles)

$A : \{ T * P \} \cup \{ P * T \} \rightarrow I, I = \{ 1, 2, \dots \}$ representing set of integers

The arcs represented by A describe the *pre*- and *post*-relations for places and transitions. The set of places that are incident on a transition t is termed the *input places* of the transition. The set of places that follow a transition t is termed the *output places* of the transition. A Marked Petri Net is one in which a set of 'tokens' are assigned to the places of the net. Tokens, represented by small dots inside the places, are used to define the execution of the Petri net, and their number and position change during execution. The marking (M) of a Petri net is a function from the set of places P to the set of nonnegative integers I , $M : P \rightarrow I$. A marking is generally written as a vector $(m_1, m_2, \dots, m_n)$ in which $m_i = M(p_i)$. Each integer in a marking indicates the number of tokens residing in the corresponding place (say p_i).

Execution of a Petri net implies *firing* of a transition. In a Marked Petri net, a transition t is enabled for firing iff each of its input place contain atleast one

token, i.e.,

$$\forall (p \in \text{InputPlace}(t)) : M(p) \geq 1.$$

Firing t consists of removing one token from each of its input place and adding one token to each of its output place, and this operation defines a new marking. An execution of a Petri net is described by a sequence of markings, beginning with an initial marking M_o and ending in some final marking M_f . Marking of a Petri net defines the *state* of the net. During the execution, Petri net moves from a marked state M_i to another state M_j by firing any transition t that are enabled in the state M_i .

For the purpose of modeling time-driven systems, the notion of time was introduced in Petri nets, calling them as Timed Petri Nets (TPN) [4]. In TPN models, the basic Petri net model is augmented by attaching an execution time variable to each node in the net. The node in the Petri net can be either the place or the transition. In a marked TPN, a set of tokens are assigned to places. A TPN N is defined as,

$$N = \{ T, P, A, D, M \} \text{ where}$$

M : P → I represents marking M which assigns tokens (dots) to each place in the net

D : P → R represents durations as a mapping from the set of places to the set of real numbers

other descriptions T, P, A are same as that of the *normal* Petri net.

A transition is enabled for execution iff each of its input place contain atleast one token. The firing of a transition causes a token to be held in a locked state for a specified time interval in the output place of the transition, if the time intervals are assigned to places. If time intervals are assigned to transitions, the token is in a transition state for the assigned duration. The execution of a Petri net process might have to be interrupted in order to carry out another higher priority activity. Pre-emption of the on-going execution of a Petri net process can be modeled using escape arcs to interrupt the execution of a process [6]. In this section, we examine Petri net based models for the purpose of describing the synchronization characteristics of the multimedia components.

3.1 Hypertext Models Using Timed Petri Nets

TPN concept has been used to specify the browsing semantics in the Trellis Hypertext System [7, 10]. This Petri net based Hypertext model allows browsing events to be initiated by the reader or by the document itself. A synchronous hypertext H is defined as a 7-tuple $H = \langle N, M_o, C, W, B, P_l, P_d \rangle$ in which

N is a timed Petri net structure

M_o is an initial state (or initial marking) for N

C is a set of document contents

W is a set of windows

B is a set of buttons

P_l is a logical projection for the document

P_d is a display projection for the document.

In the above definition, the structure of the timed Petri net specifies the structure of the hypertext document. A marking in the hypertext hence represents the possible paths through a hyperdocument from the browsing point it represents. The initial marking of a synchronous hypertext therefore describes a particular browsing pattern. The definition also includes several sets of components (*contents*, *windows*, and *buttons*) to be presented to the user going through the document. Two collections of mappings or *projections* are also defined: one from the Petri net to the user components and another from the user components to the display mechanism.

The content elements from the set C can be text, graphics, still image, motion video, audio information or *another hypertext*. A *button* is an action selected from the set B . A *window* is defined as a logically distinct locus of information and is selected from the set W . P_l , the logical projection, provides a mapping from the components of a Petri net (place and transitions) to the *human-consumable* portions of a hypertext (contents, windows and buttons). A content element from the set C and a window element for the abstract display of the content from the set W , are mapped to each place in the Petri net. A logical button from the set B is mapped to each transition in the Petri net. P_d , the display projection, provides a mapping from the logical components of

the hypertext to the tangible representations, such as screen layouts, sound generation, video, etc. The layout of the windows and the way text information and buttons are displayed, is determined by P_d .

Execution Rules For Trellis Hypertext

The net execution rule in Trellis hypertext is a hybrid one, with a combination of a singleton transition firing rule (representing the actions of the hyperdocument reader) and a maximal transition firing rule (to allow simultaneous automatic or system-generated events). The execution rule works as follows.

- All the enabled transitions are identified. The enabled transitions set consists of both *timed out* ones (i.e., have been enabled for their maximum latency) and *active* ones (i.e., have not timed out).
- One transition from the group of active ones and a maximal subset from the group of timed-out transitions will be chosen for firing.

This two step identification of transitions to be fired is basically for modeling both reader and timer based activity in the hypertext. A reader may select a button (fire a transition) during the same time slot in which several system-generated events occur. In this case, the Trellis model gives priority to the reader's action by choosing the reader-fired transition first. When a transition is fired, a token is removed from each of its input place and a token is added to each of its output place.

The implication of the firing rules is such that the document content elements are displayed as soon as tokens arrive in places. The display is maintained for a particular time interval before the next set of outgoing transitions are enabled. After a transition t becomes enabled, its logical button is made visible on the screen after a specified period of time. If the button remains enabled without being selected by the reader, the transition t will fire automatically at its point of maximum latency. The way the hypertext Petri net is structured along with its timings describes the browsing actions that can be carried out. In effect, windows displaying document contents can be created and destroyed without explicit reader actions, control buttons can appear and disappear after periods of inactivity, and at the same time user interactive applications can be created with a set of active nodes. Petri nets being a concurrent modeling tool, multiple tokens are allowed to exist in the net. These tokens can be used to effect the

action of multiple users browsing one document or to represent multiple content elements being visible together.

Trellis Hypertext : An Example

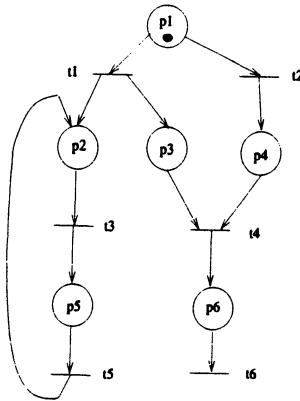


Figure 4 Guided tour using trellis hypertext.

We shall consider the guided tour example discussed in [7]. In a guided tour, a set of related display windows is created by an author. All the windows in a set are displayed concurrently. A tour is constructed by linking such sets to form a directed graph. The graph can be cyclic as well. From any one set of display windows, there may be several alternative paths. Figure 4 shows a representation of guided tour using the Trellis hypertext. Here, the set of windows to be displayed at any instant of time is described by a Petri net place. A place is connected by a transition to as many places as there are sets of windows to be displayed. A token is placed in the place(s) representing the first set of windows to be displayed. The actual path of browsing is determined by the user going through the information. For example, when the token is in *p1*, the information contents associated with *p1* are displayed and the buttons for the transitions *t1* and *t2* are selectable by the user.

Summary : The synchronous hypertext models both the logically linked structure of a hyperdocument and a timed browsing semantics in which the reader can influence the browsing pattern. The Trellis Hypertext model uses Petri net structure for representing user interaction with a database (basically for browsing actions). However, in the Trellis model, the user's inputs such as

freeze and restart actions, scaling the speed of presentation (fast forward or slow motion playout) and scaling the spatial requirements cannot be modeled. Another aspect in the Trellis model is that the user can interact with the hypertext only when the application allows him to do so (i.e., when the *buttons* can be selected by the user). Random user behavior where one can possibly initiate operations such as skip or reverse presentation at *any* point in time, are not considered in Trellis hypertext.

3.2 The Object Composition Petri Nets

The Object Composition Petri Net (OCPN) model has been proposed in [8, 12] for describing multimedia synchronization. The OCPN is an augmented Timed Petri Nets (TPN) model with values of time represented as durations and resource utilization being associated with each place in the net. An OCPN is defined as

$C_{OCPN} = \{ T, P, A, D, R, M \}$ where

$R : P \rightarrow \{ r_1, r_2, \dots \}$ represents the mapping from set of places to a set of resources.

other descriptions being same as that in the TPN model.

The execution of the OCPN is similar to that of TPNs where the transition firing is assumed to occur instantaneously and the places are assumed to have states. The firing rules for the OCPN are as follows :

1. A transition t_i fires immediately when each of its input place contain an *unlocked* token.
2. Upon firing the transition t_i , a token is removed from each of its input place and a token is added to each of its output place.
3. A place p_i remains in an *active* state for a specified duration τ_i associated with the place, after receiving a token. The token is considered to be in a *locked* state for this duration. After the duration τ_i , the place p_i becomes inactive and the token becomes *unlocked*.

It has been shown in [8, 12], that the OCPN can represent all the possible thirteen different temporal relationships between any two object presentation. The Petri nets have hierarchical modeling property that states that *subnets* of a Petri net can be replaced by equivalent abstract places and this property is applicable to the OCPN as well. Using the subnet replacement, an arbitrarily complex process model composed of temporal relations can be constructed with the OCPN by choosing pairwise, temporal relationships between process entities [8]. The OCPNs are also proved to be deterministic since no conflicts are modeled, transitions are instantaneous, and tokens remain at places for known, finite durations. OCPNs are also demonstrated to be *live* and *safe*, following the Petri nets definitions for these properties.

Synchronization Representation Using OCPN : An Example

We can consider a specific example orchestrated presentation and use the example to describe its OCPN representation. Figure 5 illustrates the media streams representation of the example, comprising of four streams of information : audio (A), motion video (V) and two image (I_1, I_2) streams. The synchronization characteristics of the presentation is such that audio and video objects are synchronized for every object. Image stream I_1 synchronizes with audio and video streams once in three object intervals and the stream I_2 synchronizes at the start and the end of presentation.

Figure 6 illustrates the OCPN model of the multimedia presentation example shown in Figure 5. The transitions shown in the net represent the points of synchronization and the places represent the processing of information. When the audio object represented by the place a_1 is presented completely, the token gets unlocked. Since the video object represented by the place v_1 is presented for the same duration as a_1 , the place v_1 unlocks its token synchronously with place a_1 . Now the common transition is fired immediately, thereby allowing the next set of video and audio objects to be presented. In a similar manner, the presentation of image object x_1 synchronizes with a_2 and v_2 and x_2 synchronizes with a_4 and v_4 .

Summary : The OCPN represents a graph-based specification scheme for describing the temporal relationships in an orchestrated multimedia presentation. Comparing the OCPN with the Trellis Hypertext model, the Trellis model can have multiple outgoing arcs from places, and therefore can represent non-deterministic and cyclic browsing, with user interactions. The OCPN specifies

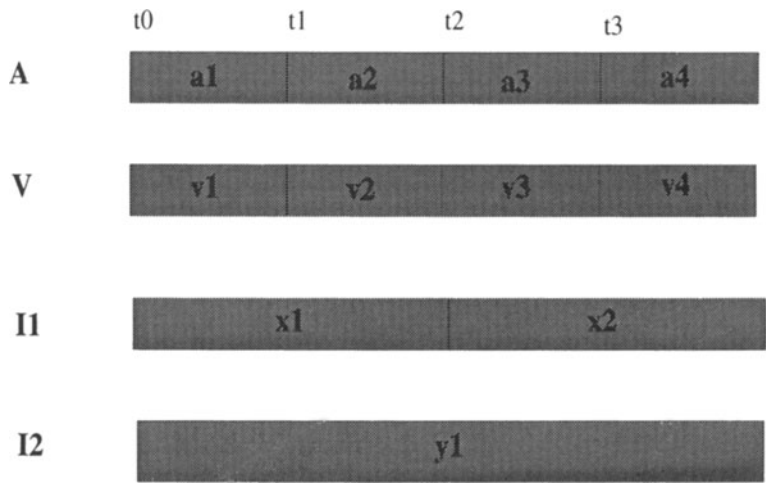


Figure 5 An example of multimedia presentation.

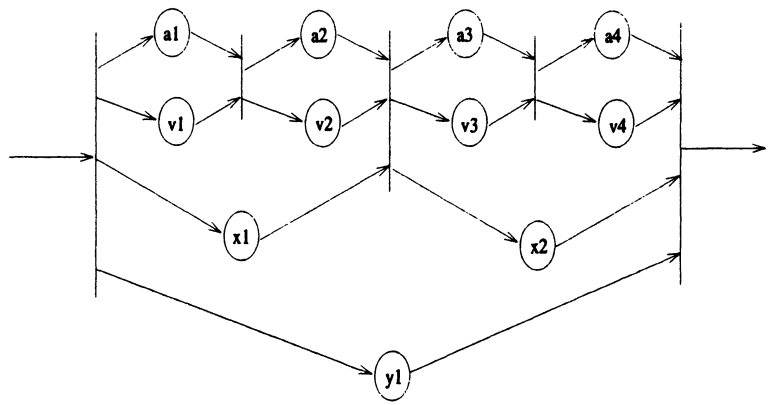


Figure 6 OCPN model for example in Figure 6.5.

exact presentation-time play-out semantics and hence useful in presentation orchestration.

The OCPN model ignores the spatial considerations required for composition of multimedia objects. The spatial characteristics can be assigned to the resource component of the OCPN model. Modeling the occurrence of spatial clashes (when two processes require the same window on the screen), however, cannot be done using the OCPN. Also, the OCPN model does not provide many facilities for describing user inputs to modify the presentation sequence. For instance, the user's wish to stop a presentation, reverse it, or skip a few frames cannot be specified in the existing OCPN architecture. Also, user inputs for freezing and resuming an on-going presentation, or scaling the speed or spatial requirements of a presentation cannot be described by the OCPN model.

3.3 Dynamic Timed Petri Nets

A *Dynamic Timed Petri Nets* (DTPN) model has been suggested in [16] by allowing user defined 'interrupts' to pre-empt the Petri net execution sequence and modify the time duration associated with the pre-empted Petri net process. In the DTPN model, nonnegative execution times are assigned to each place in the net and the notion of instantaneous firing of transitions is preserved. Basically, following types of modifications of execution time after pre-emption are allowed :

1. Deference of execution :
For the operation of pre-emption with deference of execution, the remaining duration associated with the pre-empted Petri net place is changed considering the time spent till its pre-emption.
2. Termination of execution :
The operation of pre-emption with termination is like premature ending of the execution.
3. Temporary modification of remaining time duration :
The operation of pre-emption with modification of execution time is like 'setting' the time duration associated with the pre-empted Petri net *place* to a 'new value', as appropriately determined by the type of user input. For temporary modification of execution time, the *remaining time duration* associated with the place is modified.
4. Permanent modification of execution time duration :
For permanent modification, the *execution time duration* associated with the place is modified.

DTPN Structure

In the DTPN model, pre-emption is modeled by using *escape arcs*. Escape arcs are marked by dots instead of arrow heads and they can *interrupt* an *active* Petri net place. Modification of execution time duration (temporary or permanent) associated with a place is modeled by *modifier arc*. Modifier arcs are denoted by double-lined arcs with arrow heads. A Dynamic Timed Petri Nets is defined as

DTPN = { T, P, A, D, M, E, C, C_m } where

C represents a set of escape arcs, C is a subset of $P * T$, and A and C are disjoint sets

C_m represents a set of modifier arcs, C_m is a subset of $P * T$, and A, C and C_m are disjoint sets

$E : P \rightarrow R$ represents *remaining duration* as a mapping from the set of places to the set of real numbers. Initially, E is set to *INVALID* before the activation of a place. After activation, E is set to a value equal to D. In case of pre-emption, E maintains the remaining time duration for which execution is to be carried out if the place is made active *again* by the firing of an input transition. After completion of the execution duration, E is set to *INVALID* again.

other descriptions being same as that in the TPN model.

A place p in a DTPN is an *escape place* of a transition t iff (p, t) is a member of the set C . The set of all escape places of t is denoted by $Esc(p)$, and the set of transitions connected by escape arcs with a place p is denoted by $Esc(t)$, $Esc(p) = \{t \mid p \in Esc(t)\}$. A transition is called a modifier transition if it has a modifier arc as an output arc. This notation is generalized in the usual way for sets of places and transitions. A typical DTPN structure with escape and modifier arcs is shown in Figure 7. The execution rules of the DTPN are discussed in [16]

Synchronization Models Using DTPN

The proposed DTPN can be effectively used in synchronization models for flexible multimedia presentation with user participation. DTPN constructions

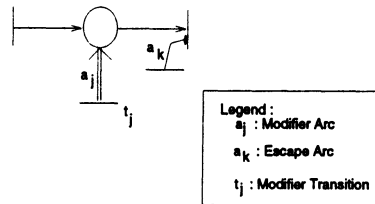


Figure 7 Dynamic Timed Petri Nets structure.

for describing handling of user inputs such as reverse presentation, freeze and restart to a *single object presentation* is the simplest case. These DTPN constructions for handling user inputs on single object presentations can be used in full multimedia presentation.

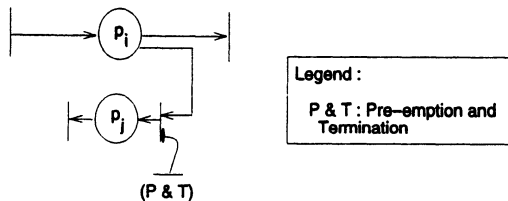


Figure 8 Reverse operation.

The reverse operation can be modeled as shown in Figure 8. Execution of p_i is pre-empted and terminated, and the transition to enable reverse presentation is fired. In a similar manner, freeze and restart operation can be modeled as shown in Figure 9. The type of interrupt to be applied to the execution sequence is pre-emption with deference of execution. The remaining time duration associated with p_i is modified to reflect the time spent till its pre-emption and the locked token in p_i is removed. A token is added to the output place p_j of the pre-empting transition. A large time duration is associated with p_j and its execution is pre-empted only on the receipt of *restart* signal from the user. Pre-emption of p_j causes the token to be 'returned' to the place p_i .

pre-emption followed by temporary modification of execution time (P&TM) for y_1 .

Summary : The DTPN model, its structure and the associated execution rules, can be adopted by the OCPN where resource utilizations are also specified. This augmented model can describe multimedia presentations with dynamic user participations.

4 FUZZY SYNCHRONIZATION MODELS

In [19, 20], multimedia synchronization involving independent sources for teleorchestra applications (remote multimedia presentation) has been considered. In the teleorchestra application, a user creates multimedia presentations using data objects stored in a distributed system. The distributed nature of the data objects may result in the non-availability of precise temporal relationships among the objects. However, *relative* or *fuzzy* temporal relationships can be specified among the objects to be presented. A Time-flow Graph (TFG) model for describing fuzzy temporal relationships has been proposed in [19, 20] where *temporal interval* is taken as a primitive. The terms *object X* and *interval X* are considered synonymous. One temporal interval may contain several objects in different media, i.e., several concurrent multimedia intervals.

4.1 Time-flow Graph Model

There are thirteen relations (seven relations and their inverses) that are considered between any two temporal intervals [5]. The relations are 'meets', 'before', 'equal', 'overlap', 'during', 'start', and 'finish'. The other relations are their inverse, e.g., 'di' is the inverse of the relation 'd' (during). The following set R describes all the thirteen possible temporal relations.

$R = \{b, e, m, o, d, s, f, bi, mi, di, oi, si, fi\}$.

The temporal relationships among the involved objects can be parallel or sequential.

Sequential Relations in TFG : The sequential relation between any two intervals can be either 'meets' (**m**) or 'before' (**b**). In teleorchestra applications, multiple intervals can be involved in sequential relation requirement. Hence, the sequential relation specification provided in TFG are :

1. $A\{B\}$: Interval(s) in B will start after all the intervals in A are finished.
2. $< A > B$: Interval(s) in B will start when one of the intervals (the first) in A is finished.

Parallel Relations in TFG : A subset R_f ($R_f \in R$) is defined as $R = \{e, o, d, s, f, oi, di, si, fi\}$.

Considering two object intervals X and Y, $X(r)Y$ describes the temporal relationship between the two intervals. When $r = 's'$, the relation $X(s)Y$ specifies that object X and Y are to be displayed with the same start time. In a tele-orchestra application scenario, the temporal relationships have to be specified despite the lack of duration information of the involved intervals. The presentation semantics are hence defined in TFG models as follows [19, 20].

1. $X(d)Y(d)Z$: All the other objects are displayed during the presentation of the object with the longest presentation duration.
2. $X(e)Y(e)Z$: The display of all the objects are started simultaneously. The presentation of all the other objects will be cut off when one of the objects (first) is finished.
3. $ch(X,Y,Z) = Y$: The presentation duration of all the involved objects should equal the one chosen. Objects are displayed according to the relations $r \in R_f$ specified between every two of them. Presentation of some of the objects might be cut off to equal the chosen duration.

Hence, three duration specifications $r \in R_m = \{d, e, ch\}$, can be applied to presentations involving concurrent multiple intervals.

Using the above sequential and parallel relationship specification, multimedia presentation scenarios can be described. In TFG, the notation **N** is used to denote all the intervals contained in a scenario. The temporal relationship of an object interval N_x with other intervals is represented by the interval vector n_x . The interval vector n_x is defined as $n_x = (S_r, O_{id}, F_s, F_e)$ where S_r denotes the *source* (or the owner) of the interval, O_{id} the object identifier, F_s and F_e the presentation semantics chosen for the start and end of the display of the object. An interval vector can represent an object presentation or an intermission. The intermission vector, represented by n_τ , describes a presentation interval that is not mapped to any multimedia object. Intermissions are used to specify *gaps* in the multimedia presentation.

For describing multimedia presentations, the model of interval vectors and the involved temporal information are maintained in a Flow Graph (TFG). In TFG, intervals are described by nodes. A TFG is defined by the tuple $TFG = \{\Delta_N, N_t, E_d\}$ where Δ_N is a set of nodes for the interval vectors, N_t is a set of transit nodes and E_d is a set of directed edges. The model Δ_x of an interval vector n_x is composed of an interval node N_x , representing the interval of n_x , and δ node(s), representing its parallel relations to other intervals. N_x may associate none, one or two δ nodes. An intermission node has no δ nodes. The sequential specifications $\{A\}B$ and $< A > B$ are represented by the transit nodes in N_t , the *sq-node* and the *tri-node* respectively.

4.2 Synchronization Specification Using TFG: An Example

The TFG specifications for the relations *A ends B* and *A overlaps B* are shown in Figure 11(i) and (ii). The square nodes N_s and N_e are the *sq-nodes* representing the sequential relation $\{A\}B$. Figure 11(ii) has intermission node N_τ to describe the time lag between the event A and B. The δ nodes are used in Figure 11(ii) to signify the end of the concurrent presentation. The TFG model for the orchestrated presentation example (discussed in Figure 5) is shown in Figure 11.

Summary : The TFG model for multimedia synchronization can handle relative and imprecise temporal requirements in teleorchestra applications. The advantage in the TFG model is that no accurate temporal information, the duration value or the occurring points, is required. The TFG can represent all the possible temporal relations involved in a multimedia presentation scenario. Comparing the TFG model with the earlier Petri nets based ones, the Petri nets based models rely on the values for the duration of presentation in formulating the temporal specification. Hence, the Petri nets based models cannot represent the relative synchronization requirements. However, the TFG model does not address the issue of dynamic user participation during a multimedia presentation.

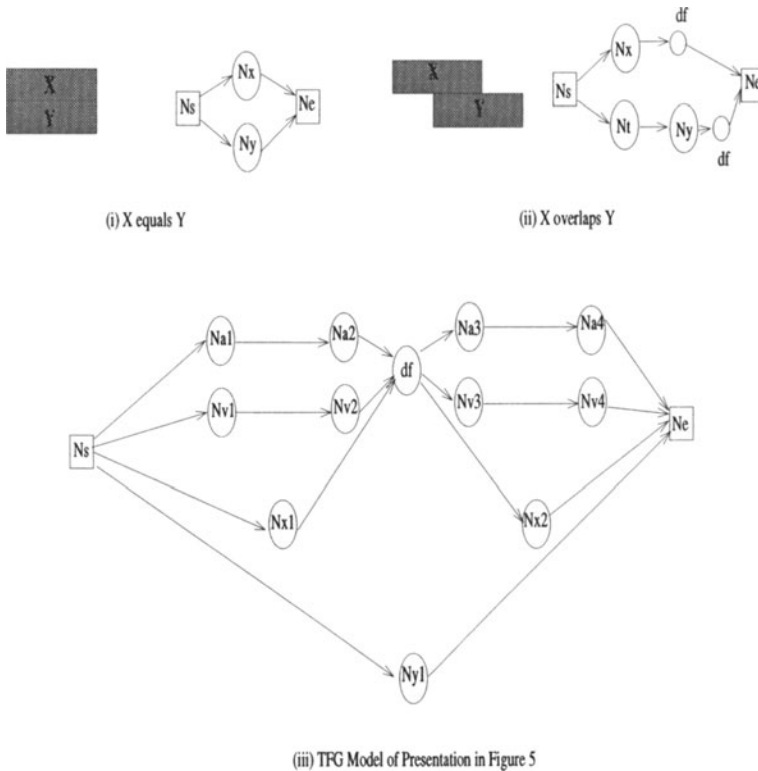


Figure 11 TFG synchronization models.

5 CONTENT-BASED INTER-MEDIA SYNCHRONIZATION

Content-based inter-media synchronization is based on the semantic structure of the multimedia data objects [21]. Here, a media stream is viewed as hierarchical composition of smaller media objects which are logically structured based on their contents. The temporal relationships are then established among the logical units of the media objects. The logical units of a media stream are the

semantic events that are either manually identified *a priori* or automatically detected by analysis of the media contents.

Traditional approaches for synchronization discussed in the earlier sections consider each media stream as being composed of set of objects to be presented at specified time intervals. Here, the synchronization specifications are given for describing concurrent relationships among the presentation of objects belonging to different media streams. Following these approaches, a segment of a media stream cannot be manipulated without re-establishing the temporal relationships among all other related media streams. Content-based synchronization, however, allows such manipulations to be done more easily. As an example, an audio stream can be dubbed onto a video stream in movie editing very easily following content-based synchronization approach.

In this section, we discuss the content-based synchronization approach discussed in [21].

5.1 Hierarchical Classification of Multimedia Objects

The top-most member in the hierarchy is an individual media stream that is considered as a *composite-object*. Let us consider a multimedia lecture on the topic *Distributed Multimedia Systems*. The video source of the lecture is a composite-object. A composite-object consists of *segment* objects. A segment is defined as an episode. The lecture presentation on Distributed Multimedia Systems can consist of the following segments : Media Characteristics (compression, coding, etc.), File System requirements, Device driver requirements, Network requirements and Distributed multimedia applications. A segment object in turn is composed of *event* objects. An event object is identified by theme changes in a segment object. As an example, the segment on network requirements can be composed of Quality of Service (QoS) requirements, network protocol features and network access methods. An event is composed of a set of *shot* objects. A shot is defined as a sequence of video frames that contains pictures taken under the same angle of a camera. The hierarchical classification of the lecture presentation on Distributed Multimedia System is illustrated in the Figure 12. Each node in this hierarchical structure is an abstraction of component objects where information such as component object IDs, temporal relationships among them, etc. is stored.

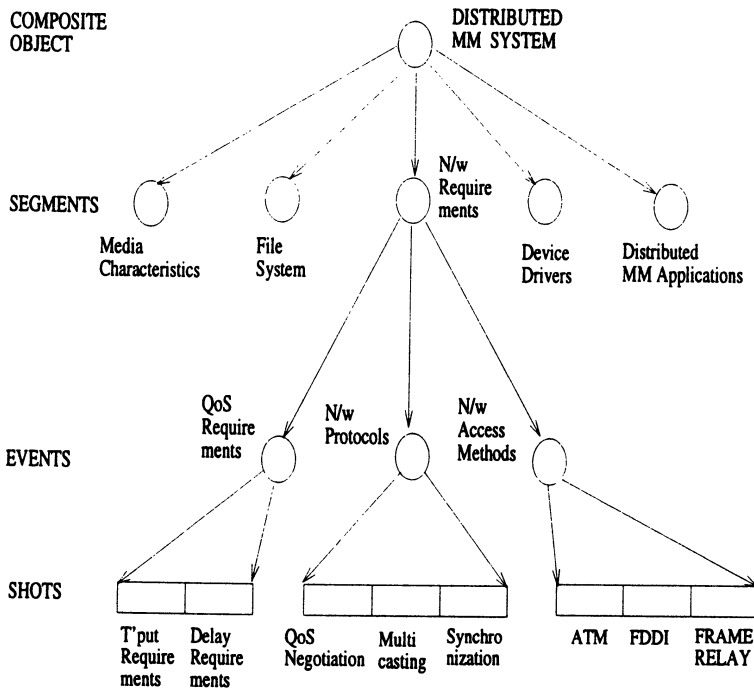


Figure 12 Hierarchical classification of objects.

5.2 Temporal Relations

The temporal relationship among media objects in content-based approach is specified with respect to their component media objects only. The temporal relationships among component objects is defined by the term *synchronization schedule*. A synchronization schedule statement consists of *Node IDs* and *operators*. *Node ID* describes the object to be presented. An *operator* is defined as an infix symbol that dictates the temporal relation between two objects. A *sequential* operator ($;$) and a *simultaneous* operator ($||$) are used for describing the synchronization relationships. Considering two nodes A and B, $A;B$ implies that the activity B starts at the end of the activity A and $A || B$ implies that the two activities go on parallelly. Using a combination of these two operators, the thirteen different possible temporal relations between media objects can be described.

The synchronization schedule can be created by the following algorithm given in [21]. This algorithm works on the hierarchical structure of the multimedia object.

1. **Do** depth first search on each subtree whose root node is a direct child of the ROOT node and create an operation schedule for each.
2. Choose a subtree **A** whose *begin[nodeID] = self*.
3. Choose another subtree **B**.
4. **If** (B's *begin[nodeID] = self*) **then**
 { add B's first object ID to A's first object ID with a || operator, B's second to A's second with a || operator, and so on until whichever runs out of its IDs first.

if A runs out of it first **then** just add B's each object ID to the end of A's schedule with a ; operator. }

else (B's *begin[nodeID] ≠ self*)

{ identify the segment that contains the object ID which is given for the value of B's *begin[nodeID]*

if it is identified **then** from the identified object ID do the same adding operation as the first part of the step 4 (**If** clause). i.e., the synchronization schedule is created at this point.}

else (it fails to be identified - this means that a user tries to bring a media object from the outside of the current object hierarchy and to compose with the current objects) { error.}

5. Repeat the steps 3, 4, 5 until all the segments are chosen and operated.

We can use the above algorithm to form the synchronization schedule for the example, lecture on Distributed Multimedia Systems, shown in Figure 12. For the segment on network requirements of distributed multimedia systems, the synchronization schedule will be :

throughput requirements; delay requirements; QoS negotiation; multicasting; synchronization; ATM; FDDI; Frame Relay.

Summary : The content-based synchronization model suits well for content-based multimedia information retrieval. The model has several advantages. It can be used to artificially *modify* or *create* synchronization between the involved

media. For example, the asynchrony in video and audio due to the differences in speeds of light and sound can be modified using the content-based synchronization scheme. In a similar manner, the scheme can be used to create a new synchronization specification between video and a dubbed audio track during movie editing.

6 MULTIMEDIA SYNCHRONIZATION AND DATABASE ASPECTS

The approaches discussed so far describe effective ways of modeling the temporal requirements of an orchestrated multimedia application. The multimedia objects have to be logically structured in a multimedia database and the structure should reflect the synchronization characteristics of the multimedia presentation. Also, multimedia data has to be delivered from a storage medium based on a predefined retrieval schedule. The size and the real-time characteristics of the multimedia objects necessitate different storage architectures. In this section, we describe the database schema representation and physical storage representation of multimedia objects with respect to the synchronization characteristics.

6.1 Database Schema Representation

Modeling of the synchronization characteristics of a multimedia presentation using the OCPN or the DTPN or the TFG approach provides a convenient visual technique for capturing the temporal relationships. However, a conceptual database schema is needed which preserves the semantics of the synchronization representation model to facilitate reproduction, communication, and storage of multimedia presentation. Synchronization models basically identify and group the temporally related multimedia objects of increasing complexity.

In [12, 8], a hierarchical synchronization schema has been proposed for multimedia database representation. Two types of nodes - terminal and nonterminal nodes - have been defined in this approach. The terminal nodes in this model indicate base multimedia objects (audio, image, text, etc.) and the nodes points to the location of the data for presentation. The nonterminal nodes have additional attributes defined for facilitating database access. The attributes include timing information and node types (sequential and parallel), allowing the assembly of multimedia objects during the presentation. The timing information

in the nonterminal node includes a time reference, playout time units, temporal relationships and required time offsets for specific multimedia objects. Figure 13 shows the hierarchical database schema for the multimedia presentation example described in Figure 5.

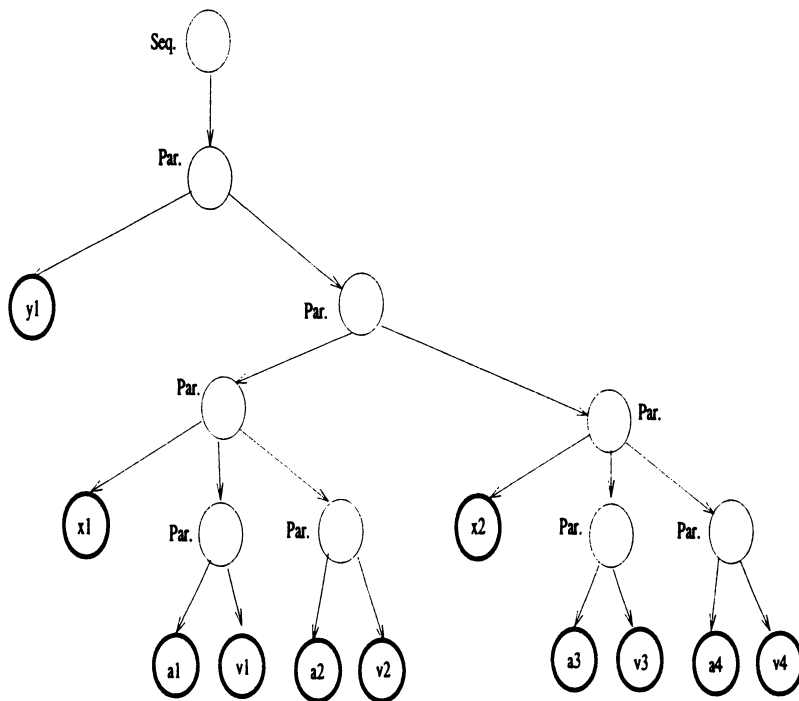


Figure 13 Hierarchical database schema for Figure 6.5.

6.2 Physical Storage Requirements

The physical medium used for storing multimedia objects should be able to meet the synchronization requirements of a multimedia application, both in terms of the storage capacity and bounds on retrieval time. Multimedia objects such as audio and video require very large amounts of storage space. For example, a data rate of about 2 Mbytes/second is required for HDTV video. Apart from the storage technology requirements, storage architectures adopted by the operating system also become important. The reason is that data must

be retrieved at a very high rate for HDTV video objects, e.g., 2Mbytes/second, from the disks.

Hence, the file system organization has to be modified for handling digital video and audio files. The aim is to handle multiple huge files as well as simultaneous access to different files, given the real-time constraint of data rates upto 2 Mbytes/second. This problem has been studied in detail in [18]. Most of the existing storage architectures allow unconstrained allocation of blocks on disks. Since there is no constraint on the separation between disk blocks storing a chunk of digital video or audio, bounds on access and latency times cannot be guaranteed. Contiguous allocation of blocks can guarantee continuous access, but has the familiar disadvantage of fragmentation of useful disk space. *Constrained block allocation* can help in guaranteeing bounds on access times without encountering the above problems. For constrained allocation, factors like size of the blocks (granularity) and separation between successive blocks (scattering parameter) have to be determined for ensuring guaranteed bounds [18]. The retrieval schedule of data for multimedia objects can be affected when the system becomes busy with some other tasks. The allocation or the data placement schemes also have to take into consideration the factor of contention with other processes in the system.

Summary : Multimedia objects composing orchestrated presentations need to be stored in database(s). The logical storage structure or the schema should reflect the synchronization characteristics of the orchestrated presentation. Also, the storage of the multimedia objects should satisfy the synchronization requirements of the orchestrated presentation in terms of the bounds on object retrieval time.

7 MULTIMEDIA SYNCHRONIZATION AND OBJECT RETRIEVAL SCHEDULES

Orchestrated multimedia presentations might be carried out over a computer network thereby rendering the application distributed. In such distributed presentations, the required multimedia objects have to be retrieved from the server(s) and transferred over the computer network to the client. The communication network can introduce delays in transferring the required multimedia objects. Other conditions such as congestion of a database server at a given

time and locking of the data objects by some other application also have to be considered. Retrieval of the multimedia objects have to be made keeping in mind these delays that might be introduced during the presentation. A retrieval scheduling algorithm has to be designed based on the synchronization characteristics of the orchestrated presentation incorporating features for allowing the delays that might be encountered during the presentation.

In [12, 15], a multimedia object retrieval scheduling algorithm has been presented based on the synchronization characteristics represented by the OCPN. Characterizing the properties of the multimedia objects and the communication channel, the total end-to-end delay for a packet can consist of the following components.

- Propagation delay, D_p
- Transfer delay, D_t , proportional to the packet size
- Variable delay, D_v , a function of the end-to-end network traffic.

The multimedia objects can be very large and hence can consist of many packets. If an object consists of r packets, the end-to-end delay for the object is :

$$D_e = D_p + rD_t + \sum_{j=1}^r D_{v_j}$$

Control time T_i is defined as the skew between putting an object i onto the communication channel and playing it out. Considering the end-to-end delay, D_e , the control time T_i should be greater than D_e . The various timing parameters, the playout time (π), the control time (T) and the *retrieval time* (ϕ) are as shown in Figure 14. The retrieval time for an object i (ϕ_i) or the object production time at the server is defined as $\phi_i = \pi_i - T_i$.

The above retrieval schedule is for a single object in a media stream. When multiple objects are retrieved, the timing interaction of the objects have to be considered. An optimal schedule for multiple objects is defined as one which minimizes their control times [12, 15]. The constraints associated with the determination of the minimum control time for a set of objects are as follows.

- An object cannot be played out before arrival, i.e., $\pi_i \geq \phi_i + T_i$.
- The minimum retrieval time between successive objects, i.e., $\phi_{i-1} \leq \phi_i - T_{i-1} + D_p$.

Following these constraints, an optimal retrieval schedule can be worked out. An optimal retrieval time for the final, m th object, is $\phi_m = \pi_m - T_m$. The

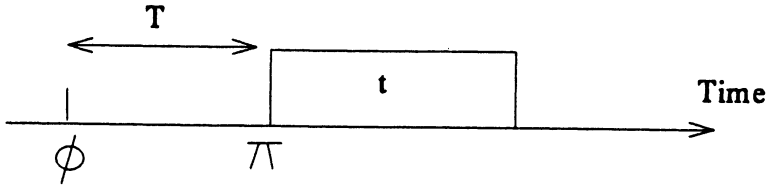


Figure 14 Delay factors in object retrieval.

retrieval times for the other objects can be worked out backwards based on the schedule ϕ_m for the final object. The scheduling algorithm given in [12, 15] is summarized as follows.

```

 $\phi_m = \pi_m - T_m$ 
for i = to m-2
if ( $\phi_{m-i} < \pi_{m-i-1} - D_p$ )  $\phi_{m-i-1} = \phi_{m-i} - T_{i-1} + D_p$ 
else  $\phi_{m-i-1} = \pi_{m-i-1} - T_{m-i-1}$ 
end.
```

Retrieval Schedule Example : Consider the orchestrated presentation shown in Figure 5. We can apply the retrieval schedule for the image stream I_1 with following assumptions. Let the channel capacity, C , be 10 Mbps. Let the size of the image objects (assuming color image with $1024 * 1024$ pixels and 24 bits/pixel of color information) be 24 Mbits. Let D_p , the propagation delay, and D_v , the variable delay, be 100 milli-seconds each. Then the transfer delay D_t is 2.4 seconds. Then, the retrieval schedule for the I_1 stream looks as follows.

Schedule = {-2.6, 0}

This implies that the first object x_1 is to be retrieved 2.6 seconds before the start of the presentation.

Summary : In this section, we discussed how an optimal retrieval schedule can be designed based on the synchronization characteristics of multimedia objects. The synchronization characteristics, as defined by the OCPN or the DTPN, specifies the object playout time instants π . The optimal retrieval

scheduling algorithm works on the fact that these playout time instances are to be satisfied given the end-to-end delay D_e involved in the transfer of multimedia objects. In the TFG model, the playout instances (π) are for the multimedia objects are described in a *fuzzy* or relative manner. This fuzzy temporal relations have to be converted into absolute schedules for retrieving the objects from a server. In [20], retrieval scheduling algorithms have been designed for the TFG specification of the synchronization characteristics.

8 MULTIMEDIA SYNCHRONIZATION AND COMMUNICATION REQUIREMENTS

In a distributed orchestrated multimedia presentation, the communication network provides support for transfer of data in real-time. The synchronization characteristics of an orchestrated presentation define a *predictable* sequence of events that happen in the time domain. This sequence of events can be used by the network service provider to understand the behavior of the orchestrated application in terms of the Quality of Service (QoS) requirements and the network load that might be generated [17, 22, 23]. The QoS parameters are characterized by a set of parameters such as the end-to-end throughput, the end-to-end delay and packet loss probabilities. The offered network load is characterized by the traffic that might be generated by the application.

8.1 QoS Requirements of an Orchestrated Presentation

In a distributed orchestrated presentation, it is essential that the object is available at the client before the presentation. In general, the client can adopt two different strategies for buffering the multimedia objects : minimum buffering strategy (B1) and maximum buffering strategy (B2). The B1 strategy tries to minimize the buffer requirements by buffering atmost one object or a frame, as the case may be, before its presentation. In the B2 strategy, the approach is to buffer upto a certain limit (say B_{max}) for each media stream on the client side, before the presentation. This implies that more than one object in a media stream will be buffered by the client during the presentation sequence.

The B1 strategy minimizes the buffer space requirements of the client and the QoS derived based on this strategy gives the *preferred* values for the client. The QoS requirements derived, based on the B2 strategy, specifies the *acceptable* values for the client. In a distributed orchestrated presentation, multimedia objects composing the presentation are retrieved from the server. The retrieval of objects do not impose any real-time demands from the network service provider. Hence, for orchestrated presentations, it is sufficient if the network service provider guarantees the required throughput.

Computation of Preferred Throughput :

For objects such as still images, we can determine the average throughput required by the application with the B1 buffering strategy, assuming that atmost one object in each multimedia stream is buffered by the application on the client system before the presentation. For objects such as video that consists of a set of frames to be presented at regular intervals, the application following the B1 strategy can buffer atmost one frame of the video object. Let us consider a multimedia stream with an object O_i at a playout time instant t_i and let the object size be Z_{O_i} . Since only one object is assumed to be buffered before the playout time instant for the stream, the retrieval of objects can be started only after the immediately preceding playout time instant t_{i-1} and has to be completed before t_i . Hence, the preferred throughput C_{app}^{pref} is [23] :

$$C_{app}^{pref}[t_{i-1}, t_i] \geq \frac{Z_{O_i}}{t_i - t_{i-1}}$$

Computation of Acceptable Throughput :

For calculating the *acceptable* QoS parameters, we should find out the minimum values that are required by the application. A minimum value of throughput (C_{app}^{acc}) implies that the application will be following the B2 buffering strategy of buffering more than one object (in the stream for which throughput is being calculated) before its presentation. When more than one object in a stream is buffered by the application, the start of the presentation is delayed by the time required for retrieving B_{max} bits using the acceptable throughput of C_{app}^{acc} . If B_{max} is the maximum buffer space available and $\sum_{i=1}^n Z_{O_i}$ is the total size of all objects to be retrieved, then the acceptable throughput is [23] :

$$C_{app}^{acc} \geq \frac{(\sum_{i=1}^n Z_{O_i} - B_{max})}{t_{final} - t_{initial}}$$

QoS Computation - An Example

Let us consider the orchestrated presentation example discussed in Figure 5. In the video stream V in Figure 5, we can assume an uniform frame interval with 10 frames/second and a frame size of 2 Mbits for the objects $v_1, v_2, \dots, v_4$. Let us assume a uniform time interval of 10 seconds for presentation of the video objects ($v_1, v_2, \dots$). The preferred QoS requirement of the application is 20 Mbps. For calculating the acceptable QoS, let us assume the maximum buffer space (B_{max}) available as 200 Mbits. The acceptable QoS will then be 15 Mbps. The preferred and acceptable QoS is shown in Figure 15. The preferred QoS values are required from the start of the presentation (with sufficient time for retrieving the first object) till the end of presentation. The acceptable QoS values are required from a time Γ seconds before the start of the presentation for retrieving B_{max} bits. ($\Gamma = B_{max}/C_{app}^{acc}$). In this example, Γ is 13.33 seconds.

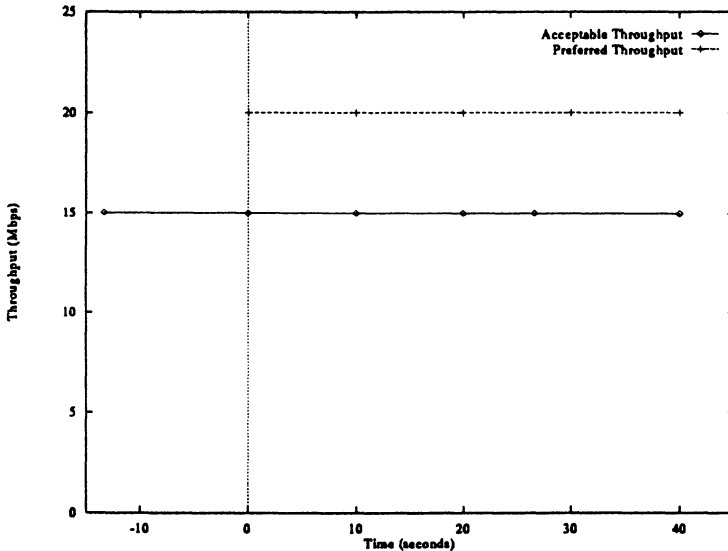


Figure 15 Throughput for stream V in Figure 6.5.

8.2 Traffic Source Modeling

The synchronization characteristics of an orchestrated presentation describes a sequence of objects with an associated duration of presentation, to be presented at different time instants. This sequence gives an implicit description of the traffic associated with the orchestrated presentation [23]. However, the traffic generated by an orchestrated presentation server also *depends on the object retrieval schedule* adopted by the client going through the presentation. This retrieval schedule basically determines the time instant(s) at which the client wants to receive the object(s). This schedule depends on the buffering that can be done at the client side. With the minimum buffering strategy (B_1), the delivery time of an object (or a frame, in case of video objects) is the *presentation time* of the previous object. As an example, object O_i will be delivered from the server at t_{i-1} just after the presentation time instant t_{i-1} of the object O_{i-1} . With maximum buffering strategy B_2 , the delivery schedule at the server is such that B_{max} bits will be delivered from the server every Γ seconds, where $\Gamma = B_{max}/C_{app}^{acc}$. C_{app}^{acc} is the minimum acceptable throughput that should be offered by the network for a multimedia stream with the B_2 buffering strategy.

Traffic Source Model Example : Consider the orchestrated presentation example shown in Figure 5. The assumptions made in Section 8.1 for computing the QoS requirements of the video stream V can be used for deriving the stream's traffic characteristics. In addition, let the network packet size, p , be 100 bits. With the B_1 strategy, the minimum throughput required from the network (C_{app}^{pref}) will be 20 Mbps. Figure 16 shows the traffic generated for the video stream using the minimum buffering strategy B_1 . Delivery of the first frame will start 0.1 seconds before the start of the presentation. Each frame is of size 2×10^6 bits and hence 2×10^4 packets ($framesize/p$) will be generated every 0.1 seconds. For the B_2 strategy, let us assume a maximum buffer space (B_{max}) of 200 Mbits for the video stream. The minimum throughput needed from the network (C_{app}^{acc}) will be 15 Mbps. With these assumptions, 2×10^6 packets (B_{max}/p) will be generated by the server every $\Gamma = B_{max}/C = 13.33seconds$. The initial chunk will be delivered 13.33 seconds before the presentation. Figure 17 shows the traffic generated for the video stream V in Figure 5 with maximum buffering strategy B_2 . It should be noted that the origin in the X-axis corresponds to the presentation start time.

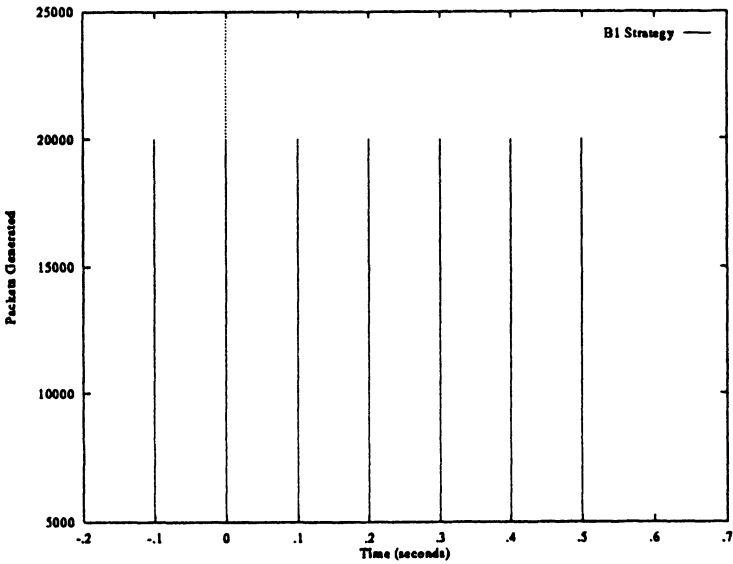


Figure 16 Traffic for stream V in Figure 6.5, with B1 strategy

9 SUMMARY AND CONCLUSION

Distributed orchestrated multimedia presentation applications such as multimedia lecture presentations, multimedia databases and on-demand multimedia servers are increasingly becoming popular. An orchestrated application has a synchronization characteristic that specifies the time instants at which the presentation of objects in the media streams must synchronize. This synchronization characteristic is specified in the form of pre-defined temporal relations and are stored along with the multimedia database. In this Chapter, we presented the problem of specifying the synchronization characteristics of an orchestrated multimedia presentation. We discussed the methodologies that have been proposed in the literature for describing the synchronization characteristics. The methodologies use concurrent programming language approach or graphical approach or content based approach. We then studied the database representation of the synchronization characteristics.

The synchronization characteristics of the presentation describe a *predictable* pattern of events that happen in the time domain. In a distributed orchestrated presentation, this pattern of events can be used to determine the object retrieval schedule that is to be adopted by a client. It can also be used by the

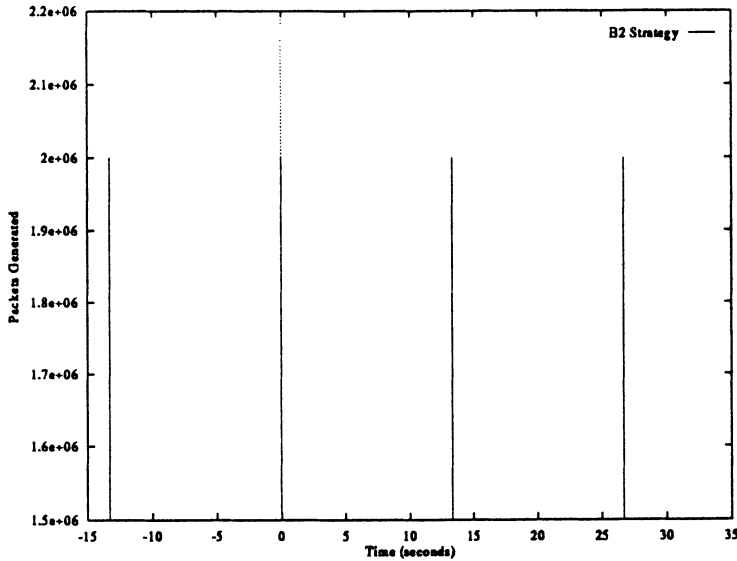


Figure 17 Traffic for stream *V* in Figure 6.5, with B2 strategy

network service provider to understand the QoS requirements of orchestrated applications and its network traffic generation characteristics. In [22, 23], an approach based on the Probabilistic Attributed Context Free Grammar (PACFG) has been proposed for deriving the QoS requirements and the network traffic generation characteristics of orchestrated multimedia presentation based on its synchronization characteristics. Based on these studies, protocols have been specified for multimedia synchronization [12] and for handling QoS negotiations in orchestrated presentation [23]. The traffic source model of an orchestrated presentation, based on its synchronization characteristics, can be used for simulation studies for assessment of network performance.

REFERENCES

- [1] J.L. Peterson, *Petri Net Theory and The Modeling of Systems*, Prentice-Hall Inc., 1981.
- [2] W. Reisig, *Petri Nets : An Introduction*, Springer-Verlag Publication, 1982.

- [3] K.S. Fu, *Syntactic Pattern Recognition and Applications*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1982.
- [4] J.E. Coolahan, Jr., and N. Roussopoulos, 'Timing requirements for Time-Driven Systems Using Augmented Petri Nets', *IEEE Trans. Software Eng.*, vol. SE-9, Sept. 1983, pp 603-616
- [5] J.F. Allen, 'Maintaining Knowledge about Temporal Intervals', *Communications of the ACM*, November 1983, vol. 26, no. 11, pp. 832-843.
- [6] W.M. Zuberek, 'M-Timed Petri nets, Priorities, Pre-Emptions and Performance Evaluation of Systems', *Advances in Petri nets 1985, Lecture Notes in Computer Science (LNCS 222)*, Springer-Verlag, 1985.
- [7] P.D. Stotts and R. Frutta, 'Petri-Net-Based Hypertext : Document Structure With Browsing Semantics', *ACM Trans. on Office Information Systems*, vol. 7, no. 1, Jan 1989, pp. 3- 29.
- [8] T.D.C. Little and A Ghafoor, 'Synchronization and Storage Models for Multimedia Objects', *IEEE Journal on Selected Areas of Communication*, vol. 8, no. 3, April 1990, pp. 413-427.
- [9] R. Steinmetz, 'Synchronization Properties in Multimedia Systems', *IEEE J. on Selected Areas of Communication*, vol. 8, no. 3, April 1990, pp. 401-412.
- [10] P.D. Stotts and R. Frutta, 'Temporal Hyperprogramming', *Journal of Visual Languages and Computing*, Sept. 1990, pp. 237-253.
- [11] S. Gibbs, 'Composite Multimedia and Active Objects', *Proc. OOPSLA '91*, pp. 97-112.
- [12] T.D.C. Little, *Synchronization For Distributed Multimedia Database Systems*, PhD Dissertation, Syracuse University, August 1991.
- [13] T.D.C. Little, A. Ghafoor, C.Y.R. Yen, C.S. Chen and P.B. Berra, 'Multimedia Synchronization', *IEEE Data Engineering Bulletin*, vol. 14, no. 3, September 1991, pp. 26-35.
- [14] J. Stefani, L. Hazard and F. Horn, 'Computational model for distributed multimedia applications based on a synchronous programming language', *Computer Communication, Butterworth- Hienmann Ltd.*, vol. 15, no. 2, March 1992, pp.114-128.
- [15] T.D.C. Little and A. Ghafoor, 'Scheduling of Bandwidth-Constrained MultiMedia Traffic', *Computer Communication, Butterworth-Heinemann*, July/August 1992, pp. 381-388.

- [16] B. Prabhakaran and S.V. Raghavan, '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.
- [17] S.V. Raghavan, B. Prabhakaran and S.K. Tripathi, 'Quality of Service Negotiation For Orchestrated MultiMedia Presentation', *Proceedings of High Performance Networking Conference HPN 94*, Grenoble, France, June 1994, pp.217-238. Also available as Technical Report CS-TR-3167, UMIACS-TR-93-113, University of Maryland, College Park, USA, October 1993.
- [18] H.M. Vin and P. Venkat Rangan, 'Designing a Multi-User HDTV Storage Server', *IEEE Journal on Selected Areas on Communication*, January 1993.
- [19] L. Li, A. Karmouch and N.D. Georganas, 'Multimedia Teleorchestra With Independent Sources : Part 1 - Temporal Modeling of Collaborative Multimedia Scenarios', *ACM/Springer-Verlag Journal of Multimedia Systems*, vol. 1, no. 4, February 1994, pp.143-153.
- [20] L. Li, A. Karmouch and N.D. Georganas, 'Multimedia Teleorchestra With Independent Sources : Part 2 - Synchronization Algorithms', *ACM/Springer-Verlag Journal of Multimedia Systems*, vol. 1, no. 4, February 1994, pp.153-165.
- [21] Dong-Yong Oh, Arun Katkare, Srihari Sampathkumar, P. Venkat Rangan and Ramesh Jain, 'Content-based Inter-media Synchronization', *Proc. of SPIE'95, High Speed Networking and Multimedia Systems II Conference*, San Jose, CA, February 1995.
- [22] S.V. Raghavan, B. Prabhakaran and S.K. Tripathi, 'Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation', to be published in *IEEE Journal on Selected Areas in Communication*, special issue on Multimedia Synchronization.
- [23] S.V. Raghavan, B. Prabhakaran and S.K. Tripathi, 'Handling QoS Negotiations In Orchestrated Multimedia Presentation', to be published in the *journal of High Speed Networking*.