Spring 2000

Universal access in digital libraries

Igg Adiwijaya
New Jersey Institute of Technology

ABSTRACT

UNIVERSAL ACCESS IN DIGITAL LIBRARIES

by
Igg Adiwijaya

Digital libraries are concerned with the creation and management of information sources, the movement of information across global networks and the effective use of this information by a wide range of users. A digital library is a vast collection of objects that are of multimedia nature, e.g., text, video, images, and audio. Users wishing to access the digital library objects may possess varying capabilities, preferences, domain expertise, and may use different information appliances. With the phenomenal growth of the Internet, the number of different information appliances will, if not already, increase substantially in the near future. Facilitating access to complex multimedia digital library objects that suits to the users' requirements is known as *universal access*.

The main objective of this thesis is to present our research work in the area of Universal Access within digital library environment. In this thesis, we will first present the current and future trend in information appliances, followed by discussion on the scope of our work. We propose an object manifestation approach in which digital library objects automatically manifest themselves to cater to the users' capabilities and characteristics. We provide a formal framework, based on Petri nets, to represent the various components of the digital library objects, their modality and fidelity and the playback synchronization relationships among them. We develop methodologies for object delivery without any deadtime under network delays. We have implemented a working system prototype to realize our approach.

UNIVERSAL ACCESS IN DIGITAL LIBRARIES

by
Igg Adiwijaya

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

May 2000

APPROVAL PAGE

UNIVERSAL ACCESS IN DIGITAL LIBRARIES

Igg Adiwijaya

Dr. Nabil Adam, Thesis Advisor — Date
Director, Center for Information Management, Integration and Connectivity
Professor of Computer & Information Systems, Rutgers University, Newark, NJ

Dr. Vijay Atluri, Co-advisor — Date
Assistant Professor MIS/IS Department, Rutgers University, Newark, NJ

Dr. James Geller — Date
Associate Professor, Computer Information Science Department,
New Jersey Institute of Technology, Newark, NJ

BIOGRAPHICAL SKETCH

Author: Igg Adiwijaya

Degree: Master of Science in Computer Science

Date: May 2000

Education:

- Doctor of Philosophy in Computer Information Technology
  Rutgers University, Newark, NJ, May 2000
- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, May 2000
- Master of Business Administration,
  Rutgers University, Newark, NJ, 1999
- Bachelor of Science in Computer Information Systems,
  Rutgers University, Newark, NJ, 1994
- Bachelor of Arts in Finance,
  Rutgers University, Newark, NJ, 1994
- IBM Certified DB2 Fundamental, December, 1997
  IBM Certified DB2 Application Developer, January, 1998
  IBM Certified Database Administrator, January, 1998
  An invited trainee, IBM Training Center, Boulder, Colorado, October, 1997

Major: Computer Science

Presentations and Publications:

N. Adam, I. Adiwijaya, T. Crithclow and R. Musick,
"Detecting Data and Schema Changes on Scientific Documents," accepted for publication IEEE ADL Conference 20000

N. Adam, V. Atluri, and I. Adiwijaya,
"System Integration in Digital Library," to appear in The Special Section of System Integration - Communication of the ACM, 2000

Presentations and Publications (continued):

N. Adam, V. Atluri, I. Adiwijaya and S. Banerjee,
"A Dynamic Manifestation Approach for Providing Universal Access to Digital Library Objects," accepted for publication in IEEE Transaction on Multimedia, 2000

N. Adam, I. Adiwijaya, V. Atluri, and Y. Yesha,
"EDI through a Distributed Information Systems Approach," 31st Hawaii Conference on System Sciences, Hawaii, January, 1998

N. Adam, I. Adiwijaya, and Y. Chung,
"The Implementation of a DW System in a Heterogeneous Environment,"International Conference on System Integration Technology and Applications, Seoul, South Korea, December, 1997

To my parents who have endured so much.

ACKNOWLEDGEMENT

First I would like to thank God for His blessing that has allowed me to complete this thesis. I would also like to thank my parents whom I am forever in debt for their support and prayers. Finally, I must thank Dr. Nabil Adam, Dr. Vijay Atluri, and Dr. James Geller for their guidance, support, and comments toward this thesis. Without their help and understanding, this thesis would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Creation and gathering of electronic information is increasing day by day. This trend can be seen among many organizations, from businesses like supermarkets that gather simple data such as information of all their customers, to hospitals that maintain complex data such as clinical information of their patients in electronic form. Recent trend is also towards digitizing paper-based information and preserving it in electronic form. Such large collections of information, known as digital libraries, is being created by many people and data gathering instruments in many forms and formats, stored in many repositories around the world, and becoming increasingly interconnected via electronic networks [5, 4, 3]. Thus digital libraries can be thought of as a large collection of distributed, heterogeneous information managed by autonomous sources.

On one hand the objects stored in digital libraries are of multiple media including text, video, audio, image, etc. On the other hand, users wishing to access these objects possess varying capabilities and characteristics. For example, users may belong to different disciplines, may speak different languages, may possess different technical expertise, and may use different appliances (e.g., PC, PDA, and TV) for the purpose of retrieval. To facilitate access to the desired data of multiple media according to the varying capabilities and characteristics of users is known as *universal access*. Al- though the ultimate objective of universal access, is to cross these barriers and enhance communication across disciplines, languages, and cultures, in this thesis we limit our focus to providing universal access to multimedia data to users with varying technical

capabilities using different information appliances. We also take the mobility of users and their preferences into account when providing universal access. Our approach in this thesis provides methodologies to develop digital library objects that have built-in intelligence that enable them to automatically manifest themselves in a way that caters to user's capabilities and characteristics.

For the purpose of motivating the need for universal access, we present, below, the following example.

Example 1 Consider a case of physician who is affiliated with a hospital and has also a private practice and whose typical daily schedule is as follows. During the morning hours she sees her patients at the hospital, then drives car to her private practice office, where she stays for a few hours, and then goes home in the evening. While she is at her office in the hospital, she has access to a powerful workstation that is capable of handing all the different modalities of an object. Her private practice office, on the other hand, is equipped with a PC that cannot display high-resolution image and her car has a cellular phone that can only handle audio. Patients' information is in the form of a digital library object, which for example, may comprise of an image with an audio as well as text explaining the symptoms, X-ray showing various levels of the condition, a treatment to this condition with a video demonstrating the surgical procedure implanting braces, and an X-ray with the braces (see Figure 2.2).

While seeing patients at her hospital office or her private office, she prefers to sup- press the audio portion of the object and concentrate more on image, video and text components. While driving, she may want to continue accessing objects through the

cellular phone thus, suppressing all portions of the object but the audio. In case she had to leave her office at the hospital while she was in the middle of reviewing a specific object, she would like to resume access to this object from her cellular phone from where she left off. Clearly, there exist manifestation constraints based on her preferences.

Having digital library objects that have built-in intelligence that enable them to automatically manifest themselves in a way that caters to user's capabilities and characteristics gives raise to several challenges, including the following. First, authoring of digital library objects itself has to be done in such a way that it enables automatic manifestation. Second, the ability to detect user's capabilities, preferences and his/her information appliance capabilities then render the object accordingly. Finally, there should be a mechanism for saving the state on some server in the network. Thus, when the user wishes to resume, the new manifestation starts from the previous state information. Thus, the user sees a continuation from the previous rendition of the object.

1.1 Trend in Internet Appliances

To further motivate the need for universal access, we present current and future trend in information appliances and some statistical figures in this section.

As a result of the tremendous growth of the Internet, the development and use of new types of appliances, capable of accessing information via the Internet, are expected to increase very rapidly. One testimonial example is the projected sale of Diamond Multimedia Rio Player, a portable MP3 player much like Sony Walkman for Music CDs, that will reach over 750,000 units in its introductory year of 1999/2000 [17]. [13] presents a nice discussion of future use of information appliances for the future houses,

where different types of appliances, capable of accessing the Internet and communicating among each others, are used in the kitchens, living-rooms, home offices, bathrooms and family vehicles. In a kitchen, a coffee maker could checked your daily schedule via the Internet and start brewing at 6:00 am, while the refrigerator could notify you via email that the milk has expired and request that you buy a new one. The dishwasher could identify that you have bought a new detergent and adjust the washing cycle for the new detergent. In a living room, one can use a single remote control to surf the Web on TV, play MP3 on audio server, and download videogames for later use in the game player. Upon playing video, downloaded from the Internet, on the TV, the lamp at the living room would automatically deem once the video starts playing. One would watch the headline news via the mirror over the sink in the family bathroom while brushing his/her teeth in the morning. Many information appliances would be used in numerous aspects of our daily live.

[17] coins Internet Appliances to be non PC-based devices that are capable of delivering contents, services, and applications of the Internet via those devices. Current types and devices of Internet Appliances are shown in the following table (adopted from [17]).

Shipment of Internet appliances worldwide is projected to increase from 5.9 million units in 1998 to over 55.7 million units by 2002 [17]. This will be an estimated of a value growth from $2.2 billion in 1998 to more than $15.3 billion in 2002. In the rest of this thesis, we will refer to information appliances to include not only the traditional PC-based devices but also Internet appliances alike. In the following section, we present some of the research work in universal access.

Table 1.1: Representative Internet Appliances

Internet Appliance	Definition	Sample Products
NetTVs	Set-top boxes that Internet-enable a television or televisions with built-in Internet Connectivity	Microsoft WebTV
Internet Screenphones	Telephones with screen that integrate Internet access	InfoGear iPhone
Internet Gaming devices	Gaming consoles that integrate Internet access	Sega Dreamcast
Internet-smart handheld devices	Cellular phones, PDAs, and other portable devices that integrate Internet access	3Com Palm VII, AT&T PocketNet Phone, Nokia 9000,9000I,9110,7110
NC clients	Devices that provide Internet access, but also run downloaded software applications	IBM NetStation
PC-intermediated	Devices that access the Internet through PCs to download content	Diamon Multimedia Rio Player, Nuvomedia Rocket e-Book, Audible Mobile Player

1.2 Prior Work

Little prior work can be found in the area of universal access. Current approaches to object delivery work as follows [2]. (1) Traditionally, when an object is accessed, the entire object is sent to the user, disregarding the capabilities of the user's appliance. This unnecessary transmission of data results in network congestion and delays. (2) In a pre-compiled approach, several modalities of an object are generated a priori and stored. This approach has the advantage of sending only useful information. However, it requires that the author of the object to store every possible data format or type the user may want. Thus it is not practical in a DL environment. (3) In other cases, only one copy of the object of a single modality is stored and is then converted to user's requested modality

and fidelity on demand. This approach focuses on translation but does not deal with mixed modalities, whereas multimedia presentations rely on the interplay of different modalities.

Since research conducted in the area of delivery of multimedia data is relevant to this thesis, we review it below. Little and Ghafoor [15] have proposed a Petri net based model, called Object Component Petri Net (OCPN), to specify synchronization constraints among multimedia objects. In this thesis, we use a modified version of OCPN that can model the modality, fidelity of the components in a digital library object and the synchronization relationships among them. Later, Woo et al. [21] have extended the OCPN to XOCPN that can model requirements of rate-controlled transmission that schedules the transmission of objects at the source by taking into account network delays and channel capacity. The synchronization among the various multimedia objects is carried out by dividing each component into smaller units, called Synchronization Interval Units (SIUs) and assigning each SIU a synchronization interval number so that they can be played according to the specified order. On the other hand, in our approach, we need not divide the components into smaller units and assign the sequence numbers to them because the synchronization information as well as the piece of software that is responsible for preserving the synchronization is already available at the client site. Guan et al. [8] have also shown how synchronization among multimedia components can be achieved by proposing a Distributed Object Composition Petri Net (DOCPN) that assigns priorities to arcs. While all the above research focuses primarily on multimedia synchronization, it does not address the universal access problem. Recent work towards universal access by Chen et al. [7] proposes an approach that routes the delivery of object

via an intermediate transformation service station that is capable of transforming the object into the desired format of the client.

In this thesis, we provide an approach that enables self-manifestation of digital library objects based on user capabilities and preferences. Central to our methodology is the notion of oblet. Oblet is a small piece of software that installs itself on the client, examines the user and system profile of the client, determines the components and the order in which they must be played, retrieves the necessary object components, and then renders the object to the client accordingly. The objective of the oblet is to take over the responsibility of rendering the object from the server. Our intention is to minimize, as much as possible, the responsibilities of the server in order to avoid congestion and overloading at the server.

We first formalize the digital library object by identifying the various relationships and constraints among its components. These constraints are of two types: synchronization, fidelity and spatial constraints. Synchronization constraints specify the various temporal relationships that must be adhered to when playing the DL object. Fidelity constraints specify the capabilities of the client that are necessary for the object to be played with the specified quality. Spatial constraints specify where each DL component will be positioned on the user screen in relation to the other components. These constraints as well as the playback duration of each component is called the object plan. We then develop a Petri Net, called *multimedia object Petri Net* (MOPN) to represent the object plan that lends itself to easy analysis. The object manifestation is comprised of object plan modification, object delivery and object rendition. The object model, MOPN and object manifestation are presented in section 2. We then compute the

latest time at which each object component must be requested by the client in order for the object to be played without any deadtime under network delays (section 3). This utilizes the deterministic and statistical network delay guarantees. We have implemented our universal access system in Java, related details are presented in section 4.

CHAPTER 2

THE PROPOSED APPROACH

Our approach to universal access has three components: the digital library object model, the object manifestation model, and the object delivery model. Below is a discussion of each.

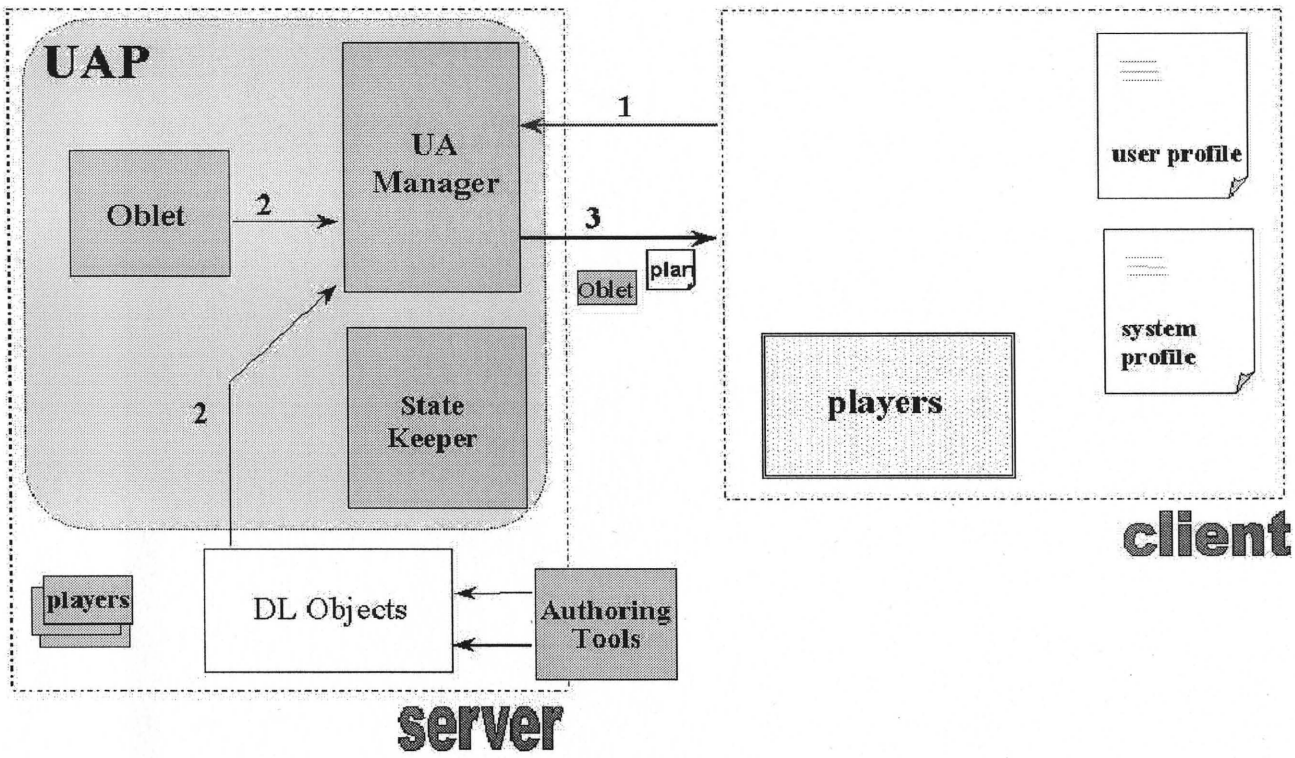

Figure 2.1: System Architecture

2.1 The Digital Library Object Model

In general, a digital library object, dlo, is of multimedia nature and is made up of a set of components, e.g., text, audio, image and video components. Formally, we define a dlo as: $dlo = (\langle c_1, \ldots, c_n \rangle S)$, where each c_i is a component of dlo and S is a set of synchronization and spatial constraints. Each component c, in turn, is a triple, $c = (ml, fd,$

pd), where ml represents the modality, fd represents the allowed fidelity range (also called the fidelity constraint), and pd represents the playback duration.

Examples of such objects include digital news objects and those objects found in medical libraries, e.g., NLM, and environmental libraries, just to name a few. For example, a DL object describing scoliosis in a medical digital library (shown in Figure 2.2) may be comprised of an image with an audio as well as text explaining the symptoms, X-ray showing various stages of the condition, a treatment to this condition with a video demonstrating the surgical procedure implanting braces, and an X-ray with the braces.

Here there exists, based on some pre-specified constraints, a timing relationship among the various multimedia components of the object. Example of such constraints in the medical library object may include:

- The X-ray must start immediately after the image has been displayed
- The text must be displayed simultaneously with the image and the X-ray

In addition to the timing relationship constraints, in a multimedia setting, the location where each component is displayed on the user screen is also of importance, known also as spatial constraints.

Besides the synchronization and spatial constraints, an object may have a set of fidelity constraints. Fidelity constraints indicate the permissible level of fidelity for each component. For example, an author of a medical object may pose a constraint that an X-ray can be played only on high-resolution video devices, since a low resolution device would not be able to render enough details. We represent fidelity constraints as a range,

for example $\geq$ 640x480 to specify that the resolution be at least 640x480 to view the object.

For users using a dumb terminal, the medical object can be rendered only in text; for users with audio devices, the object can be rendered only in audio. We may also have special manifestations for users with physical limitation, for example, closed captioned news for hearing-handicapped users and audio only for visually handicapped users.

These synchronization, fidelity and spatial constraints are assumed to be specified by the domain expert (the author of the object) in the form of a plan. An *object plan* is a set of the object components with the information as to the duration of each object component as well as its synchronization, fidelity and spatial constraints.

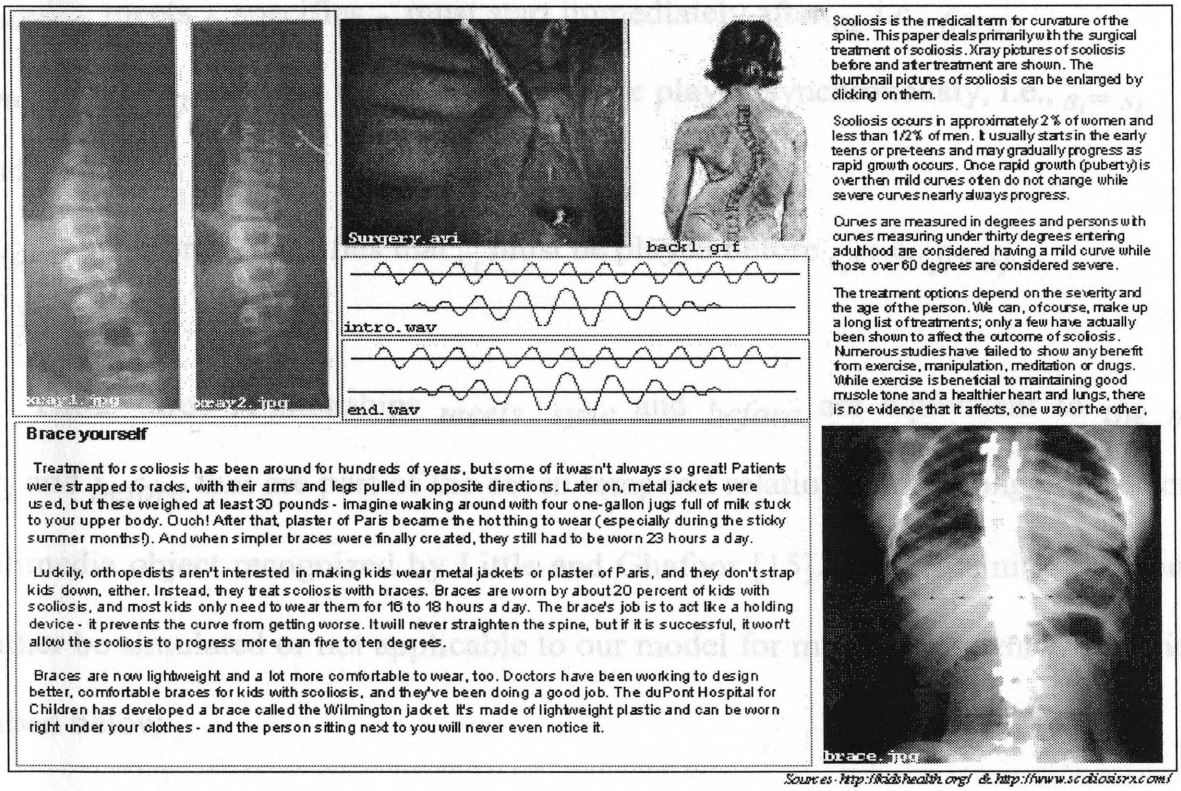

Figure 2.2: An Example of a DL object in a Medical Digital Library

Synchronization Constraints:

To model the various synchronization constraints among the *dlo* components, we use interval temporal logic. The philosophy of interval temporal logic was first introduced by Hamblin [10]. In [6], twelve possible relationships between two distinct intervals have been identified, including *during, before, later,* and *immediately after*. These twelve relationships are not independent of one another and it is sufficient to define six operators [18] as the other six are complements of the rest. Among these six, we identify the following three as necessary for capturing the synchronization constraints among the *dlo* components.

Let s_i and f_i indicate the playback start time and playback finish time of c_i, respectively.

1. *meets:* c_i meets c_j specifies c_j must start immediately after c_i, i.e., $f_i = s_j$.
2. *sync:* c_i sync c_j specifies that c_i and c_j must be played synchronously, i.e., $s_i = s_j$ and $f_i = f_j$.
3. *before:* c_i before c_j specifies that c_i must be played before c_j, i.e., $f_i < s_j$.

These three relationships *meets, sync* and *before* are equivalent to *the meets, equals* and *before* that are part of the seven temporal relationships among components of a multimedia object recognized by Little and Ghafoor [15]. The remaining relationships can either be simulated or not applicable to our model for multimedia synchronization, as described below.

c_i *starts* c_j: This specifies that c_i and c_j must start playing at the same instant. This can be realized by assuming the presence of another component c_k (possibly dummy component). Then, c_i starts c_j can be represented as c_k meets c_i and c_k meets c_j.

ci finishes c_j : This specifies that c_i and c_j must finish playing at the same instant. As above, this can be realized by assuming the presence of another component c_k. Then, c_i finishes c_j can be represented as c_i meets c_k and c_j meets c_k.

c_i *during* c_j : This specifies that c_i should be played during the playing of c_j. This constraint can be transformed into c_i finishes c_j (or c_j finishes c_i) because this will not effect the synchronization.

ci overlaps c_j : This specifies that the time intervals of c_i should overlap with c_j. A DL object component synchronization does not require such a constraint as it synchronizes neither the start nor the finish of c_i and c_j.

Fidelity Constraints:

Let *FV (ml)* denotes the set of all possible fidelity values for modality *ml*. We model fidelity constraints as a range of values in *FV (ml)*.

Formally, we define a fidelity constraint as follows. A set of fidelity constraints for a modality *ml*, *FD(ml)* is a set of fidelity ranges *fd* such that

- if op $\in \{=, <, >, \leq, \geq\}$ and $fv \in FV\ (ml)$, then $fd = op\ fv$;
- if fd_1 and fd_2 are two fidelity ranges, then $fd_1 \wedge fd_2$, $fd_1 \vee fd_2$ are fidelity ranges;
- if fd is a fidelity range, then (fd) is a fidelity range.

Spatial Constraints:

Given a DL object with a set of components, $\{c_1, c_2, \ldots, c_n\}$, we recognize three spatial constraints among these components based on object reference. They are: *right, above* and *front*. Specification of spatial constraints based on object reference has been presented earlier [19, 9, 14].

For each c_i, let x_i and y_i represent the X and Y coordinates of lower-left corner of the window where c_i must be displayed, w_i, and h_i its width and height. In addition, we also assume z_i, a positive integer associated with each c_i whose default value is 0 unless otherwise specified.

Given c_i and c_j, we say,

- c_i is to the right of c_j if $(x_j + w_j) - x_i \leq 0$,
- c_i is above c_j if $(y_j + h_j) - y_i \leq 0$, and
- c_i is to the front of c_j if $z_j - z_i < 0$.

While the first two constraints define the spatial relationships among components in two dimensional space the third constraint allows the specification of one component to be displayed on the foreground of another. However, the third constraint is applicable only if the two components overlap.

2.2 Petri Net Representation of the Object Plan

In this section, we present how Petri nets can be used for representing the synchronization and fidelity constraints among the multimedia components with different modalities of the DL object. There are a number of reasons for this choice. First, they are

able to model synchronization and timing relationships among different events. Second, Petri net provides visualization and enactment. Third, they lend themselves to easily model constraints representing policies, user and system profiles, and user preferences. Fourth, Petri nets are amenable to analysis; this allows for optimization of the presentation of the DL object. For example, optimization can be performed with respect to the cost of playing an object by associating a cost function to the components in the DL object, with respect to optimally retrieve an object from the server by considering the network traffic, as described in section 3. We call the Petri net representation of a DL object the object plan.

A *Petri Net (PN)* is a bipartite directed graph consisting of two types of nodes: *places* (represented by circles) and *transitions* (represented by bars). Arcs (edges) are either from a place to a transition or from a transition to a place.

A *marking* may be assigned to places. If a place p is marked with a value k, we say that p is marked with k tokens. Weights may be assigned to the edges of PN, however, in this thesis we use only the ordinary PN where weights of the arcs are always equal to 1.

A PN can be formally defined [16] as a 4-tuple, $PN = (P, T, F, M)$ *where*

$P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places and

$T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, where $P \cap T = \varnothing$, and $P \cup T \neq \varnothing$,

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs, *and*

$M_0 = P \rightarrow \{0, 1\}$ is the initial marking.

A transition (place) has a certain number (possibly zero) of input and output places (transitions). Given a PN, the input and output set of transitions for each place p_i

are denoted by $\bullet p_i = \{t_j | f(t_j, p_i) \in F\}$ and $p_i \bullet = \{t_j | f(p_i, t_j) \in F\}$, respectively. Similarly, the input and output set of places for each transition t_i are denoted by $\bullet t_i = \{p_j | f(p_j, t_i) \in F\}$, and $t_i \bullet = \{p_j | f(t_i, p_j) \in F\}$, respectively.

At any time a transition is either enabled or disabled. A transition t_i is enabled if each place in its input set t_i has at least one token. An enabled transition can fire. In order to simulate the dynamic behavior of a system, a marking in a PN is changed when a transition fires. Firing of t_i removes the token from each place in $\bullet t_i$, and deposits one token into each place in $t_i \bullet$. The consequence of firing a transition results in a change from original marking M to a new marking M'. For the sake of simplicity, we assume firing of a transition is an instantaneous event.

To model the duration as well as the synchronization and fidelity constraints and the modality of the object components, we use an extended PN model, *called Multimedia Object Petri Net* (MOPN). MOPN can be formally defined as follows:

A *Multimedia Object Petri Net* (MOPN) is a tuple $MOPN = (PN, D, MF)$ where

(i) $PN = (P, T, F, M)$ is an ordinary Petri net,

(ii) D is a duration function, $D : P \rightarrow \tau$ where $\tau = \{\alpha \in \Re$ where $\Re$ represents the set of all real numbers, $| \alpha \geq 0\}$,

(iii) $ML = \{ml_1, ml_2, \ldots, ml_n\}$ is a set of modalities and MOD is a modality function such that $MOD : P \rightarrow ML$, and

(iv) $FD = \{fd_1, fd_2, \ldots, fd_n\}$ is a set of fidelity ranges and FID is a fidelity function such that $FID : P \rightarrow FD$ such that $FID(p_i) = fd_i$ where $fd_i \in FD(MOD(p_i))$.

The above definition states that each place is assigned a modality and a duration, where the modality indicates the type of player to be used to play this component, and the

duration how long the component is to be played. For example, $MOD(p_i)$ = image indicates that the modality of place p_i is image, and $D(pi)$ = 5 indicates that the duration of p_i is 5 time units. Also, $FID(p_i)$ = " ≥ 640x480" where " ≥ 640x480" is an element in the set of fidelity ranges applicable to the modality of p_i which is image.

The firing rules for the $MOPN$ can be formally stated as follows.

Definition 1

Given a transition t_i

1. for any p_j marked with a token $m(p_j)$, $m(p_j)$ is said to be available if it remains in p_j for δ^m_j,
2. t_i is said to be enabled if $\forall p_j \in \bullet t_i\ m(p_j) = 1$ and $m(p_j)$ is available, and
3. an enabled transition may $fire$. Firing of t_i results in a new marking M' as follows:

$\forall p_j \in \bullet t_i,\ m'(p_j) = m(p_j) - 1;$

$\forall p_k \in t_i \bullet,\ m'(p_k) = m(p_k) + 1$

The first firing rule states that each token must reside in a place indicated by the delay associated with the place. After this time delay, the token is said to be available. The second firing rule states that a transition is enabled only if tokens in all its input places are available. The third rule states that when a transition fires, one token is removed from each of its input places and one token is added to each of its output places.

We represent the object plan of dlo as a $MOPN$ as follows: Each component $c_i \in dlo$ is represented as a place p_i in the MOPN. The attributes modality and fidelity of c_i are represented as a label (ml_i, fd_i) of p_i. This means that, when a place p_i is filled with a token with a label (ml_i, fd_i), the appliance corresponding to the modality ml_i will be

activated to play c_i if its fidelity is in the range specified by fd_i. We do not include the spatial constraints in the Petri net representation.

A place with a *Null* modality indicates an empty component or disabled component and therefore does not activate any appliance. A place with a *Null* fidelity indicates that there exists no fidelity constraint. The attribute pd_i indicating the duration of c_i is represented as a delay associated with the corresponding place p_i. Thus when a token arrives to p_i, the token is available only after pd_i, thereby ensuring that all the components that follow c_i start only after the finish time of c_i.

The *meets, sync* and the *before* synchronization constraints between c_i and c_j are represented as shown in Figure 2.3. The MOPN of the DL object in Figure 2.2 is shown in Figure 2.4.

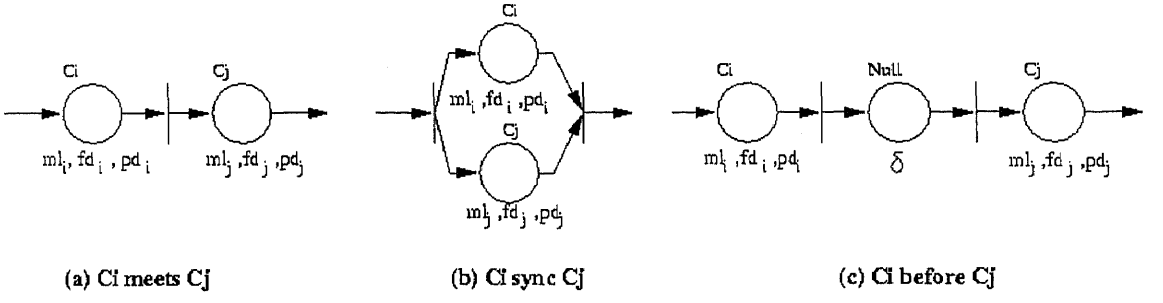

Figure 2.3: MOPN Representation of meets, sync, and before Synchronization Constraints

2.3 Object Manifestation

The first step in rendering a given digital library object is to be able to identify user's preferences and profiles as well as the capabilities of the information appliance. We assume that this information is stored on the user's client. At the receipt of a user's request at the server, an oblet is invoked which then installs itself on the user's machine

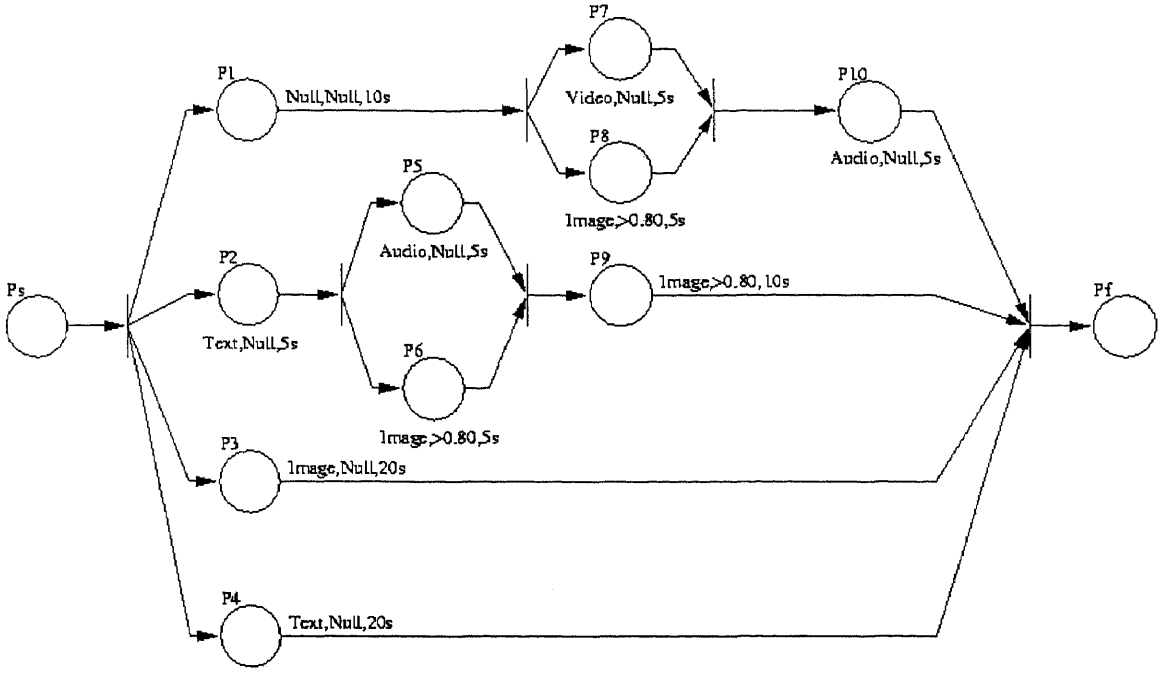

Figure 2.4: MOPN Representation of example 2.2

(client) and gains access to the file containing user's preferences and profile as well as the capabilities of the user's appliance. The notion of an oblet is an extension of the "install" programs concept available in most PC software installations that check the PC for certain resources (availability of disk space, version of operating systems, etc.) before installing a copy of the software.

Once the user's preferences and profiles as well as the capabilities of the information appliance have been identified, the following steps are executed.

(1) *Object Plan Modification.* The object plan, MOPN, is modified accordingly (based on algorithm 1, below), resulting in a modified plan, which is pruned by removing and/or merging disabled components.

(2) *Object Delivery.* The object components as specified in the modified plan are delivered to the client (based on algorithm 2, below).

(3) *Object Rendition*. The object is rendered according to the modified plan. This is accomplished simply by placing a token in the starting place (p_s) of the modified plan. Then each transition is fired according to the rules in definition 1. A fired transition places a token in each of its output places. Presence of a token invokes the corresponding player. The token in this place is removed after the specified duration because its output transition fires. At this point, the oblet sends a message to the state maintainer at the server that the component has been successfully rendered. When the token reaches the final place (p_f), the rendition is complete. At this point both the object and plan are erased from the client.

We present, below, both algorithms for object plan modification and object delivery respectively.

Algorithm 1 [Object Plan Modification]

Step 1:

for each $p_i \in P$

if ($MOD(p_i) \notin$ sys-prof-set **or** $MOD(p_i) \notin$ usr-prof-set) **and** $FID(p_i) \notin$ sys-prof-set **then**

$MOD(p_i) \leftarrow null$

$FID(p_i) \leftarrow null$

end if

end for

Step 2:

for each $p_i \in P$

if $MOD(p_i) = null$ **and** $\exists p_j$ *such that* $(p_i, p_j \in {}^\bullet t_k \wedge p_i, p_j \in t_l{}^\bullet)$ where $t_k, t_l \in T$, **then**

remove place p_i and edges $f(t_i, p_i), f(p_i, t_k)$

end if

end for

Step 3:

for each $t_i \in T$

if $card(\bullet t_i) = card(t_i \bullet) = 1$ **and** $MOD(p_j) = MOD(p_k) = null$

such that $p_j \in ot_i$ and $p_k \in t_i \bullet$, **then**

remove t_i, p_k, $f(p_j, t_i), f(ti, pk)$

remove every $f(p_k, t_l)$ such that $t_l \in pk\bullet$ and add $f(p_j, t_l)$

$D(p_j) \leftarrow D(p_j) + D(p_k)$

end if

end for

Step 4: repeat steps 3 and 4 until no changes to the modified plan occur

□

We explain below each step of the above algorithm by considering the DL object represented in Figure 2.4. Step 1 of the algorithm simply replaces the modality of each component of the object by *null* if that modality is either not supported by the client or not among the preferences of the user. If the client's appliance is capable of playing only audio and textual components and for user who would like to view only text and audio, the MOPN in Figure 2.4 will be modified as shown in Figure 2.5.

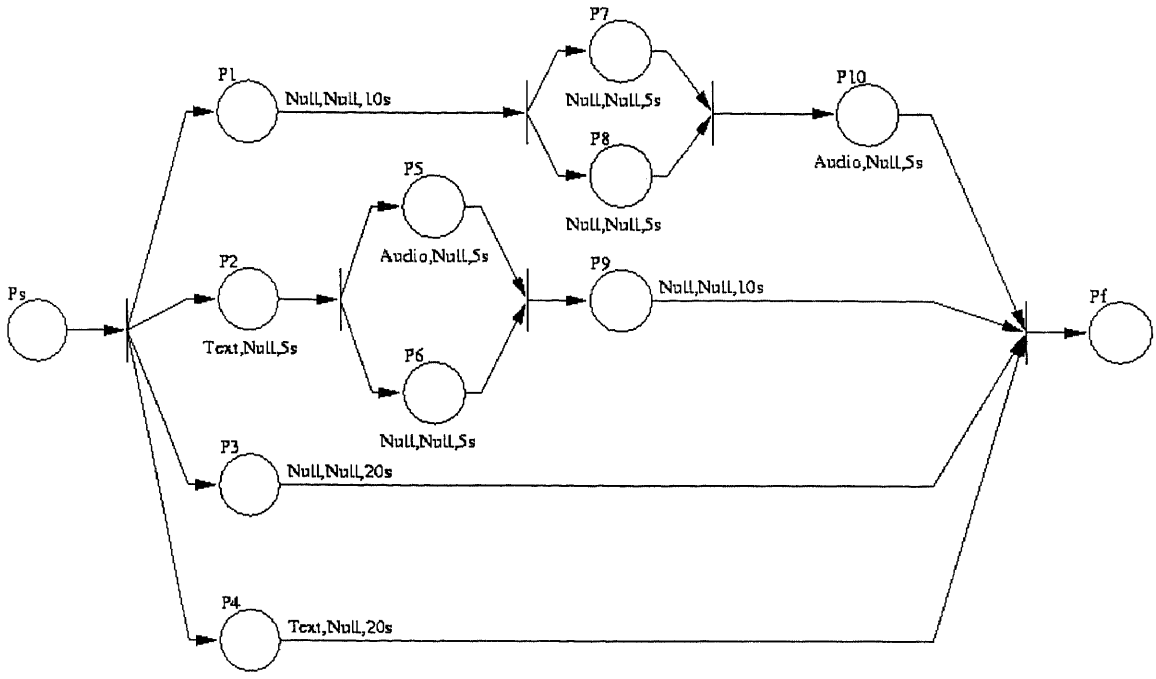

Figure 2.5: The modified plan (with only audio and text capabilities)

Steps 2 and 3 are used to prune the modified object plan. This is carried out by removing and/or merging two or more `null` nodes. The pruning is done in such a way that the modified plan does not alter the execution order of the components of the original plan. Step 2 removes any places with null labels as in Figure 2.6, which we call *parallel removal*. Step 3 merges consecutive null nodes into one node as depicted in Figure 2.7, which we call serial merge.

Since the execution of parallel removal may result in a series of null nodes and vice-versa, steps 2 and 3 of the above algorithm have to be performed repeatedly until they no longer result in any parallel removals or serial merges. The result of applying algorithm 1 on the object plan in Figure 2.5 is as shown in Figure 2.8.

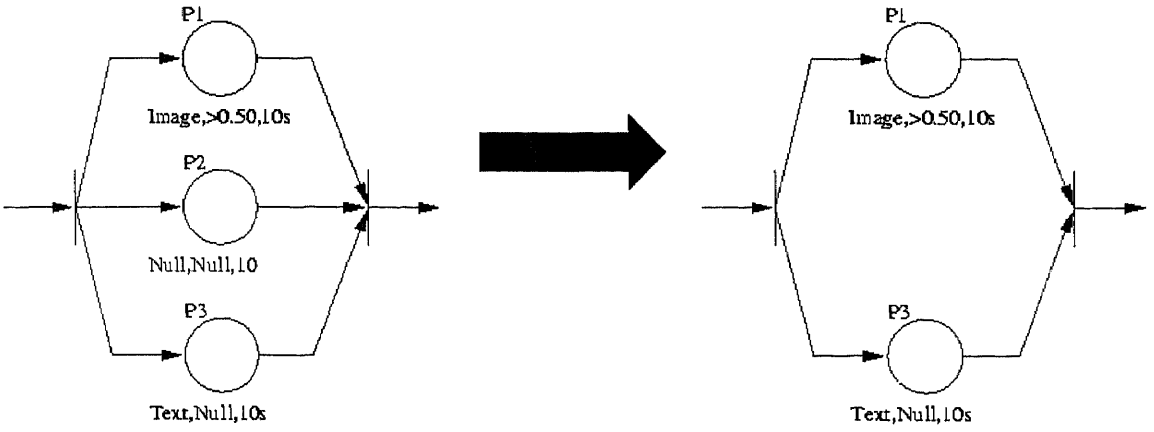

Figure 2.6: Removal of parallel null nodes

Figure 2.7: Merging of consecutive null nodes

Algorithm 2 [Object Delivery]

get each object component p_i in the modified plan such that $L(p_i) \neq null$

insert a token in p_s

whenever a transition t_i fires,

for every $p_j \in \bullet t_i$

mark status p_j as played and

send the status of p_j to the state maintainer

for every p_i, if $m(p_i) > 0$ then invoke the player $L(p_i)$

when $m(p_f) > 0$

erase oblet and plan from the client

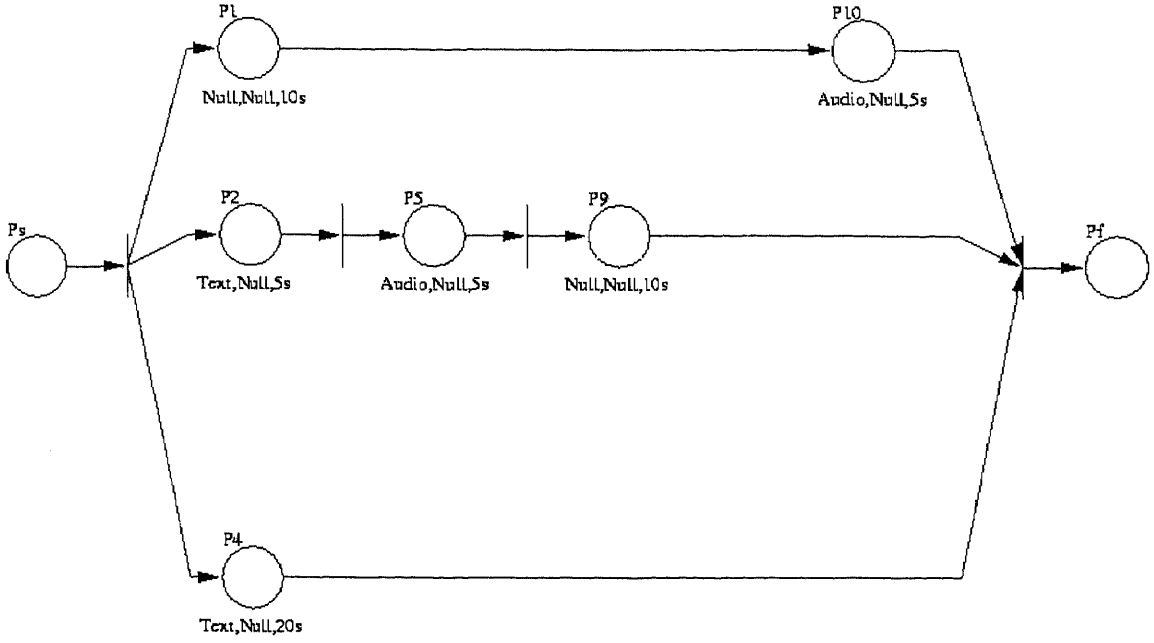

Figure 2.8: An optimized plan

CHAPTER 3

OBJECT DELIVERY IN THE PRESENCE OF NETWORK DELAY

In most existing networks, the delay suffered by packets varies depending on the current network load, the number of switching hops, the network control algorithms deployed, etc.. Although emerging networks hold the promise of guaranteeing various network performance parameters, the variability is expected to continue. Further, it is not possible in most cases to obtain complete information about the extent of this variability in network performance. Thus, networked applications such as the universal access bear the onus of making decisions based on partial knowledge of the network performance.

The network may provide deterministic or statistical guarantees [12, 20] on the delay suffered in the transmission of each message. Deterministic delay guarantees are specified as

$$\text{delay} \leq \text{delay}_{\max} . \qquad (3.1)$$

Statistical guarantees on the other hand are specified as

$$\text{Prob}\{\text{delay} \leq \alpha\} < \beta . \qquad (3.2)$$

In the above equations, $\text{delay}_{\max}$, α and β are parameters that will depend on specific applications.

For each object component c_i, we define the following time instants:

- object component transmission time at server: tr_i
- time instant the object component arrives at the client: a_i
- time instant object component retrieval request received (rr) at the server: rr_i
- time instant object component retrieval request sent (rs) by the client: rs_i

- delay suffered by the request message: dr_i
- delay suffered by the object component: d_i
- server delay in retrieving the object component: $proc_i$

The following relationship holds for each object component:

$$pd_i = f_i - s_i \tag{3.3}$$

$$dr_i = rr_i - rs_i \tag{3.4}$$

$$s_{i \geq} \; a_i = rr_i + proc_i + tr_i + d_i \tag{3.5}$$

In the context of this application, the problem reduces to the computation of the time instant the request for each object component c_i is to be sent by the client (rs_i), given a specific type of delay guarantee. The playing of each stage must commence right after the previous stage has completed and the objective is to retrieve each object component in time to begin playing it before its playback time.

The Petri net execution proceeds in stages, with each stage having several inputs and several outputs. Each transition that fires may cause some object component to be played. For a transition to fire, there must be a token in each of its input places, or in other words, all the previous object components (including null object components) should have finished playing.

Assume that transition t_i fires at time $fire_i$. For t_i to fire, all the object components representing its input places must have finished playing. In other words, the longest playing object component must finish. Let pk be the place in Petri net that represents object component c_k. Thus,

$$fire_j = max\{f_k | p_k \in \bullet t_j\} \tag{3.6}$$

Firing of a transition t_i initiates the playback of every object representing the places in $t_i \bullet$. Playback of an object component c_j begins when any of the input transitions to the place p_j representing c_j fires. For continuous playing, each object component should have been received at the client before its playback time. This implies the following:

$$s_j = min\{fire_k | t_k \in \bullet p_j\} \tag{3.7}$$

$$a_j \leq s_j \tag{3.8}$$

Clearly, for the object components in the first stage of the Petri net, assuming that $rs_j = 0$ (start time), $a_j = s_j = dr_j + proc_j + tr_j + d_j$

3.1 Deterministic Delay Guarantee

Consider a network in which only a maximum end-to-end delay between the client and the server (D_{max}) in either direction is guaranteed and this is the only information provided to the application. This is an example of a deterministic guarantee. In this case, the application can compute the worst-case time instants to send out requests only based on this maximum delay information.

$$dr_j \leq D_{max} \text{ and } d_j \leq D_{max} \tag{3.9}$$

Further, for the playback to start, the object component must be available at the client at the time instant s_j, which implies that the request for the object component

should have been received at the server in time to transmit the object component and suffer in the worst case, a delay of D_{max}. From the above two equations we get,

$$rs_j \leq s_j - tr_j - 2D_{max} - proc_j \tag{3.10}$$

Thus, given the structure of the Petri net (containing the playback durations, and the inter-relationships between the object components), and the size of the object components, we can calculate the worst-case time instants when each object component should be requested. This can be recursively done for each stage of execution.

If $rs_j < 0$, this implies that the object component j must be pre-fetched (before the object plan execution begins) using some anticipatory techniques at the client. If such anticipation is impossible, then those object components for which $rs_j < 0$, will have to be requested at the start of the plan invocation. It is possible that these object components may not arrive in time and some dead time may be introduced. Realistically, in most networks, the probability that the maximum delay is incurred is low, and the probability of incurring any dead time is dependent on the actual distribution of the delay.

3.2 Statistical Delay Guarantee

If the network provides a probabilistic delay guarantee of the sort depicted in Equation 3.2, then the rs_j time instant has to be calculated based on α and β.

$$Prob\{dr_j > \alpha\} < \beta \text{ and } Prob\{d_j > \alpha\} < \beta \tag{3.11}$$

From the previous discussion, $rs_j = s_j - tr_j - proc_j - (d_j + dr_j)$.

In this case, the application can bound the value of rs_j with a specific probability. To do this, the sum of the network delays $(d_j + dr_j)$ has to be bounded to say, τ, which requires the estimation of the following probability p.

$p = Prob\{d_j + dr_j \leq \tau\}$ and choose

$rs_j \leq s_j - tr_j - proc_j - \tau$

Clearly, higher the value of p, the estimate of rs_j will be more effective. A high value of o will cause the value of p to be high, but at the same time, will cause the object component c_j to be fetched very early. So, the compromise is for the application to choose a τ with a corresponding p that is acceptable in most example cases.

To pick a target τ corresponding to a target p, the application needs information on the probability distribution of the network delay $f_D(x)$. The approach proposed by us is the following. First pick a probability distribution function for the delay that has been hypothesized to be a good fit for network end-to-end delay. Examples of such distributions are the Gamma and the Normal distributions. Then pick parameters (may be non-unique in some cases) of the distribution such that the following probabilistic delay guarantee is satisfied.

$$Prob\{d_r > \alpha\} = \int_{\alpha}^{\infty} f_D(x)dx < \beta \tag{3.12}$$

Further, assuming that dr_j and d_j are statistically independent and identically distributed as $f_D(x)$, then the values of p and τ can be easily calculated using standard methods.

CHAPTER 4

IMPLEMENTATION

In implementing universal access, we have used the Java programming language because of its wide use on the Internet and its machine-independent execution capability. We have selected the readily available HTTP as the digital library object transfer protocol. There are two approaches for using Java for implementing universal access - (1) applet programming and (2) application programming. Applet programming approach, which is widely used on the Internet and executed through common WWW browser, is not adequate for the purpose of universal access. This is because, by its very nature, applets are not capable of accessing any of the client files. For universal access, the oblet requires access to client resources (such as files and directories) to detect the client appliance capabilities. Therefore, we use the Java application programming approach, which does not rely on the use of common WWW browser and thus can provide the necessary access required to implement universal access. However, because the task of detecting client-appliance capabilities is dependent on the appliance itself (such as detecting for audio capability), we have combined the implementation of such task with the implementation of the Object Requester. Thus, the implementation of the Oblet and the rest of the system components can be made platform independent.

4.1 System Components

Our implementation follows the system architecture depicted in Figure 2.1. Below is a discussion of each of the system components.

Authoring Tool. The main function of the authoring tool is to provide authors of digital library objects with a user-friendly graphical interface to specify all the com- ponents in a DL object and the various constraints among them. We are currently implementing this using Java to run on Unix Solaris. The authoring tool allows object specification as follows.

1. The author must first specify the synchronization and fidelity constraints by means of Petri net model. A drag-and-drop user friendly-tool to create the Petri-Net representation, partly similar to [1], will be provided.
2. Once the Petri net representation has been specified, the authoring tool checks for its validity and computes the time interval between each component and the component that follows it.
3. For each interval computed in step 2, the authoring tool presents all the components scheduled for display within the period. The author is then required to specify how these components should be spatially displayed by moving the components on the screen. Based on this, the actual X and Y coordinates as well as the Z of each component are determined.

UA Manager. The primary function of the UA Manager is to listen for in-coming requests for digital library objects and deliver the requested objects to the clients. All digital library objects (components) are transmitted by means of the HTTP protocol. We have adopted Netscape WebServer 2.0 [11] for Unix Solaris OS as our UA Manager.

Object Requester. The function of the Object Requester is to provide users with the capability to send requests for digital library objects to the server. Since the implementation of universal access requires access to the client machine, which cannot be accomplished by the conventional Java applet, we have developed our own universal access object requester instead of adopting the common WWW browsers. Since the task of the requester is to serve as a user interface for requesting digital library objects, it runs with minimal client resources. The implementation of the requester consists of two modules. The function of the first module is to accept user requests and fetch the corresponding oblet from the server. The second module is to execute the already fetched oblet locally. The requester also provides additional features such as user and system profile modules which manage and store client profile. Figure 4.1 depicts the Object Requester interface along with its user and system profile modules.

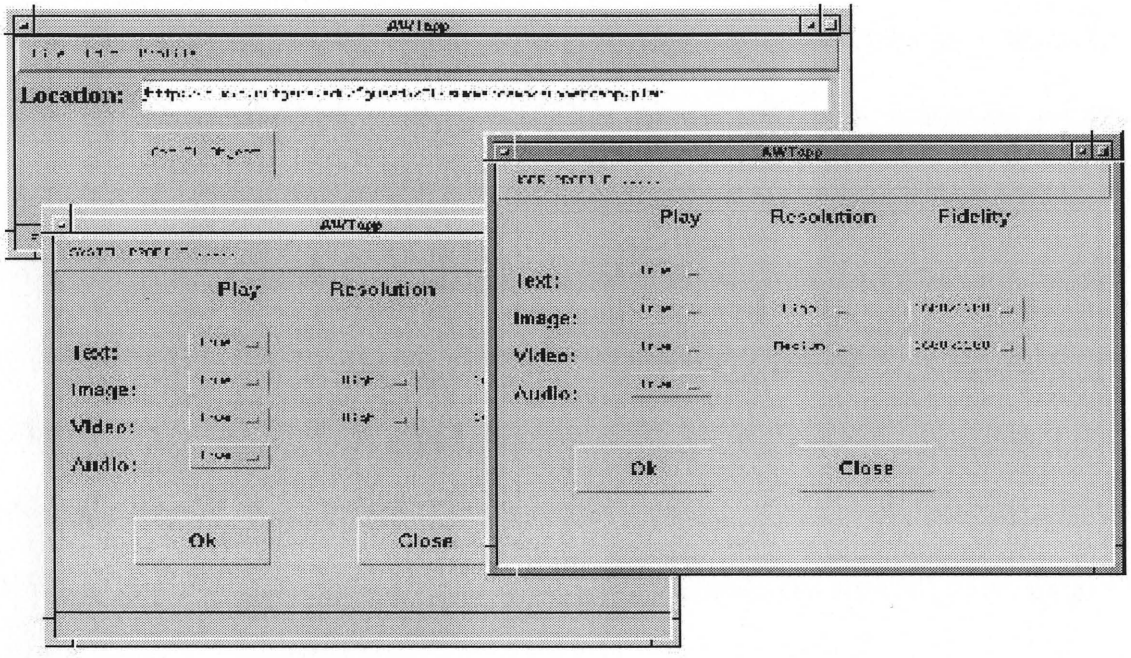

Figure 4.1: Digital Library Requester Interface

Oblet. The primary function of the oblet is to fetch the DL objects and object plans, modify the original object plan based on system and user profiles, and play the object on the client machine according to the modified plan. Since the oblet is central to universal access, most of the implementation effort has been focussed on this component. We have divided the implementation of the oblet into three modules as described below.

- The function of the first module is to detect the client appliance capabilities and update the system profile accordingly. The approach of detecting system capabilities of one appliance is different from that of a different appliance. To have the oblet be platform-independent, we have implemented this module as part of the Object Requester component. Upon arriving at the client machine, the oblet will have the result of detecting system capabilities ready for use. The implementation of this module is similar to that of the program used to detect machine setting during the

installation of any commercial software packages (i.e. prior to software installation, the installation program of commercial software packages detects system setting and resources). Currently this module is under development. In order to use the current system, users need to manually edit the system profile through Object Requestor.

- Once appliance capabilities are detected, the second module evaluates the object plan. The module then uses the information on system and user profiles to modify the original plan. The modified plan is then used to retrieve the necessary components from the server. Only those components whose modalities are supported by the appliance are retrieved. All the retrieved components are stored in the cache directory ready for display.
- Once all the necessary components are retrieved, the third module displays them according to the modified plan. We have used Java programming language to develop each player necessary to display the different modalities.

The implementation architecture and the above system components are depicted by Figure 4.2. An example run is presented in the following section.

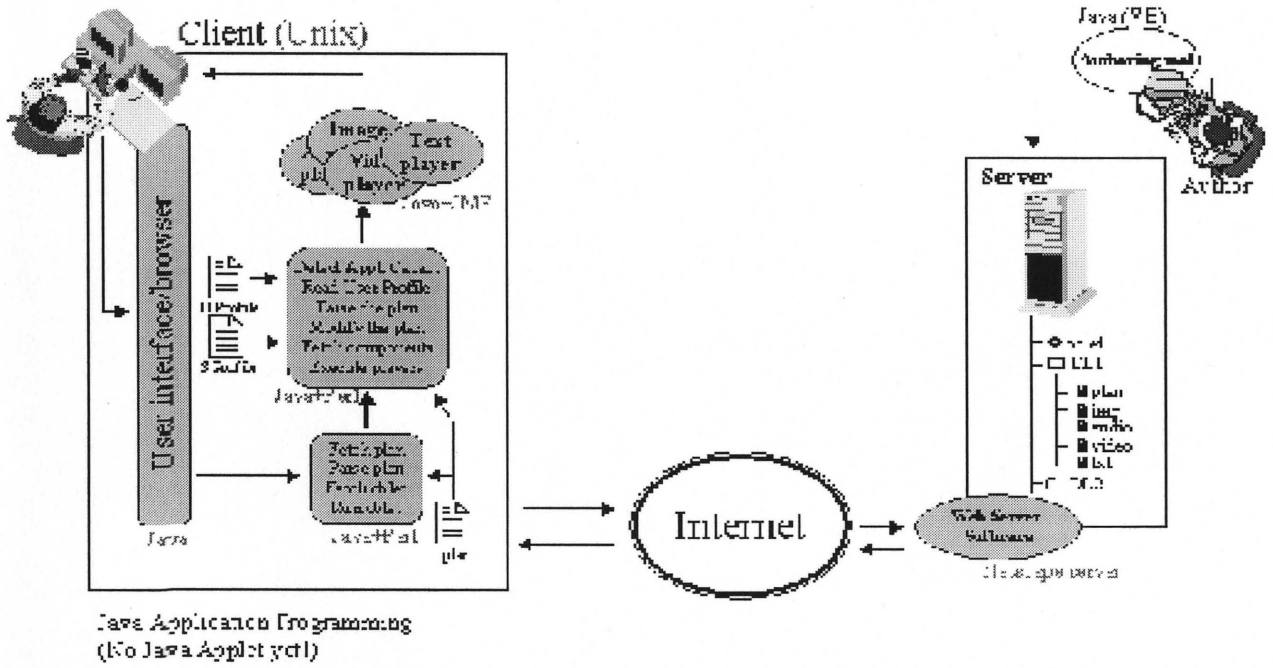

Figure 4.2: Implementation Architecture

4.2 An Example

In the following, we present snapshots taken during the run of our universal access system. For the sample run below, we use a DL object called *SummerCamp98*. Every year the Newark campus of Rutgers University offers several outreach programs to local youth. One such activity is the summer computer workshop which is a joint effort between Center for Information Management, Integration, and Connectivity (CIMIC) and Making Healthy Multi User Sessions In Community (MUSIC) program. Detail of the program and activities can be found at *http://cimic.rutgers.edu*. SummerCamp98 DL object describes how local high school students make use of NASA Satellite re- sources to solve a realistic environmental problem the NJ Hackensack Meadowlands Development Commission (HMDC) is facing. SummerCamp98 DL object consists of

different modalities, including text, image, audio and video. The video component of SummerCamp98 is taken from NJN, a public broadcasting television, which covered the students summer scientific-activities. Figure 4.3 and Figure 4.4 depict all of the components used for SummerCamp98 DL object and the object plan, respectively.

The time unit shown in the figure is in second (e.g. the video component, *camp1.mpg(P14)*, plays for 108 seconds). The total running time for the SummerCamp98 DL object is 173 seconds, as can be seen from Figure 4.4. The final two figures, Figure 4.5 and 4.6, depict two client-screen snapshots taken during the playback of SummerCamp98 DL object on the client machine. Figure 4.5 depicts a snapshot at 15th second where two different modalities, text and image, are displayed. At 62nd second, a video and two images are displayed at the same time as depicted by Figure 4.6.

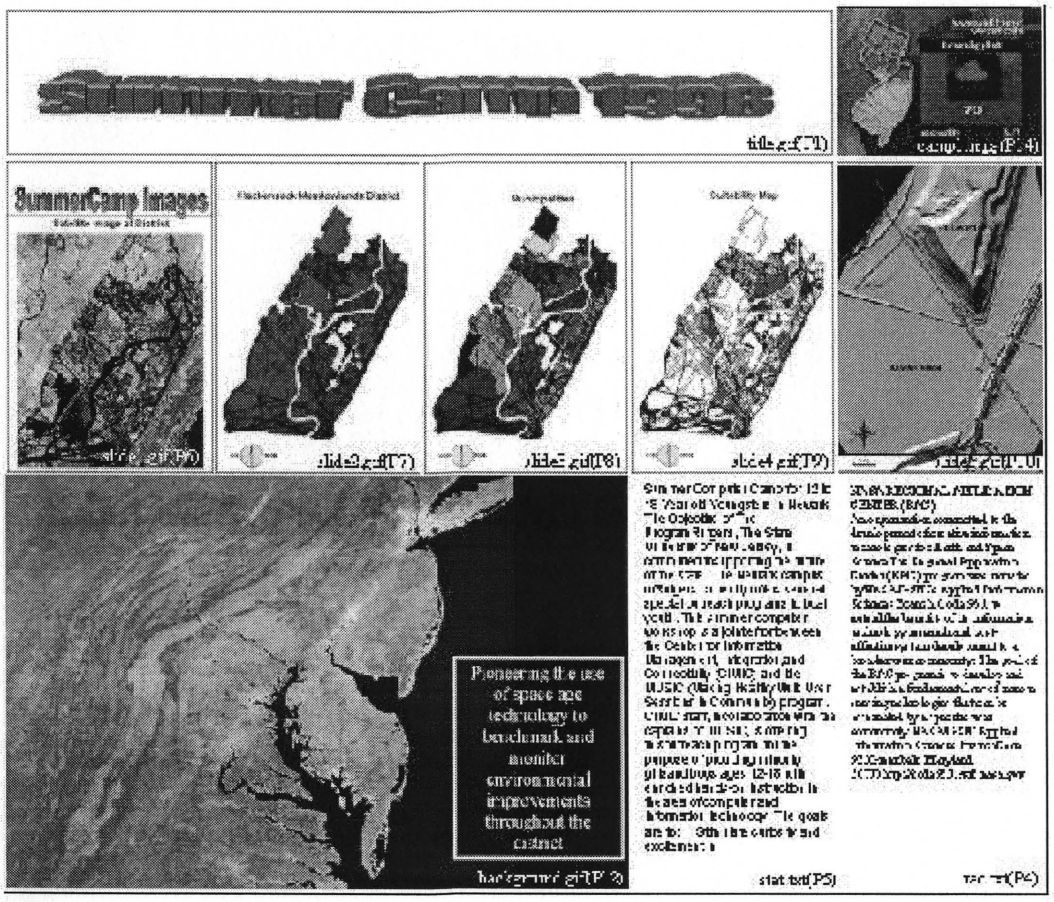

Figure 4.3: The Components used to construct SummerCamp98 DL Object

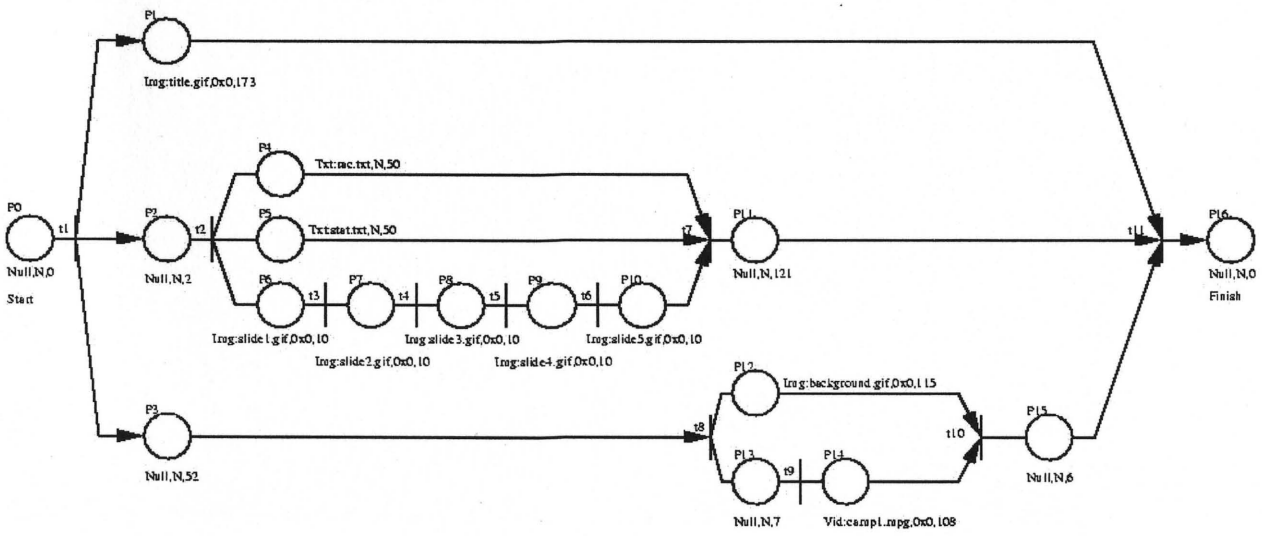

Figure 4.4: The Petri net Model for SummerCamp98 DL Object

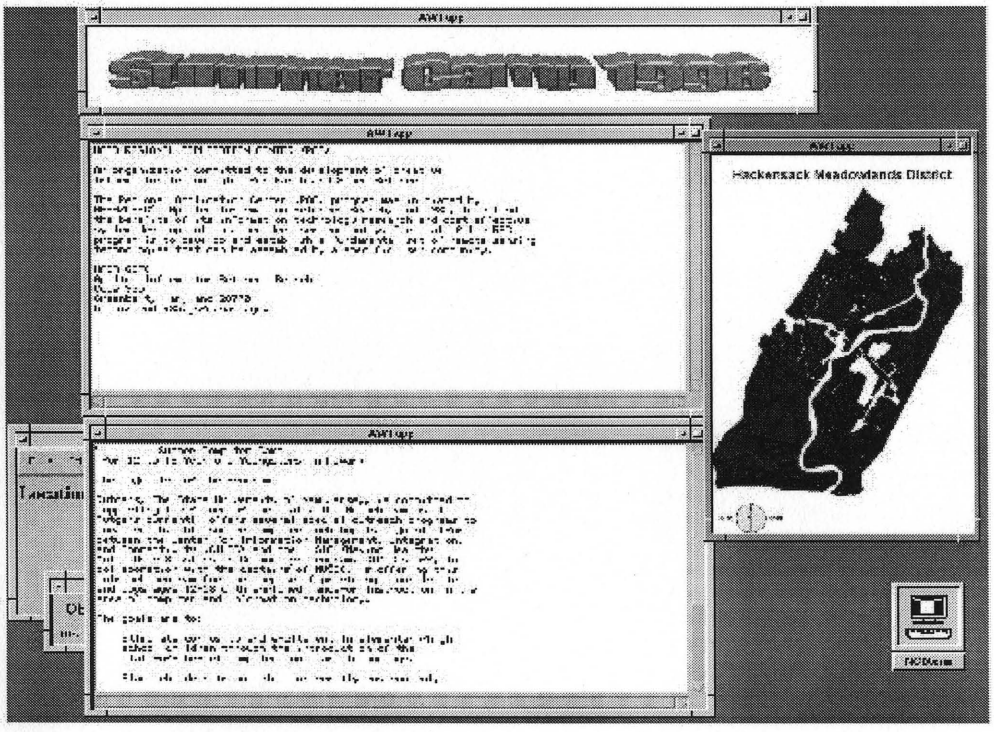

Figure 4.5: Display of SummerCamp98 DL object at 15th second

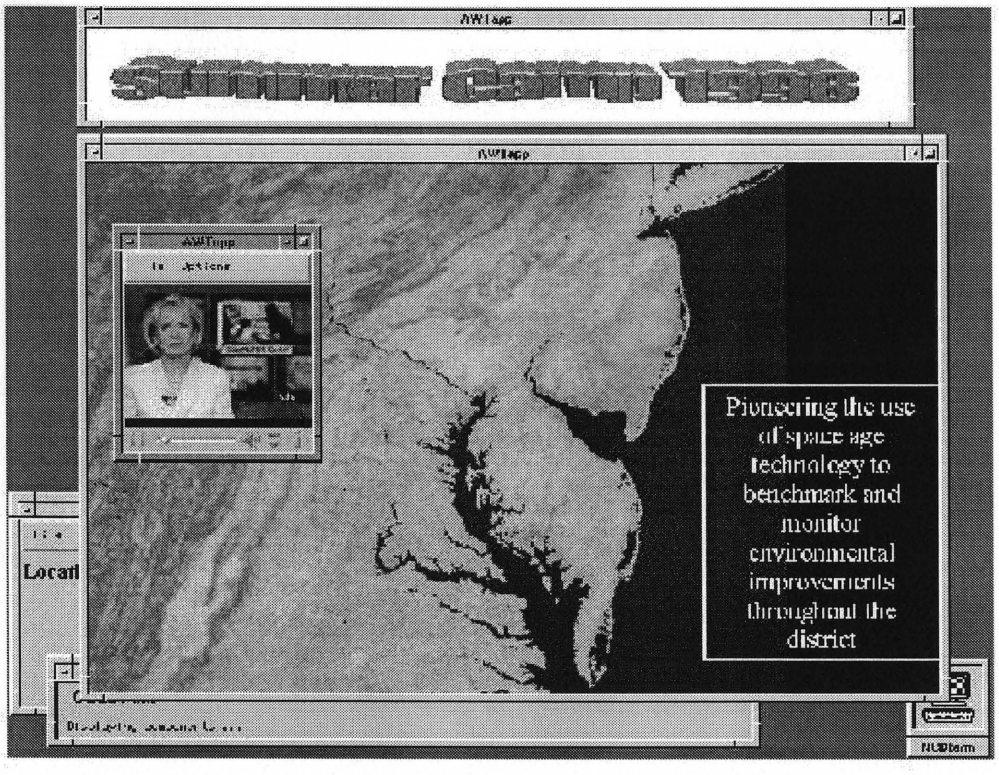

Figure 4.6: Display of SummerCamp98 DL object at 62nd second

CHAPTER 5

CONCLUSIONS

In this thesis, we have presented an approach to provide universal access that facilitates access to complex multimedia digital library object that suits to varying users' capabilities, preferences and requirements. Our approach is based on self manifestation of DL objects, accomplished by using a component called oblet, which is a small piece of software that installs itself on the client and renders the DL objects based on user and system profiles. We have used a Petri net model to represent DL objects that can model synchronization and fidelity constraints.

While this thesis is one of the earliest to address universal access, there are a number of research issues that require further investigation. We are currently investigating several related issues including the following: effective representation of all constraints, conversion of one modality to another, and performance study on the delivery of DL components based on network traffic. We provide an insight into each of them below.

Currently, our MOPN does not include representation of the spatial constraints. Even Though fidelity constraints are incorporated into MOPN, they are merely represented as labels of each component. We are in the process of extending MOPN to uniformly represent all the three types of constraints.

In this thesis, we assume that whenever the client is incapable of rendering one type of modality, all components of that modality are not rendered to the client. This results in loss of information. To overcome this problem, one needs to convert one

modality to another, whenever feasible. We are currently investigating incorporating that feature into our model thus, the author of the object would be able to specify all applicable transformations of modalities for a given DL object component. Based on user's prefer- ence and client's capabilities, the selected modality would then be rendered. Different modalities will be assigned different priorities in accordance with user preferences and author specifications.

We are also conducting a performance study that takes into account the network delay guarantees, both deterministic and statistical. We intend to develop an object plan that takes the cost of rendering the object into consideration; this is accomplished by associating a cost function to each component. We will modify the object plan by taking into account the client's buffer constraints such that it incurs minimum rendering cost by guaranteeing in-time delivery of DL object with minimal dead time.

REFERENCES

1. DAIMI, "The World of Petri Nets," http://www.daimi.aau.dk/PetriNets, May 1999.

2. N. Adam and S. Naqvi, "Universal access in digital libraries," *ACM Computing Surveys*, 28, no. 4es, pp. 105-106, December 1996.

3. N. Adam and Y. Yesha, "Electronic Commerce: Current Research Issues and Applications," in *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, Heidelberg, 1996.30, pp. 13-26.

4. N. Adam and Y. Yesha, "Electronic Commerce and Digital Libraries: Towards a Digital Agora," *ACM Computing Surveys*, 28, no. 4es, pp. 21-32, December 1996.

5. N. Adam and Y. Yesha, "Digital Libraries: Introduction," *International Journal of Digital Libraries*, vol. 1, no. 3, pp. 11-14, April 1997.

6. J.F. Allen, "Towards a general theory of action and time," *Artificial Intelligence*, vol. 23, no. 2, pp. 123-154, August 1984.

7. D. Chen, R. Colwell, H. Gelman, P. K. Chrysanthis, and D. Mosse, "A Framework for Experimenting with QoS for Multimedia Services," *Proc. Multimedia Computing and Networking Conference*, pp. 186-197, January 1996.

8. S. Guan, H. Yu, and J. Yang, "A Prioritized Petri Net Model and Its Application in Distributed Multimedia System," *IEEE Transactions on Computers*, vol. 47, no. 4, pp. 477-481, April 1998.

9. V. Hakkoymaz and G. Ozsoyoglu, "A Constraint-driven Approach to Automate the Organization and Playout of Presentation in Multimedia Databases," *Multimedia Tools and Applications*, vol. 4, no. 2, January 1997.

10. C.L. Hamblin, "Instants and Intervals," *Stadium Generale*, vol. 27, no. 3, pp. 127-134, May 1971.

11. Netscape Inc., "Netscape Enterprise Server," http://www.netscape.com/comprod/server_central/product/enterprise/index.html, May 1999.

12. J. Kurose, "Open Issues and Challenges in Providing Quality of Service Guarantees in High Speed Networks," *ACM Computer Communications Review*, vol. 23, no. 1, pp. 6-15, January 1993.

13. S. Levy, "The New Digital Galaxy," *Newsweek Magazine*, May 31, 1999.

14. L. Li, A. Karmouch, and N.D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 1 and Part 2," *ACM Spinger-Verlag Journal of Multimedia Systems*, vol. 4, no. 1, March 1994.

15. T. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 413-427, April 1990.

16. T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541-580, April 1989.

17. D. Rimer and P. Noglows, "Internet Appliances and Universal Access," *iWord*-http://www.iword.com, March 1999.

18. Y. Shoham, "Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence," *The MIT Press*, Cambridge, MA, 1988.

19. V.S. Subrahmanian, "Principles of Multimedia Database Systems," *Morgan Kaufmann Publisher, Inc.*, San Francisco, California, 1998.

20. D. Towsley, "Providing Quality of Service in Packet Switched Networks," In L. Donatiello and R. Nelson, editors, *Performance Evaluation of Computer and Communications Systems*, pp. 560-586, Berlin: Springer-Verlag, 1993.

21. M. Woo, N.U. Qazi, and A, Ghafoor, "A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information," *IEEE Network*, vol. 8, no. 1, pp. 52-61, January/February 1994.