CWI Quarterly Volume 7 (1) 1994, pp. 3 – 25

Managing the Adaptive Processing of Distributed Multimedia Information

Dick C.A. Bulterman
Multimedia Kernel Systems Project
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Email: Dick.Bulterman@cwi.nl

The term *multimedia* conjures up visions of desktop computers reproducing digital movies, high-resolution images and stereo sound. While many current systems support such functionality, none do so elegantly–especially when data is fetched and synchronized from dissimilar sources distributed across resource-limited networks. Our research investigates general approaches for managing the flow of multimedia information in a distributed computing environment, providing adaptive support for time-sensitive retrieval and presentation based on multimedia document specifications. The benefit of our approach is that it provides flexible, content-based utilization of resources without overburdening the application author/developer.

1. Introduction

During the past decade, multimedia computing has grown from a curiosity to a standard tool in the arsenal of systems designers, implementors and users. The availability of high-density media such as digital compact disks and digital audio tape and the rapid development of input/output hardware have coupled bountiful sources of data with a rich presentation environment. Multimedia computing has progressed to the point where it is common for low-cost computers to provide continuous multimedia output at performance levels that rival the quality of commercial television. While this is impressive, it is not totally unexpected: a typical low-cost computer *is* a television set that has had its tuner replaced by a computer processor and its antenna by a computer's I/O bus. As the processor and I/O bus have increased in speed, and as the concentration of information on source devices has increased, it has been relatively easy to take data and route it to the TV's picture tube and speakers.

Unfortunately, the key to supporting sustained high-rate data transfers is the absence of any substantial information processing activity. As long as the computer simply controls the flow of data and does not overload its I/O bus,the desired output effects can be achieved. (This is one reason why single-tasking PC's with simple operating systems are so successful as multimedia hosts.) Once independent data streams need to be selected or synchronized, however, current operating systems and processor architectures introduce delays and uncertainty that make time-critical delivery of data difficult or impossible to guarantee. As a result, current-generation multimedia only works well within a carefully calculated and restricted computing environment.

The research focus of the Multimedia Kernel Systems group at CWI has been to study user and systems issues associated with *adaptive multimedia*. Our goal is to allow the type and representation of a piece of abstract information to depend on a set of parameters that reflect the (probably changing) state of the presentation environment. Selection and presentation of data can then be guided by evaluating a set of constraints defined and manipulated by the authors, users and servers of that abstract information. Examples of such constraints are: the types and representations of the source data items available to an application; the nature of the relationships among those items; the preferences of a user to view a subset of the set of source items; the abilities of the local presentation environment to support the data representations associated with each of the selected items; the abilities of the network infrastructure to support the selected representations; and (last, but not least), the abilities of servers to fetch (separately or in concert) the requested items in a manner consistent with the timing requirements of the application.

The approach we take to supporting multimedia is to separate characteristics that deal with the representation of data from those that deal with the data's content. Static and dynamic selection mechanisms can then be studied that determine the types of representations used in terms of particular *projections* of information. When selection is made dynamically, the choice among projections can be delayed until the nature of the presentation environment is known. Such a strategy has two advantages: it provides natural support for heterogeneous multimedia and it allows for dynamic balancing of the load on the infrastructure's resources by selecting projections that can be supported by the infrastructure at the moment that an application is being run.

In the following sections, we give an overview of our approach to supporting adaptive multimedia in a distributed environment. Section 2 begins this discussion with a consideration of multimedia data. Section 3 discusses issues that impact the management of multimedia data in a networked environment. Section 4 discusses research issues in supporting adaptive processing in a distributed environment and summarizes our initial results in this area. Although much of our research is in an early stage, we will highlight the various topics under study with examples that will illustrate the results achieved to date.

2. What is multimedia?

Trying to understand the scope and nature of multimedia is a difficult task. The use of the term is so wide-spread and is applied in so many contexts that the label has become almost meaningless. In order to synchronize vocabularies and expectations, this section considers our basic definitions for multimedia computing. It also considers the types of manipulations required for adaptive processing of distributed multimedia.

2.1. Defining multimedia

From the perspective of operating systems research, the following definition might describe the essence of current multimedia computing:

> **mul • ti • me • di • a** <*buzzword; adj*>: a property of applications software that allows for the mixed use of several (chiefly output) media—such as sound, video, text, image and graphic data—in a manner that makes the unsuspecting user think that something unusual is happening on an otherwise conventional computing system.
> *Examples*: multimedia mail; multimedia documents; multimedia demos
> **See also**: Clothes, Emperor's New.

This definition, which is only slightly facetious, highlights the fact that current multimedia emphasizes the presentation of raw data. While a user's perspective on multimedia may focus on the video, audio, picture and text information that is presented in concert by a computer, lower-level manipulation of multimedia data remains restricted to bytes or bits that are processed as all other bytes or bits: in response to normal I/O instructions, generated by standard computer languages and processed by conventional operating systems, running on mainstream computer architectures.

Although current multimedia presents few technological improvements over past information sharing architectures, its mere presence has served as a catalyst for considering software and hardware models that radically alter the way that information is manipulated via computer. As this research work bears fruit, we can expect an updated definition from a future dictionary of computing terms to be:

> **mul • ti • me • di • a** <*adj*>: a generalized ability of programs and architectures to define, manipulate and present all forms of temporal and spatial information.
> *Examples*: multimedia programs; multimedia computers
> *Synonyms*: continuous media, digital media
> **See also**: Data, Universal.

Here, the emphasis has changed from the presentation of data to the processing of information—information that is time-based as well as the more traditional collection of time-less, structure-less bytes. Support for this kind of information processing will require new languages that allow the expression of not only what information is to be manipulated, but also how it is to be gathered; it

will require the development of new operating systems that deal with data and resources based on the content of information rather than just the order in which requests are made; and it will require new computer architectures that will permit the flexible presentation of information based on the state of local and global resources in an interconnected environment. Although we do not address all of these issues directly, our research is aimed at supporting multimedia processing in the broadest possible context. In particular, we look at the development of operating systems and architectures for distributed transfers of multimedia data across heterogeneous systems, under various resource-availability constraints.

2.2. Multimedia data versus multimedia information

Consider the example multimedia application suggested by Figure 1, where several types of data are presented to provide a single message. (For a discussion on how such messages can be created, see [6] in this issue.) This message consists of several groups of text—some with special navigation semantics attached to them—as well as a video fragment and a computer-generated illustration. Although not visible, the message also contains several sound tracks that provide multi-lingual commentary on the other items shown on the display.

FIGURE 1. A multimedia presentation

Instead of describing the details of this application, let us focus our attention on the individual components that the application must combine to create the presentation. Figure 2 illustrates a number of icons that we will use to represent these components. The figure shows three sets of multi-lingual data, two pictures and a film.[1] The multi-lingual data consists of structured text (containing instructions on how to interpret the application), a pair of audio data sets (containing spoken versions of the structured text) and two collections of unstructured text (containing headlines used in the presentation). The two picture sets contain a collection of static illustrations and the film contains sequenced collections of pictures (with and without an integrated audio track).

[1] We will use the terms *film, movie*, and *video* interchangeably.

An unfortunate consequence of the technology-based development of multimedia is that issues associated with the representation of a particular data type are not cleanly distinguished from issues associated with the encoding of the more abstract content-based properties of that data. In order to avoid this confusion, we make a distinction between *multimedia data and multimedia information*, where 'data' refers to a logical or physical entity that can be stored, fetched/generated and displayed and 'information' refers to a logical entities that describes how a data item or collection of items are used in the context of an application.

FIGURE 2. Example components of a multimedia presentation

The distinction between information and data is often application dependent. Even within an application, the mapping between information and data encoding may be context dependent. For example, consider the top-left portion of Figure 2. Inside the box labelled *instructions (text)* are two smaller boxes, each with a structured-text icon and a language label. In the context of Figure 1, these boxes represent two projections—Dutch and English—of the information used as part of the introduction. (Note that Figure 1 shows only the English text.) Next, consider the center-top box labelled *instructions (audio)*, and the two interior boxes with audio icons. The smaller audio boxes also carry the labels Dutch and English; as with the text boxes, each audio box contains a projection of the introductory information in the language identified on the box. From the perspective of data creation and storage, it is often appropriate to consider each data item as a separate entity. In terms of abstract content, however, it may be appropriate to group all four representations together, since all four boxes hold identical information, or to partition the elements into subsets based on language or data type.

In current-generation multimedia systems, the application is responsible for selecting the representations used for particular items. Often, representations for particular target architectures also need to be selected and managed by the application. This approach is unfortunate, since it restricts the flexibility available for viewing and processing multimedia data. More automated means of specifying information objects and data encodings—to ease the burden of the author and to support heterogeneous presentation environments—will be important in the next generation of multimedia systems.

2.3. Selecting information projections

The advantage of separating information and data issues is that, depending on the flexibility of the support environment, individual data encodings of abstract information items can be chosen dynamically (at runtime) based on the preferences of the presentation's author and user. Such selection may be further constrained by the ability of a local presentation environment to support the requested media types or the ability of the support infrastructure to deliver information within the requirements of each data encoding. As we will see, a key issue is determining *who* makes the appropriate selection and *when* the selection occurs.

This is a picture of the initial screen of a sample multimedia document that we call the 'Amsterdam Demo'. This demo allows a user to discover several things about the city of Amsterdam—including, but not limited to, the social and cultural activities of several of its prominent citizens (namely, the CWI Multimedia Kernel Systems group).
The picture is made up of several pieces of information:

1) *A three word phrase ('Welcome to Amsterdam') that actually welcomes the user to the demo;*
2) *A video frame of a charming house along one of Amsterdam's many charming canals;*
3) *A single paragraph of text explaining the purpose of the demo;*
4) *A paragraph containing three hyperdata buttons allowing the user to: take a short tour through the city, get an overview of leisure activities in the city, and get some helpful hints so that the visitor is able to visit the city again...*
5) *A help paragraph, inviting the timid to use the help facility; and*
6) *the CWI logo.*

The picture is actually a low-resolution screen dump. It looks much better when you see it in action.

(a)

```
-rw-r--r--  1  dcab  cl      1094  Feb 2  21:51  asd.1.text
-rw-r--r--  1  dcab  cl    830986  Feb 2  21:51  asd.2.audio
-rw-r--r--  1  dcab  cl   3104129  Feb 2  20:11  asd.3.ps
-rw-r--r--  1  dcab  cl  38093592  Feb 2  20:25  asd.4.film
```

(b)

FIGURE 3. Information vs. data efficiency.

The selection of a particular encoding can be based, in part, on the efficiency of the projections to convey the abstract information. For example, Figure 3(a) describes the illustration in Figure 1; this information could be used as part of a general introduction or as a stand-alone description of the application. Figure 3(b) describes the storage efficiency of several representations of the information contained in the text block. The raw text, in ASCII form, requires approximately 1,100 bytes on a conventional file server. An audio version of the

text, read at a moderate pace using a sampling rate of 8kHz and an encoding size of 8 bit samples, requires over 830,000 bytes in uncompressed form. A PostScript printer file containing a full-scale, 24-bit color screen dump of the original illustration and the text requires over three megabytes to hold the information, and a small video clip (with integrated audio) of the illustration and a verbal reading of the text block consumes nearly 40 megabytes. Depending on the resources available at the data server(s), the I/O bus, the network, and the presentation devices, one encoding may be more appropriate than another in satisfying system constraints for delivering information to the user.

While it is possible to produce compressed versions of some of the data, there would still be a hierarchy of representations, each containing essentially equivalent content but of differing efficiency. Our work uses this observation as the starting point for selecting representations that are appropriate to the needs of the author, the various system-level components and the reader.

3. Issues in processing multimedia informtion and data

Processing multimedia data and information can be divided into three basic tasks: defining the source information set to be used by the application; defining the application-dependent relationships among these set members; and controlling the presentation of subsets of data encodings at runtime. In the following sections, we consider each of these areas in the context of support for adaptive processing of distributed multimedia information.

3.1. Defining source items

The process of source material creation consists of the development of the abstract content of the information and one or more projections of that information onto a specific data representation. (See Figure 4.) Each projection can be the result of a one-time construction operation (such as taking a video camera and filming a particular scene), it can be the result of an authoring/editing process, or it can be the result of an on-demand computer projection of a model of the information item.

Figure 4. Data projections of abstract information.

The creation of a data projection (and the abstract information item) is one of the most time-consuming and complex tasks in the entire multimedia pipeline. The prevalence of television, video games, CD-quality audio and even first-generation commercial multimedia software has made users accustomed to seeing high-quality and error-free productions and presentations. While

new tools to create source material will undoubtedly help, the basic capturing and encoding of source material will remain an artistic rather than technical endeavor. (The difficulty of this task is reflected in the long list of production and support personnel required to create even the most trivial television or radio production.)

Luckily, many of the problems associated with source information creation may be transient. As more information is made available in electronic form, the focus of the problem will shift to information characterization and content-based retrieval. In essence, this will be a process of defining inverse projections from data encodings to abstract information objects (Figure 5.) Each abstract object can be tagged with a number of attributes that will allow it to be retrieved in the context of a particular application. For new material, information classification and tagging can be integrated into the creation process: most encodings do not spontaneously generate, but are the result of careful planning and scripting. A TV news broadcast or Hollywood film could have its plans and scripts included as part of its basic data definition. (The script could be considered a text projection of the abstract object, while the video and audio encodings would be either a single composite projection or a set of single-media projections.)

FIGURE 5. Collecting and classifying multimedia information.

The reality of current multimedia, however, is that many pieces of data exist that are not classified or where no script information is available. Unfortunately, the representations used to hold multimedia information provide little information as to the content of the data itself, making automatic classification a challenging matter of content evaluation. (An exception is text, where information content and data representation are highly integrated.) While advances have been made in audio signal processing that allow the recognition of words and phrases, much needs to be done before sound fragments can be classified and then retrieved for general-purpose use. (A similar situation exists for film and video, and for general pictures.) The problem is compounded by the fact that nearly all source data is inherently context-dependent: the picture of

Amsterdam in Figure 1, for example, could be used in applications on travel, architecture, urban planning, the effects of global warming on low-lying areas, the effects of overcrowding on car insurance rates, etc. Finding a common classification scheme will not be trivial.

Note that in addition to content-based classification, it may also be possible to save representation-based classification information with each object's projection. This could be used to select a particular projection in the context of a specific presentation environment. A choice between two projections could be based on author desires or on the constraints on a presentation environment to support particular encodings. Such decisions could be made statically, as part of the authoring process, or dynamically, in response of the needs of the application presentation environment. As described in section 4, we use this observation as the basis for part of our own multimedia research.

3.2. Combining source items into presentations

Once a set of source elements is available, a subset can be selected and assembled to present a particular narrative. The selection can take place by an application author or at run-time under control of a hyper-information navigation system. Occasionally, special-purpose applications will exist to create, edit and/or classify multimedia information. These applications typically take an object-oriented approach to manipulating data items. (For a discussion of one such approach, see the article on MADE elsewhere in this issue [10].) Many other applications will consist of ordering a collection of existing or dynamically generated information items (and their data projections); these applications typically use a *document model* to describe information interaction.

3.2.1. The document model.

A document describes a collection of information objects and a set of interaction rules that define the relationship among these objects. The document model, although not universally applicable, has a number of advantages for studying multimedia systems:

1) documents model a wide range of real multimedia applications (including electronic books, manuals, letters, and mail; information kiosks; annotated lectures, product presentations, etc.);
2) documents provide a behavioral specification for an entire application, providing a basis for planning resource use—although the hyper-information relationships within the document and other non-linear flow constructs will make the actual behavior dependent on runtime interaction, the document provides a scope on the use of information objects during the application;
3) documents cleanly separate the actions of *authoring* and *reading*;[2]

[2] We will use term *author* to represent the entity–usually a person–that creates a document as a constrained collection of information objects, and *reader* to represent the consumer of a document. While readers can occasionally by authors (while making annotations in documents) and while authors are almost always readers, this general distinction will be useful when considering constraint types in section 4.

4) the document concept is sufficiently narrow to allow comparison of various solutions for encoding and presenting multimedia information.

Note that there is nothing in the document model that requires documents to be static or 'read-only'. A document could define an n-way interactive conversation (as, say, a set of concurrently active communication channels with simultaneous access allowed) or a database browsing system (where the objects are selected at access time). In all cases, the document provides an externalized specification of the behavior of an application.

As with source data creation, applications based on the document model often are difficult to develop. As anyone who has put together a simple multimedia presentation knows, combining various streams of information in a well-synchronized manner—especially in a heterogeneous environment—is no easy task [4],[7]. The cost of producing a quality document is so great that our research is predicated on the belief that documents should be defined only once and then relatively-automatically adapted for use on a wide range of target environments.

3.2.2. Information presentation and synchronization within documents

The primary function of the multimedia document is to describe a collection of information that is to be given to a user and a set of presentation and synchronization constraints on that information. Presentation constraints define the placement of information, while synchronization constraints 'guarantee' that placement will occur at the desired time. When discussing presentation and synchronization, it is convenient to consider data as arriving in *streams*, where each stream contains simplex or complex data that flows from a single source to a single destination.

FIGURE 6. The presentation of synchronized information streams.

Consider the illustration in Figure 6. Here we see a presentation that consists of two streams of headline text, a video stream, two formatted text streams and two audio output streams. Each of the data streams has its own presentation requirements that depend on the characteristics of the data encoding and the

semantics of the message. The presentation requirements for this document include the placement of information (involving the allocation of screen space and audio channels) and the relative ordering of information. Concretely, if both of the headline streams are allocated the same space on the screen, then only one stream should be selected for display. (The one selected would depend on the user or the document.) Alternatively, the two formatted text streams containing English and Dutch captions could be defined to allow both to be displayed at the same time if a user wished to do so. Note that if these captions have a content-based relationship to the video and/or audio streams, the relative presentation time of each stream also becomes important.

The temporal presentation aspects of a document are usually expressed in terms of synchronization requirements. These take two broad forms: *intra-stream* and *inter-stream*. Intra-stream synchronization is usually considered in terms of *jitter* and *drift*. (Jitter is a variation in data arrival rate, while drift is a variation in transmit and receive clocks). Inter-stream synchronization concerns the relative presentation of two or more independent data flows; unlike intra-stream synchronization—which is closely related to the representation of the stream's data encoding—inter-stream synchronization is tied to content-based relationships among the various streams.

The content-based nature of inter-stream synchronization has either pushed solutions for this problem down into the data's representation or up into the application program supporting the document. From the perspective of the document, this has made inter-stream synchronization implicit or explicit. With implicit inter-stream synchronization, a composite data representation is chosen that implements the synchronization relationship by requiring all streams to be stored and presented together. This approach, typical of CD-ROMs, reduces inter-stream synchronization to intra-stream synchronization, in that all synchronization can be supported in terms of jitter and delay control. For example, Figure 7(a) shows an encoding for the example in Figure 6 in which a particular data carrier has had each of its samples (shown between the dark bar) allocated to holding a fragment of video, two fragments of audio, two fragments of formatted text and two fragments of headline text. The format of the data on the carrier may be fixed or variable; that is, each sample may hold small slices of sampled data (providing an approach similar to frequency division multiplexing of communications data) or as discrete samples interleaved within particular time constraints of each medium (an approach similar to time division multiplexing). In both cases, the application designer and/or the application authoring system needs to cram all of the desired data into the available carrier space, essentially forcing the author to resolve inter-stream synchronization issues during application design. Explicit inter-item synchronization (Figure 7(b)) does not require a composite representation; instead, the source items are accompanied by a separate formal or informal synchronization specification that describes how the streams are related. The disadvantage of the explicit approach is that it requires the computer to spend time performing detailed control of the information flow (which is something that current sys-

FIGURE 7. Inter-stream synchronization for Figure 6: (a) reduced to intra-stream; (b) dynamically determined by the presentation program.

tems have no time for), while the advantage is that the quality and combination of items presented can be tailored to the desires of the user and the abilities of the runtime time environment.

Implicit synchronization is by far the most prevalent way of defining complex inter-stream interactions. This type of synchronization is time-tested. It is used to encode film and video and stereo audio: the multiple tracks are glued together on the information carrier—occasionally, as in film, at fixed offsets or otherwise physically in parallel, as on audio tape. Since no special synchronization processing is required at presentation time, relatively simple processors can support seemingly complex data interactions. However, since all data associations must be defined at the time the document is authored, little flexibility is available for custom production scaling (in terms of quality or range of information types supported) and specific assumptions need to be made regarding the nature of the presentation environment.

Explicit inter-item synchronization lightens the authoring burden at the cost of more complex runtime processing. The author 'simply' specifies the nature of the synchronization relationships among the document's streams, leaving implementation to the processor controlling the actual presentation. While this requires significantly more processor intervention than implicit synchronization, it does have the advantage that presentation authoring becomes less media dependent and therefore more portable. It also allows for more customization in the presentation of the document. We return to this approach in section 4.

3.3. Controlling the presentation

Both dynamic selection and content-dependent inter-stream synchronization require some form of runtime control over application behavior. As a means of motivating our discussion in Section 4, the following paragraphs overview the control options available for processing multimedia data.

3.3.1. Local control

The easiest approach to supporting multimedia is to define an application using a fixed document that is tuned to a local environment. By restricting data flow through the computer—and by restricting any other computer activity—impressive multimedia performance can be achieved. This approach is taken by single-threaded PC's that route information from a single source (such as a CD-ROM) to a single multimedia device driver that controls access to the system's display and audio hardware. Simple user navigation is typically provided, but no real processing of the data is supported.

The advantages of the local control approach are essentially the composite advantages of using a fixed data stream in an environment where the maximum system load can be determined at the time the document is authored. The disadvantage is that documents created for such environments are not portable to other environments and often not even scalable across several configurations of a single environment.

Even if enough processing power was available to implement application control over data projections, the limited data storage capabilities of a local environment would probably limit the flexibility in presentation alternatives. Ultimately, data would need to be imported from external sources, forcing consideration of network-induced delays and unpredictable server behavior.

3.3.2. Networked multimedia

A more general approach to supporting multimedia is to start with a networked client/server model. In such a model, data elements may be stored and accessed locally or across the network. Client/server computing based on the networked model is not new. What is new for multimedia is the time-critical nature of multimedia data: it is often important for an application to know the source and/or destination of its data so that effective control over data arrival and departure can be provided. Such control may ensure that sufficient samples are being sent/received per second or it may enforce resource access restrictions. In our case, we assume that such control will also determine the actions of the system if a particular data stream cannot be supported as required or if a wanted resource is not available.

One problem with classical networked systems is that data transfer protocols handle data in a content-neutral manner. All application-level concerns (which, by definition, are content dependent) are handled separately from the basic access and transfer of data bytes. For standard networking, where any data-dependent delay is little more than a nuisance, this is acceptable. For multimedia, where content-dependent delays can corrupt the entire application, it

is not an appropriate model.

Recent studies on performance requirements for networked presentation indicate that a time period of 200 ms is available between samples of continuous video data across networks [5]. Current operating systems and device controllers consume over 80% of this 'buffer period' performing transfer-related processing. If an application needs to be rescheduled to make any content-based decisions in the remaining 40 ms, there is little hope of sophisticated synchronization processing being performed. As a result, current models of network interaction need to move content-based processing lower to the data transfer instead of leaving these as application-based issues.

4. Toward adaptive, distributed multimedia

A major research goal of CWI's Multimedia Kernel System project is studying mechanisms for adaptive multimedia. By adaptive multimedia, we refer to the process of selecting appropriate representations of information items based on semantics of an individual application and the operational context of the underlying environment.

In order to support adaptive, distributed multimedia processing, several models are necessary that address particular aspects of system-wide processing, including:

- a control model of the execution environment that supports the efficient selection of individual representations at runtime;
- a storage model for abstract information that supports the selection of a particular representation based on information content and system requirements; and
- a document model that allows an author (and a user) to specify the abstract behavior of an application in an implementation-independent manner (when possible).

In essence, a general system framework needs to be developed that answers the *who*, *what*, *where*, *when* and *why* questions associated with adaptive representations. The following sections describe our current approaches to addressing these questions.

4.1. Distributed control model

In Section 3.3.2, we introduced the notion of networked multimedia. Distributed multimedia is an extension to this basic approach that includes a more sophisticated control model. In order to illustrate the concerns of distributed multimedia, consider the small network of multimedia sources and sinks in Figure 8. This figure shows three data servers and three user workstations, all sharing a single network. Each server may provide highly structured access to its data (using a database management system), it may provide access via a conventional file server interface, or it may provide access to raw devices connected to external inputs. Note that in each case, servers may access prestored data or generate data 'on request'.

FIGURE 8. The environment

A critical issue is the distribution of synchronization and resource control. For intra-stream synchronization, such control can usually be implemented by low-level client and server services without intervention of application-level code. (For example, a video server on machine y may coordinate its sending of data packets with a video device driver acting on behalf of the client on workstation b.) In most cases, data resources are reserved in advance and low-level processes within the operating systems of each host monitor the correct transfer of data. A problem arises when demand for resources exceeds resource availability. Most current systems address this problem by providing an *admission control* protocol for major network and server resources. Most admission control schemes are biased toward existing 'customers' in the network; if a new customer wants to join, a check is made to see if enough extra resources are available: if so, the new customer is welcomed, and if not, the new entry is told to come back later or to accept lower (or no) system guarantees on resource availability. (The response of the application is not considered part of the protocol.) Control in this case is limited to providing transfer resources between communicating parties; no guarantee is given that all components will actually be able to deliver or use the data within the content-based constraints of the application.

In the case of inter-stream synchronization, the control problem is more complex. Consider the situation where workstation a is receiving text and audio data from servers x and z, respectively, and workstation c is receiving video and audio data from servers y and z. (We can also assume that workstation b is still receiving the video data from server y that it requested in the previous paragraph.) In all of these cases, the inter-stream synchronization needs of each application will determine the demand for data at the servers, and the demand for network resources. Some of these needs may be less severe than others.

In the case of workstation *a*, where text is being combined with audio, exact synchronization of the data streams may not be required. For workstation *c*, where audio and video may be presented in lip-synchronous fashion, the needs will be more critical. Unfortunately, the intended use of each stream and the relationship among streams is typically buried in the application. The servers have no idea of the relative importance—from a context point of view—of each request. Also, since each application executes independently, there is no global state that would encourage one application to give up some of its resources for the good of another application.

In our adaptive support model, we wish to structure the decision making process in a way that increases global awareness without overburdening the application program. To do this, we need to consider the various points at which multimedia information must be fetched/generated, transformed and controlled. We represent this situation in the Amsterdam Multimedia Framework (AMF) [1], as is shown in Figure 9. In this framework, many applications (AP) communicate with adaptive information objects (AIOs; see Section 4.2) via an infrastructure that is managed by a set of local operating systems (LOS's) and one (distributed) global operating system (GOS). The LOS's and GOS handle resource constraints, while the APs and AIOs manage author/user/data constraints. (Note that the implementation structure of the LOS and the GOS is not a principal concern; rather than developing a new operating system model, we currently favor the development of a multimedia co-processor that provides specialized processing support for distributed I/O operations [2].

FIGURE 9. The Amsterdam Multimedia Framework

The purpose of the AMF is to separate control issues at the local and global levels. In addition, we distinguish between content-related processing done by or on behalf of the application and data representation processing done by the AIOs. For example, each LOS is presumed to manage resources that may be needed by one or more (independent) applications on a particular workstation. Since none of the applications will be able (or should be expected!) to manage local caches or local device access, the LOS provides a focus for investigating the local component of time-sensitive data presentation management. Con-

sequently, a given AIO server would communicate with the LOS instead of the application code to determine which of its representations is suited for use on the local system. Similarly, LOS/AIO control communication can be used to meet intra-stream selection and synchronization needs. For inter-stream control, additional support is required. While it is tempting to delegate such content-dependent control to the application, most application authors have little idea how network resource management should be done so that a synchronization goal can be met. Instead, we assume that some form of control will be distributed across the set of AIO servers involved in supplying the data items that need to be synchronized. For example, in terms of our discussion of Figure 8, there is little point in forcing the video server on server y to supply data at a rate consistent with the needs of lip-synchronized audio/video if the audio server on z cannot produce data fast enough. Once an AIO server is informed that the other is not keeping up (or that it is not keeping up), each of the AIOs could communicate directly to determine a good strategy for future support. While it is possible for the application of the target workstation (c) to involve itself in the decision making, the application does not know the state of each AIO server, nor does it know much about the network or the other loads on servers y and z. As a result of its limited knowledge, we assume that the application should concern itself with high-level issues, but not detailed control strategy implementation.

In a similar vein, consider the GOS. It is the only element that maintains a total picture of the resource use throughout the environment. While it is not appropriate for individual transfers to be micro-managed by the GOS, it may be appropriate for the GOS to resolve resource use conflicts. Rather than forcing the network to default to admission control, the GOS may allow more flexible use of resources by providing assistance in server and representation selection.

The control behavior of the AMF is based on a notion of shared implementation responsibility for an application. Such responsibility is only possible if a specification is available that describes the needs (and user/author-acceptable alternatives); this specification can be shared in whole or part with components in the environment. Just as aircraft in the air-traffic control system provide such information in the form of flight plans (allowing detailed control decisions to be distributed across local and regional domains), we assume that applications will also file an 'intended resource use' specification: the application document. Rather than having the application code resolve issues relating to individual object use, the document pre-specifies the alternatives acceptable to the document author. Control for implementing the document can then be distributed to all interested parties, keeping processing as far away from the runtime application code as possible.

We describe the document structure we advocate in section 4.3, after first discussing the nature of the AIO.

4.2. Adaptive Information Object Model

One of the principal control operations in our proposed adaptive environment is a freedom to select individual data representations of an adaptive information object at runtime. Our abstraction for this object is the AIO[3] [3], shown in Figure 10.

FIGURE 10. An example adaptive information object.

The adaptive nature of the AIO is supported through a set of control interfaces. These interfaces include:

- *representation control*: as part of the basic data transfer operation, one of the data projections of an AIO is provided. The choice is governed by the constraints of the application and the nature of the environment. In general, support is provided for heterogeneous target environments or environments supporting a range of data quality (for example, high-resolution images or stereo sound for high-end workstations, low-resolution images or text substitutions and low-quality audio for the low end. The goal is to provide a natural way for supporting heterogeneity within the environment without overloading the application programmer.
- *resource control*: while many of the basic operations of the AIO relate to one-time selection of a representation for a particular instance of the object, it may be necessary to control the delivery of data because of limitations of the environment. These limitations may occur at the server—perhaps triggered by the AIO's own LOS—or they may come for the target host. They may also come from other AIOs who are not able to meet their data requirements or from the GOS, which may signal global resource limits.
- *synchronization control*: the timely delivery of data, either by a single object or by several objects working in concert, is of fundamental importance to the operation of multimedia computing. The synchronization control interface provides a general mechanism for inter- or intra-stream synchronization. The AIO may also initiate control communication with other components via this interface.

[3] Readers in The Netherlands may recognize the AIO acronym as being the same as is used for Dutch graduate students. As with (good) students, the AIO is supposed to take an abstract request and 'do something creative' with it—including figuring out the requestor's basic intentions and satisfying them within the constraints of the implementation environment.

As shown in the figure, a particular AIO may be complex, having many representations available, or it may be a simple server that knows how to present controlled delivery of a single data type. Each AIO may be a separate entity or it may be managed via an object server. The overall role of the AIO is to manage the implementation of constraints on the presentation of data. We return to this in section 4.3.2.

4.3. Document structure

The unifying element among the AMF components is the document specification, which is used to externalize the behavior of a particular application. Each document identifies a set of abstract items that need to be accessed and a set of access constraints that can guide object selection and support. By externalizing behavior, constraints can be resolved based on anticipated future actions, rather than by observing a current state and manipulating the behavioral history of the application. (Note that since it is not clear whether generic multimedia applications exhibit any sort of locality of reference in reusing input or output devices, history and present state do not seem to be particularly good predictors of future activity.) Another advantage of externalizing behavior is that the support environment can manipulate several independent applications concurrently (performing device allocation or interconnect bandwidth) in a manner that would be impossible within the context of a single application.

4.3.1. The CMIF document structure

In order to support the notion of constraint-based document processing, we allow the author to build a document specification that gives a document logical hierarchy and a mapping of information objects on various servers. This approach allows a user to decouple the logical relationship among information objects from their particular data characteristics. In Figure 11(a), we see an application that refers to 9 information objects (leaf nodes A–I). The tree is ordered as a collection of parallel and sequential modes that are evaluated top-down, left-to-right. In (b), the nodes are mapped to *virtual channels*, and (semi-automatically) placed on an activation timeline by the authoring system. Fine-grained interactions can be specified by the author using *synchronization arcs*; these indicate (relative) presentation constraints of a particular object.

Each of the virtual channels can be displayed at runtime, or a subset can be selected by a reader. The header of each channel (which is indicated by the icon of the default data type of that channel) can contain global channel constraints, while each AIO instance on a channel can contain instance-related constraints. (A full description of CMIF and the Amsterdam Hypermedia Model are beyond the scope of this paper. For more details, see [8], [9] and the article on authoring in this issue [6].)

4.3.2. Example constraint operations

Constraints are a set of parameters that express choices in or limits on the use of information objects. The purpose of developing a multi-level set of constraints

FIGURE 11. An application specification as a hierarchy (a) and a set of constrained objects (b).

is to guide the runtime selection of information object projections. These constraints fall into two broad categories: *static constraints*, whose impact on a particular projection selection is known when the document is defined, and *dynamic constraints*, whose impact on projection selection is not known until the document is accessed. (Figure 12.)

Static constraints on the selection of a projection are those that are defined when the document or information object is authored/created. While the resolution of these constraints occurs at runtime (that is, a particular projection can be selected based on the alternatives provided with the document), the alternatives usually can be analyzed before presentation begins. Examples are:

- *information encodings*: information can be mapped to one of several projections, each of which may use one or more types of presentation media. While selecting among projections is a runtime task, the range of projections available are assumed to be know statically—even though the information itself may be synthesized at the time it is referenced. The projections may be kept together in an information object database, they may be stored separately in media-related databases or they may be stored as files in a file system. Access/charging constraints may also exist.
- *author preferences*: while defining a presentation, the author can specify the representation(s) that best project the information under a variety of circumstances. These circumstances will have both syntactic and semantic components, such as 'end together' or 'place object A to the left of B because the content of B is identified as being to the right of A.' They

FIGURE 12. Coarse projection-constraint classifications

can be conditional, such as:

if audio is active **and** if video is active
then delay until end of longest sequence
else delay *n-units* of time;

or they can involve user interaction. Author preferences are static because they are defined before a particular presentation takes place; the selection of a given alternative will also depend on the combination of user preferences and system states.

Dynamic projection constraints are those that depend on the combination of a particular user of a document and a particular presentation environment. Here, support is provided for heterogeneous presentation systems *and* heterogeneous users. In addition, we also include constraints based on the runtime state of the interconnection environment. Dynamic factors include:

- *reader preferences/abilities*: for a given presentation, the needs of readers in consuming that information will depend on a variety of factors, including general user preferences (which can be called *reader style preferences*) and basic user functional limitations (which can be called *reader functional preferences*). Style preferences may dictate that a user would rather receive text-based projections of information than audio-based projections, while functional preferences may dictate that, because a user is blind, audio data is required in place of text;
- *system preferences/abilities*: for a given presentation, a homogeneous presentation platform cannot be assumed. Some presentation platforms will support a wide range of input and output devices, but others will contain only a subset of those potentially available. (This is the *presentation system heterogeneity*—in general, our approach is to adapt the information representation dynamically to the needs to the presentation system, rather than adapting the underlying application.) As with readers, constraints on the presentation environment are analogous to style-based or

ability-based, where style constraints are local preferences (perhaps based on performance or reliability concerns), while function constraints would be based on the presence/absence of a particular type of input/output functionality;

- *environment preferences/abilities*: for a given presentation, the basic ability of the support environment to provide information to the user will be constrained by resource availability across the environment. While, for example, a particular user may prefer to see a sequence of video images instead of a block of text, the underlying environment—including information servers, transport networks, intermediate buffering hosts, local operating systems, etc.—may not be able to provide this service (even if the local presentation environment can).

The nature of constraints is not yet well understood, and the list above is by no means complete. What is clear, however, is that some mechanism is required to specify system behavior in a more direct than is currently available. The approach of having an author specify a set of interpretation constraints allows for a degree of extra information on intent and content evaluation that can be important in making resource allocation decisions across the network. Similarly, having execution environments specify the limits on their resource availability should allow allocation decisions to be made at a lower level than is currently available. This philosophy is more important than any single set of constraint characterizations.

5. Current status

The goal of our work is the development of a shared-specification architecture that supports adaptive application management and (whenever possible) non-reservation resource control. Our work to date has included a specification and authoring system for describing and executing application specifications [7],[8],[9]. We have also implemented a player to decode and display specifications for a broad class of applications [11]. The process of AIO-based constraint resolution involves taking the relatively informal and still environment-specific application descriptions and integrating negotiated AIO support. This requires revising the specification language to include options suitable for selecting individual representations, and low-level support for feeding AIOs with application *and* environment context information to transparently select appropriate information representations. Several small-scale experiments have been completed, such as integrating an adaptive JPEG image server in a network of heterogeneous systems [3]. Work on defining and integrating distributed scheduling algorithms is currently in progress, as is a definition of higher-level information structuring mechanisms that can be used to support multi-projection information models.

Acknowledgements

The research described in this paper has benefited from the interactions and implementations provided by various members of the Multimedia Kernel Sys-

tem project. Lynda Hardman, Guido van Rossum, Jack Jansen and Sjoerd Mullender have all contributed insights, programs, demos and advice that have determined the shape of our work during the past three years. In addition, Robert van Liere and Dik Winter (both of CWI) have given generously of their time in discussing and, in come cases, implementing portions of our environment.

References

1. D.C.A. Bulterman (1993). Specification and Support of Adaptable Network Multimedia. *ACM/Springer Multimedia Systems*, 1(2), 68–76.
2. D.C.A. Bulterman and R. van Liere (1991). Multimedia Synchronization and UNIX. *Proc. 2nd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, Heidelberg, Nov., 108–117.
3. D.C.A. Bulterman (1993). Retrieving (JPEG) Pictures in Portable Hypermedia Documents. In *Multimedia Modelling*, T.-S. Chua and T. L. Kunii (Eds.), World Scientific, Singapore, 217–226.
4. M.C. Buchanan and P T. Zellweger (1993). Automatic Temporal Layout Mechanisms. *Proc. ACM Multimedia '93*, Anaheim CA, 341–350.
5. S. Crimmins (1993). Analysis of Video Conferencing on a Token Ring Local Area Network. *Proc. ACM Multimedia '93*, Anaheim CA, 301–310.
6. L. Hardman and D.C.A. Bulterman (1994). Authoring Interactive Multimedia: Problems and Prospects. CWI Quarterly **7** (1); elsewhere in this issue.
7. L. Hardman, D.C.A. Bulterman and G. van Rossum (1994). The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. *Comm. of the ACM*, **37** (2), 50–62.
8. L. Hardman, D.C.A. Bulterman and G. van Rossum (1993). Links in Hypermedia: The Requirement for Context. *Proc. ACM Hypertext '93*, Seattle WA, 183–191.
9. L. Hardman, G. van Rossum and D.C.A. Bulterman (1993). Structured Multimedia Authoring. *Proc. ACM Multimedia '93*, Anaheim CA, 283–289.
10. I. Herman, G.J. Reynolds and J. Davy (1994). MADE: A Multimedia Application Development Environment. *CWI Quarterly*, **7** (1); elsewhere in this issue.
11. G. van Rossum, J. Jansen, K.S. Mullender and D.C.A. Bulterman (1993). CMIFed: A Presentation Environment for Portable Hypermedia Documents. *Proc. ACM Multimedia '93*, Anaheim CA, 183–188.