Multimedia Systems (1993) 1:68–76

Multimedia Systems

Specification and support of adaptable networked multimedia

Dick C.A. Bulterman[*,**]

The Multimedia Kernel Systems Project, CWI: Centrum voor Wiskunde en Informatica, Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands

Received February 6, 1993/Accepted March 1993

Abstract. Accessing multimedia information in a networked environment introduces problems that do not exist when the same information is accessed locally. These problems include: (1) competing for network resources within and across applications, (2) synchronizing data arrivals from various sources within an application, and (3) supporting multiple data representations across heterogeneous hosts. Often, special purpose algorithms can be defined to deal with these problems, but these solutions are usually restricted to the context of a single application. A more general approach is to define an adaptable infrastructure that can be used to manage resources flexibly for all currently active applications. This paper describes such an approach. We begin by introducing a general framework for partitioning control responsibilities among a number of cooperating system and application components. We then describe a specification formalism that can be used to encode an application's resource requirements, synchronization needs, and interaction control. This specification can be used to coordinate the activities of the application, the operating system(s) and a set of adaptive information objects in matching the (possibly flexible) needs of an application to the resources available in an environment at run time. The benefits of this approach are that it allows adaptable application support with respect to system resources and that it provides a natural way to support heterogeneity in multimedia networks and multimedia data.

Key words: Networked multimedia – Resource allocation – Adaptive data models

1 Introduction

Networked multimedia is a generic term that describes a model of information distribution in which data sources are located separately from data sinks. Networked multimedia offers a number of advantages to applications: (1) the network provides a convenient means of distributing information to other sites, (2) it provides access to compute servers where special purpose processing of multimedia data can take place, and (3) it provides access to central servers that can be used to store the often vast amounts of data required to represent multimedia information fragments. At the same time, however, networked multimedia presents an application with a number of disadvantages; compared to accessing and manipulating multimedia data locally: (1) the data delivery characteristics of the network are difficult to predict and control, (2) the contention for critical system and data resources across the network makes balanced data access difficult to achieve, (3) and differences among network hosts may make data objects difficult to share.

In order to make networked multimedia more useful to application designers and users, considerable effort has been devoted to studying the way that data servers, operating systems, and network infrastructures provide access to time-sensitive data. Most of these approaches define extensions to "conventional" means of accessing remote data to provide predictable network service and performance. For example, predictability is provided in data object servers (either file servers or database systems) by supporting efficient object storage and retrieval/delivery (Danthin et al. 1992) and in operating systems by supporting quality of service guarantees for delivery of (possibly) complex data types (Anderson et al. 1992; Ferrari 1991; Govindan and Anderson 1991; Hanko et al. 1991; Lesley et al. 1992; Tokuda et al. 1990). At the network level, support for predictable multimedia is provided by, among others, admission control techniques that regulate the use of resources and by technologies that provide deterministic network/data access (Clark et al. 1992; Hayter and McAuley 1991; Jeffay et al. 1992; Little and Ghafoor 1991; Topolcic 1990; Verma and Ferrari 1990). The basic premise of this work is that an application will request a data object (or a collection of objects) requiring a specific amount of resources during a specified time. If these resources are available, the application can execute; if not, the application is either delayed or it is denied access to the resources.

An implicit assumption in current approaches is that the application program bears a significant control burden in requesting and coordinating multimedia information. Consider, for example, the application environment shown in Fig. 1. Suppose that one of the requested data streams could not be made available at the required level of service. An application may

* *e-mail address:* dcab@cwi.nl
** *Present address:* Department of Computer Science, PO Box 1910, Brown University, Providence, RI 02912, USA
e-mail: dcab@cs.brown.edu

decide to skip this data object (or the collection of objects associated with that stream), or it may decide to substitute another data object or object server. In effect, the application program is engaged in a process of resource allocation. It is attempting to match its data needs to the resources available at various locations in the support infrastructure. Unfortunately, to allocate resources efficiently – even if this means only selecting from a set of available data streams – the application needs to know how to best use the available infrastructure. This involves issues that most applications programs are ill equipped to resolve. (It also requires applications to be rewritten when they are moved to new environments.) Alternatively, the operating system or the data servers could handle all resource allocation, but the (local) operating system will have only limited knowledge of the state of each of the servers and other applications active within the networked environment, and the data servers will be able to manage only their own streams but not other streams in the infrastructure.

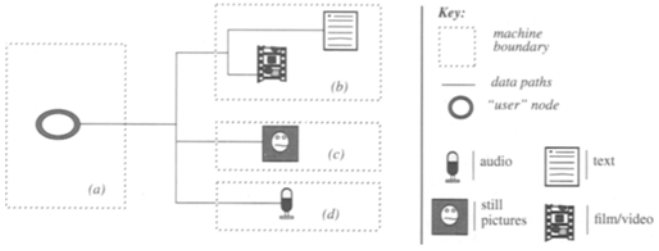

Fig. 1. Simple multimedia information client/server example. The client is fed by three servers one of which supplies two data types. (The structure of the client and the client's application are not shown)

This paper presents an alternative approach to supporting networked multimedia. Our work is aimed at studying coordinated application and infrastructure-based support for adaptable applications. Here, "adaptable" means that an infrastructure can be defined so that an application can adapt to the resources available at the time the application is run. The types of adaptability we consider include responding to (possibly transient) variations in the number and composition of network and remote resources that are available during application execution, as well as application and server support for heterogeneous collections of input/output devices. Our approach is based on two mechanisms. First, we define an application specification that explicitly describes the data objects used by an application, the manner in which the objects interact, and the available ranges of alternatives that are acceptable to the application at run time. Second, we define an interface to the data objects that allows alternative representations to be selected at run time by a process of application-transparent negotiation at run time. This approach is specifically geared to applications that have a document or presentation structure. An authoring system [such as that of Van Rossum et al. (1993)] can be used to generate a specification that can be accessed/executed at a later time. By allowing the execution to be adaptable, one specification can potentially allow an application to be available within a heterogeneous environment under a range of resource availability conditions. As will be discussed, this can help to reduce the high cost of authoring multimedia applications and it can lead to more efficient use of multimedia infrastructures.

In the following sections, we first describe a framework for partitioning control responsibility within the system infrastructure to support adaptable applications. Next, we describe a specification formalism that can be used by all components of the infrastructure that support the application. We then describe adaptive information objects, which provide a front end for a flexible object storage/synthesis interface. We close with a discussion of the current status of prototype implementations supporting these components and with directions for further work.

2 A Framework for adaptable networked multimedia

In order to support adaptable networked multimedia, an underlying framework is necessary that defines how information is structured, composed, accessed, and manipulated, as well as how it is stored and transmitted among sources and sinks. In this section, the Amsterdam Multimedia Framework (AMF) is presented. To put the AMF in context, its description is prefaced with a discussion of the type of multimedia applications it was intended to support and a review of the control issues that the framework must address.

2.1 Multimedia application descriptions: the document

Our abstraction for organizing multimedia information is the document. A document defines a collection of data objects and a description of how these objects interact. Each object may consist of previously stored information or information that is generated dynamically. Such information can be of either a single data type (such as pure audio or video) or of a composite data type (such as video with embedded audio). An active document is called a presentation.

Figure 2 provides an example of a document-based multimedia application – in this case, a fragment of a walking tour of Amsterdam. This fragment contains a title bar using text data, a description of typical shopping street using video data, several "buttons" using text data that control navigation through the document, a CWI logo using still image data, and two sets of captions (one in English, one in Dutch) using text data. The document from which this example is taken also has two sound tracks (one in Dutch, one in English) that provide audio commentary during the tour. The data objects can be stored on various servers located throughout the environment. When the document is accessed, each of the individual object streams is sent to a document player, which implements any high-level (nonembedded) synchronization constraints among the streams (such as matching the subtitle text with the audio data). Each document, such as the tour of Amsterdam in our example, is specific to a particular application; the player is a general purpose program that must be able to play many different documents.

Fig. 2. An example multimedia application. The *rectangles* along the bottom represent navigation controls; the *square* in the picture is a hyperbutton. The two lines of text are captions that accompany multilingual audio

The primary advantage of using a document model is that it provides an explicit behavioral specification. This behavioral description can be used to fetch individual data objects by a player, but it can also be used prior to execution to analyze expected application resource use and feasibility for a given environment (Buchanan and Zellweger 1992a). Assuming the specification was defined to run in a general purpose environment (that is, it was not designed for use on one particular platform), the specification can also be used to determine how (and if) the synchronization needs of the application can be supported at run time (Bulterman et al. 1991).

Creating documents using authoring systems or program-based toolkits is typically an arduous task (Anderson and Chan 1991; Hodges et al. 1989; MacroMind 1990). One motivation for investigating adaptable networked multimedia was to provide reduce the overall effort of producing multimedia presentations by means of reusing document structures in multiple environments once they were authored (Bulterman et al. 1991, Hardman et al. 1993b).

2.2 Supporting adaptable documents: data representation and document content issues

During analysis of a document, it is usually assumed that the specification provides a precise description of the needs and characteristics of the application. Our work investigates the use of a specification as a guide to possible resource and data use, depending on the resources available at execution time of the document. While pre-execution analysis can provide a useful first step in determining specification feasibility, it cannot resolve all of the issues that may influence the run time needs or run time behavior of an application. In defining a basis for adaptable documents, two classes of issues can be identified that influence document analysis and support: (1) issues associated with the physical representations of multimedia data and (2) issues associated with the content-based interactions of users with multimedia data.

Representation-based issues

One major difference between multimedia data and "conventional" electronic data is that multimedia information can require specific service guarantees to preserve synchronization properties of the data. These properties are the consequence of how multimedia data is represented; they are not the meaning of the data itself. While the representations of each data type vary, there are several common issues that are relevant for all time-sensitive multimedia data:

1. Intraobject synchronization: each component can have synchronization constraints that are related to the type of data being retrieved. For example, the video, audio, and caption-text data in Fig. 2 each have their own synchronization constraints. These constraints must be supported by the source environment, the network infrastructure being traversed, and the destination environment. These constraints can usually be managed on an end-to-end basis (Ferrari 1990, 1991).
2. Interobject synchronization: in general documents, data will be encoded in separate streams of objects, each of which may be located at different hosts. While interobject synchronization is often controlled in the context of an application, the composite transfer of data may need to be coordinated to improve system efficiency. For example, audio data and caption text can be synchronized by the application, but the use of markers placed in the data objects and evaluated by the support software improves efficiency.
3. Heterogeneity: in general environments, all of the presentation workstations will not be identical. Information may need to be adapted at either the source or the sink to meet the needs of a presentation environment, where the adaptation process may itself have an influence on which parts of a document are available to a user – a process that may also impact scheduling, resource allocation, and synchronization with the network.

Bandwidth management can also be included among the representation-related issues. In spite of the trend toward faster networks and more highly encoded information, the transfer capacity of the various interconnects will remain a critical resource that must be managed, either because application demands will grow or because multiple types of networks will coexist at a site, requiring a degree of coordination and management to allocate local and global resources efficiently.

Content-based issues

The reason for isolating representation-based issues is to consider ways of providing other than worst-case resource allocation in an adaptable environment. In a similar manner, the actions that occur based on the content of a document will also affect the way that documents are fetched, composed, and delivered. These include:

1. User selectivity: not all of the information available in a document may be used each time the document is accessed. For example, although the document in Fig. 2 supports multilingual audio and/or captions, users usually do not want to

hear or read all of the available languages simultaneously. (Note that the selection of desired information is made at run-time – not author time, and that the selection may be influenced by the facilities available on a given playback platform.)

2. Presentation nonlinearity: the order in which objects are accessed and presented depends on the document structure and the result of user interaction at run time. Users may want to jump around in a document by scrolling forward or backward or by following hyperlinks that have been defined statically or dynamically in the document. For example, in Fig. 2, a small rectangle is visible over a traffic sign in the midright portion of the street. Selecting this button will transfer the user to a section discussing the merits of getting around by bicycle, car, and tram in the city.
3. User flexibility: in general, documents are activated because a user wishes to obtain information. Given a choice, it is our experience that users will tolerate a lower-quality presentation instead of being denied access to a presentation totally. Such lower quality may manifest itself as (slight) delays in the presentation of parts of a document or in the substitution of a lower-resolution form of information for a higher-resolution one. (The term "resolution" is used broadly: it could mean substituting a piece of text for a picture or an audio fragment for a piece of video.)

Each of these factors affects the support mechanisms required to provide adaptability in a document. The notion of user selectivity means that static analysis of a document before it is executed may not provide an insight into how a document will actually be used. Similarly, presentation nonlinearity could result in "jumping" to various parts of a document, each with its own quality of service requirements. As a result, efficient use of an infrastructure will require dynamic, rather than static, assignment of resources across the network. User flexibility means that some degree of run time negotiation may need to be supported so that the information presented to the user can be matched to the resources available at the time individual data access requests are made.

2.3 The AMF

Although many of the techniques required to support representation-based control and, to a lesser extent, content-based control can be taken from existing research results, it is important that these results be applied within a framework that provides an explicit partitioning of control concerns across components in a network infrastructure. This provides a definition of the scope of each technique and can result in better interaction among components. The AMF provides this partitioning for our work.

Figure 3 illustrates the AMF. Here, many applications (AP) communicate with adaptive information objects (AIOs) via an infrastructure that is managed by a set of local operating systems (LOSs) and a global operating system (GOS). The LOSs and GOS coordinate resource allocation, while the APs and AIOs request and deliver information, respectively. Note that

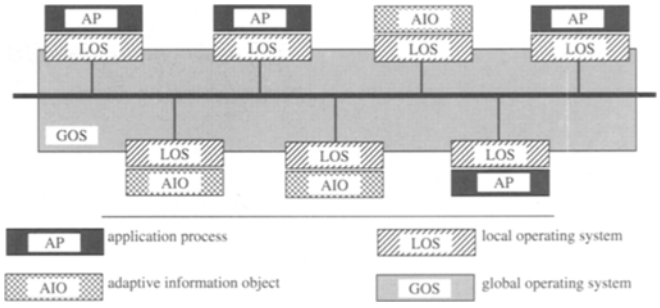

Fig. 3. AMF "active" components

the AMF does not solve the multimedia data transfer problem, it only characterizes the components in an environment and it indicates their interactions. Individual models that implement the general functionality of the framework need to be developed.

The general structure of the AMF is similar to client/server models of networked computing. The difference between the AMF and these models is that within the AMF, the control of multimedia is a cooperative process that requires content-based coordination among all components. For example, assume that one of the APs requests two object streams, each from separate AIOs on two separate hosts. Assume further that one of the AIOs is able to meet the service quality request of the application directly, while the other one is not. In this case, both could inform the application of their available degree of service (leaving the application to select an appropriate recovery action) or the two AIOs could communicate with each other to determine if there was a common level of service that both could provide that was acceptable for that application. This could be possible if:

1. Each of the AIOs was aware of the other's presence
2. Each AIO was aware of other's service constraints, either directly (from copies of the application specification) or by intervention of the GOS and/ or each LOS.
3. Both AIOs were aware of the range of options acceptable to the application and supportable by the LOS/GOS.

Standard client/server architectures do not provide a basis for this type interaction. As we will show, the AMF was specifically designed to provide it. The underlying assumption of the AMF is that none of the individual components in a transfer has sufficient information to efficiently control resource allocation and interobject synchronization. A pair of components, such as an AP and a single AIO, is also insufficient, since both end points could think that they could provide a degree of service without realizing that the network interconnect was overloaded or that other applications were about to request service. Instead, by using the information in a document specification to be able to look ahead into an application's future behavior, new techniques for resource allocation in its broadest form can be studied for each component. Unlike typical client/server models, these techniques are not based on a notion of lower-level protocol data independence, but rather on distributing control so that support decisions can be made in the light of the needs of applications throughout the network.

The scope of the AMF control activity is discussed in the following paragraphs.

The AP

The role of the AP is to supply the other components within the AMF with a specification of the object streams used by an application, as well as a definition of any interobject-stream synchronization requirements and a set of options that can be used in providing adaptable control (Sect. 3.1). The AP itself functions like the player described in Sect. 2.1: it provides a control interface to the user to provide high-level interaction with the network. ("High level" means operations like start, stop, pause, fast-forward, seek, etc.)

In terms of the issues defined in Sect. 2.2, the player provides a user interface to the execution environment, allowing the user to select the parts of a document that need to be played, to navigate through the document and to define the degree to which a document can be adapted. (For example, if a user plays a document on a disconnected portable machine, more tolerance for missing data objects may be specified). The player has only a limited role in implementing any representation or content-based control operations other than possibly supporting heterogeneous data; this is because the player is a general purpose interface, while the specification provides the other AMF components with the information necessary to adapt to the needs of the multimedia application.

The LOS

The LOS serves as a scheduling authority that controls access to I/O devices attached to the local workstation. The LOS would typically allocate resources based on its architecture-specific knowledge of the local operating environment and the document specification provided by the application. While the LOS is responsible for controlling the flow of information in and out of the local environment, including presenting information to and receiving information from the network controller(s), it cannot control activity outside of its environment because it has only a limited view of what is happening across the network. Individual sources may need to subsample or presynchronize streams within a document, or there may be other active documents generating competing requests for resources that are totally outside the scope of a local operating system.

The LOS can participate in managing various data streams for an application by implementing a negotiation process among data providers within the network. The LOS (together with the LOS of an information provider) can also be used to implement the end-to-end protocols associated with intraobject synchronization. Both of these types of service can be provided directly or in conjunction with a GOS. In general, local resource control should be as lightweight as possible; this provides the user with a responsive environment and the rest of the network with a nonintrusive element.

The GOS

The role of the GOS is to allocate resources on a network-wide basis. It has a view of network activity that is more comprehensive than the APs, the AIOs or the LOS, since it can coordinate activity among independent applications that use the central network but which originate from different workstations. The GOS can provide support that is independent of any particular workstation architecture, acting as moderator or mediator if conflicts arise. (Such a role may be more appropriate in wide area implementation than in local area networks.) Note that it would be possible for a given implementation model to combine the functions of the LOS and the GOS, although from the point of view of the framework, it is important to recognize that the functions served by both abstractions are different. The primary practical motivation for keeping the LOS and GOS separate is that workstations in a heterogeneous environment cannot be assumed to have similar local operating systems. (They will also most likely have local systems that cannot be altered or adapted to provide extended multimedia support.) The architecture of the GOS allows global concerns to be factored out of the local environment, even to the point that it is possible to design attached processor implementations supporting GOS functions (Bulterman and van Liere 1991).

The AIO

The AIO provides applications with an interface to stored, synthesized, or interactive information. In supporting access requests, the AIO separates the notions of multimedia information and multimedia information representation. In this way, AIO presents an abstract interface that is used to control access to one of several representations of a block of 'information.' For example, it can be used to substitute an audio description of a video if the user, the user's workstation, the network, or the server's host cannot support video delivery. By providing alternative representations of information, the AIO provides quality of information support rather than quality of service support. (The latter term is more appropriate for representation-dependent manipulations, while the former is more appropriate for content-based selection.) Note that the AIO does not give you something for nothing. It simply provides a general framework that needs to be filled in by data-dependent code and, if appropriate, alternative representations.

Based on the contents of an application specification, the AIO can enter a process of negotiation to provide an application with an appropriate representation of information that meets the constraints of conditions in the AP, LOS and GOS. The goal of the AMF is to allow individual implementation models to negotiate transparently; the motivation for this is that by the time a user goes through the operations necessary to select an alternative representation interactively, the resource constraints that prompted the original negotiation request could have changed. We also assume that most authors would prefer to select the alternative representations that should be used, based on the author's insight into the application domain. (Note that individual AP implementation models

may provide both types of control.) A prototype implementation of the AIO is described in Sect. 3.2.

[Earlier versions of the AMF used the "intelligent information object" (IIO) to label the component AIO. The term "intelligent" was misleading in that control decisions were not based on rule-sets or other automated means.]

3 Examples of AMF-based specification and support structures for adaptable multimedia

3.1 The CWI multimedia interchange format (CMIF)

CMIF (Bulterman et al. 1991; Hardman et al. 1993b; van Rossum et al. 1993), is a document specification that was developed to provide the basis for research into LOS, GOS and AIO support. In AMF terms, CMIF provides a description of the AIOs that are used by a particular application and any inter-AIO synchronization constraints. (Intra-AIO synchronization constraints are a property of each object and are not specified explicitly in CMIF.) Embedded within a CMIF description are the range of options that an application author will tolerate if a specified AIO representation or an inter-AIO synchronization relationship cannot be satisfied.

A CMIF specification consists of two descriptions of a document: a hierarchy description and a virtual I/O channel description (or, more simply, the channel description). (See Fig. 4.) The hierarchy is used to define the content-based relationships that exist among document AIOs. This description capitalizes on the inherent modularity of many multimedia applications by using a tree structure to partition the application's data objects. Within the tree, individual objects can be defined as occurring in parallel or sequentially. This relationship defines the implicit, coarse grain synchronization within the document. A hyperstructure is superimposed on the tree to provide nonlinear navigation control (Hardman et al. 1993a).

Where the hierarchy defines the relationships among the data objects, the channel description associates each data object with a virtual I/O channel. Each channel represents a collection of information of similar type that shares a common resource allocation policy. From the document's point of view, the channel is managed as an atomic entity, using channel-wide resource allocation attributes. [One of these attributes can be used to turn the channel on or off; other attributes are as diverse as defining font families to providing general scheduling quality of service bounds for the channel and AIO control options for adaptive data retrieval (Bulterman and Winter 1993)]. Each channel consists of a collection of event blocks, where each such block is an instance of an AIO. (Event blocks actually point to data descriptors that give the general properties of AIO. This indirection allows a given data object to be instanced multiple times in a document and separates content-based from representation-based concerns.) Objects placed on a channel can be either of a single medium (such as text or audio data) or can consist of a composite of media types. The relative placement of event blocks is a function of the coarse grained synchronization of objects defined in the hierarchy; this placement is done automatically by the

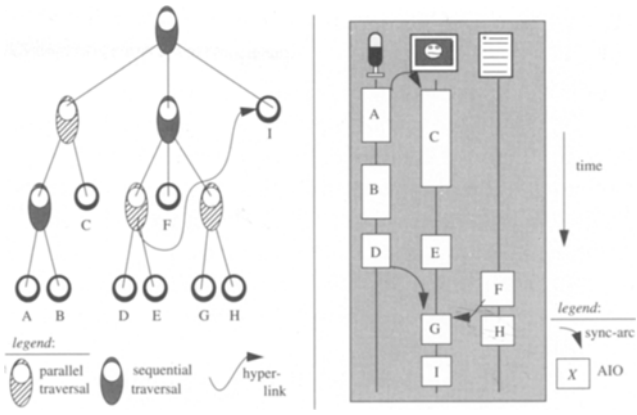

Fig. 4. CMIF in a nutshell. A CMIF specification describes a document as a *hierarchy* of components, each of which is of a specific (possibly composite) media type. The hierarchy is traversed either sequentially or in parallel. Note that hyperlink and other user interactions such as *fast forward*, *reverse*, *replay*, can change the order of tree traversal dynamically. The hierarchy in Fig. 4a is mapped to a set of *channels*, which relate the logical components to (virtual) output devices. Each logical block is placed on a channel and is associated with an AIO. Fine grain synchronization can be specified using *synchronization arcs*, shown between nodes A/C, D/G and F/G. At runtime, channels may be active or inactive, influencing document synchronization

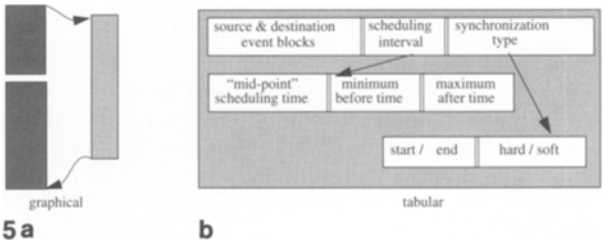

Fig. 5. The synchronization arc

authoring environment (van Rossum et al. 1993). Explicit (fine grain) synchronization among event blocks is defined by a synchronization arc (or sync arc) between the events where such synchronization is required.

Each sync arc can encode a tuple of information that is used to schedule events in the application. The major elements of the tuple, shown in Fig. 5, are:

1. The source and destination event blocks of the arc: explicitly defining these allows all of the synchronization arcs to be extracted from a document for scheduling analysis without losing context information.
2. The allowable scheduling interval of the timing arc: the requested scheduling start time, enveloped in two intervals (each of which may be zero) that indicate the flexibility given to the scheduling services to implement the synchronization request.
3. Two types of synchronization relationship: one specifying whether the synchronization relationship is relative to the beginning or the end of an event, and the other specifying if the event is hard or soft. A hard synchronization arc must be supported exactly within the synchronization in-

terval, while in a soft arc transient delays are tolerated by the application.

Information in the synchronization arc can be used for a variety of purposes. The hard/soft indication within the arc can support the notion of user flexibility, discussed in Sect. 2.2. The use of interval-based timing can allow a single document to be supported on a heterogeneous set of platforms by providing content-based tolerances among data items. In all cases, the document specification only gives the desires of the document author. Support for the document needs to be provided by other components in the AMF.

CMIF was designed to support a single specification on a wide range of systems by allowing an author to express alternatives in the manner that information is fetched by the support environment. CMIF is similar to the Harmony system (Fujikawa et al. 1991) in that both support the definition of timing relations directly between media objects, rather than defining them in relation to a time axis. CMIF provides a somewhat richer set of synchronization primitives, however, allowing the underlying support environment to adapt the data transfers in an application transparently. In this respect, CMIF also differs from systems in which specific constraints are used to derive exact timing relationships in a specification for a particular platform (Buchanan and Zellweger 1992a,b). CMIF uses its constraints to support adaptable timing relationships rather than restrictive ones. In practice, both approaches are probably useful. The more restrictive analysis can be used to determine if a document is feasible for a particular platform or for a stable environment, while our approach can be useful in executing documents of a variable nature in a more general environment.

3.2 Prototype AIOs

The AIO is an active component that manages information. As with the generic AIO in the AMF, the primary value-added advantage of the AIO is that it provides an abstraction that localizes the policy aspects of supporting synchronization and heterogeneous presentation in a content-based framework. The prototype AIO differs from that in the AMF by being based on a client/server model instead of a complete LOS/GOS infrastructure.

One example of an AIO is shown in Fig. 6. Here we see a single object containing three alternative representations of the same abstract information. One representation may be a video clip (with composite sound), the second may be an audio track and several still pictures, while the third may contain a text-based description of the information being shown, along with two captioned diagrams. In addition to data, the AIO also contains access control interfaces. The control operations can be divided into three groups:

1. Resource control – this interface allows the AIO to negotiate low-level control of the amount and type of data that is used to represent the information selected. Operations may include conventional activity, such as buffering (local or remote), subsampling, and (de)compression, or it may consist of operations where an alternative set of representations may be selected to reduce resource use, or where an object can be excluded if necessary.
2. Synchronization control – this interface allows a data-dependent synchronization action to be selected that meets the type of constraint specified by a CMIF synchronization request. Synchronization can be implemented by the control interface, but it will probably be more efficient to delegate implementation to the data support mechanism (the database or file system) and the operating system code once a policy has been negotiated.
3. Representation control – this interface allows the application to negotiate the details of the required data representation. The representation can be stored or it can be generated from a baseline representation.

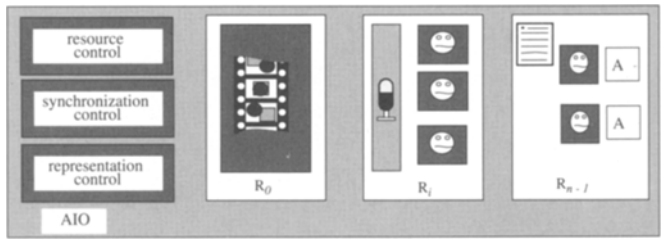

Fig. 6. The adaptive information object (AIO)

While the AIO is similar to general object-based paradigms, the focus of the interface provided by the AIO to the network environment is unique. The AIO maintains responsibility for managing a specific body of information, independent of any particular (set of) representation(s) of that information. There is an explicit set of interfaces that allow negotiated access to take place across a range of control models. These interfaces can be used during an initial access to information, where a specific representation with particular synchronization and resource characteristics could be selected, leaving the actual transfer of information to protocols and processes at the source and sink ends of a transfer. The AIO can also be used to adaptively change the representation of information based on changing characteristics of the network and or application if the object supports this functionality (Bulterman and Winter 1993).

An example of the negotiation protocol between an application and an AIO is illustrated in Fig. 7. The process starts when the player encounters a request for an image from the application. (This request is the result of referencing an AIO node within the CMIF description.) The client code on the application's hosts sends a message to the server (path a in Fig. 7) requesting a particular object, and it sends a vector of application-specified constraints (also encoded in the CMIF description, either as channel attributes or as attributes to an event block), as well as a state vector indicating resource characteristics of the client's machine (including resolution, color map depth, etc.). The server analyzes the current state of the interconnection environment plus the load on the server, and makes a first request to the AIO for a particular type of representation of the object. In performing this analysis, it mimics functions that would be performed by the LOS and/or GOS.

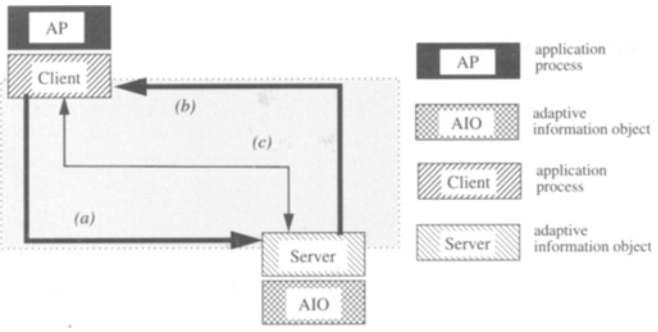

Fig. 7. The AP, AIO and surrogate client and server components

If the AIO is able to respond with the requested data object, this object is immediately returned to the application (path b in Fig. 7). (This will be the usual case.) If, on the other hand, this is not possible, the AIO may offer an alternative representation, based on the application's constraint vector. The alternative representation is also returned immediately, since the application has already implicitly given approval for receipt.

If the AIO cannot support any of the options, it returns this information as a status result that is passed back to the client (path c in Fig. 7); the client can either decide how to act on the status directly, or it can cause the application to query the user for a selection. In all of the processing, the user is kept out of the decision making activity unless the AIO and/or the application specification cannot provide an appropriate response.

Note that if the AIO passes back an alternative representation of an object (for example, a text block instead of a requested image), the client must inform the player that different resources may be necessary to support the request. To implement this, the player must realize that a picture channel may need to be replaced (temporarily) by a sound or text channel to support the mapping.

4 Current status and summary

The AMF and the two support models described in the paper are based on the assumption that resource control in a multimedia network should be adaptable, and that the adaptive process should be distributed over the application, the LOS, the (distributed) GOS, and the AIOs involved in a transfer. Each of these layers has a specific insight that is important in controlling multimedia transfers. Although each of these insights are necessary, the AMF also attempts to limit the scope of any one layer by giving each layer a specific set of concerns to process.

Support for the AMF, the CMIF specification and the AIO are ongoing research activities at CWI. Of the implementation projects, the CMIF authoring environment and its run time player is the most advanced, while support for general AIO manipulations is in an early stage. Work on the AIO is tied to the development of an LOS/GOS infrastructure and the development of semantic facilities that can be provided to support a wide range of resource, synchronization, and representation control operations. We have performed initial presentation mapping experiments (Bulterman and Winter 1993), but it is too early to draw any conclusions about the utility of this approach.

All of our activity in the Multimedia Kernel Systems Project is aimed at understanding the basic relationships that exist in supporting multiple multimedia applications in a heterogeneous network environment. In the current version of our work, this global function is replaced by a separate client and server pair. They transparently negotiate the format of the information to be used to satisfy a particular object reference based on the characteristics of the target system, the load on the network, the types of alternative representations that the client will accept, etc. This transparent interaction is important because it offers the system an opportunity to respond quickly to transient conditions in the environment, but it is difficult to achieve in the light of closed operating systems and multimedia devices. It is our long term intention to investigate the support of distributed operating systems technology that will allow CMIF specifications (or its successor) to be passed along to all of the components of the AMF, each of which will pick out the information it needs to support the synchronization and resource requirements of the application (Bulterman 1992).

Acknowledgements. The general frameworks and various implementation projects described in this article have developed during the past two years as part of the Multimedia Kernel Systems Project at CWI. Chief contributors to this project have been Guido van Rossum, Lynda Hardman, Jack Jansen, K. Sjoerd Mullender, Robert van Liere, and Dik Winter. Funding for this work has been provided by the Dutch Ministry of Education and Research, the Dutch Ministry of Economic Affairs and the Euromath Foundation.

References

Anderson TE, Bershad BN, Lazowska ED Levy HM (1992) Scheduler activations: effective kernel support for the user-level management of parallelism. ACM Transact Comput Syst 10:53–80

Anderson DP, Chan P (1991) Toolkit support for multiuser audio/video applications. Proc 2nd International Workshop on Network and Operating Systems Support for Digital Audio and Video, Heidelberg, pp 230–241

Buchanan MC, Zellweger PT (1992a) Scheduling multimedia documents using temporal constraints. Proc 3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video, San Diego, pp 237–249

Buchanan MC, Zellweger PT (1992b) Specifying temporal behavior in hypermedia documents. Proc ACM ECHT'92 Conference on hypertext, Milan, pp 262–271

Bulterman DCA (1992) Synchronization of multi-sourced multimedia data for heterogeneous target systems. Proc 3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video, San Diego, pp 119–129

Bulterman DCA, Liere R van (1991) Multimedia synchronization and unix. Proc 2nd International Workshop on Network and Operating Systems Support for Digital Audio and Video, Heidelberg, pp 108–119

Bulterman DCA, Rossum G van, Liere R van (1991) A structure for transportable, dynamic Multimedia documents. Proc Summer 1991 Usenix Conference, Nashville, pp 137–155

Bulterman DCA, Winter DT (1993) A distributed approach to retrieving JPEG pictures in portable hypermedia documents. Proc IEEE Symposium on Multimedia Technologies and Future Applications, Southampton, England pp 93–99

Clark DD, Shenker S, Zhang L (1992) Supporting real-time applications in an integrated services packet network. Proc ACM SIGCOMM'92, pp 200–208

Danthin A, Baguette Y, Leduc G, Leonard L (1992) The OSI 95 connection-mode transport service: the enhanced QoS. Proc 4th TCG WG GY IFIP Conference on High Speed Networking, pp 138–143

Ferrari D (1990) Client requirements for real-time communication services. IEEE Commun Magazine 28:65–72. See also RFC 1193, 1990

Ferrari D (1991) Design and implementation of a delay jitter control scheme for packet-switching internetworks. Proc 2nd International Workshop on Network and Operating Systems Support for Digital Audio and Video, Heidelberg, pp 72–83

Fujikawa K, Shimojo S, Matsuura T, Nishio S, Miyahara H (1991) Multimedia presentation system "Harmony" with temporal and active media. Proc USENIX Multimedia Conference, Nashville, pp 75–93

Govindan R, Anderson DP (1991) Scheduling and IPC mechanisms for continuous media. Proc Thirteenth ACM Symposium on Operating Systems Principles, pp 68–80

Hanko JG, Kuerner EM, Northcutt JD, Wall GA (1991) Workstation support for time-critical applications. Proc 2nd International Workshop on Network and Operating Systems Support for Digital Audio and Video, Heidelberg, pp 4–9

Hardman L, Bulterman DCA, Rossum G van (1993a) The Amsterdam hypermedia model: extending hypertext to real multimedia. Hypermedia Journal, 5:47–69

Hardman L, Bulterman DCA, Rossum G van (1993b) Structured multimedia authoring. Proc ACM Multimedia '93, Anaheim, Calif., pp 283–290

Hayter M, McAuley D (1991) The desk area network. Technical Report No. 228, Cambridge University Computing Laboratory, Cambridge, UK

Hodges M, Sasnett R, Ackerman M (1989) A construction set for multimedia applications. IEEE Software 6:37–43

Jeffay K, Stone DL, Talley T, Smith FD (1992) Adaptive, best-effort delivery of digital audio and video across packet-switched networks. Proc 3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video, San Diego, pp 3–14

Lesley IM, McAuley D, Mullender SJ (1993) PEGASUS-operating systems support for distributed multimedia systems. ACM Operating Syst Review 27:1, 69–78

Little TDC, Ghafoor A (1991) Scheduling of bandwidth-constrained multimedia traffic. Proc 2nd International Workshop on Network and Operating Systems Support for Digital Audio and Video, Heidelberg, pp 120–131

Director version 20, MacroMind (1990) (authoring tool for the Apple Macintosh)

Rangan PV, Vin H (1991) Designing file systems for digital video and audio. ACM Operating Syst Review 25:81–94

Rossum G van, Jansen J, Mullender KS, Bulterman DCA (1993) CMIFed: A presentation environment for portable hypermedia documents. Proc ACM Multimedia '93, Anaheim, Calif., pp 183–188

Tokuda H, Nakajima T, Rao P (1990) Real-time Mach: towards a predictable real-time system. Proc USENIX Mach Workshop, pp 213–220

Topolcic C (1990) Experimental internet stream protocol, Version 2 (ST-II). Internet RFC 1190

Verma D, Ferrari D (1990) A scheme for real-time channel establishment in wide-area networks. IEEE JSAC 8:368–379

DICK C.A. BULTERMAN received a B.A. degree from Hope College in 1973, an Sc.M. degree in Computer Science from Brown University in 1977, and a Ph.D. in Computer Science from Brown University in 1982. He was an assistant professor for engineering at Brown University from 1981 to 1988. Since 1988, he has been head of the department of Computing Systems and Telematics in Amsterdam, The Netherlands. During this period he had also held visiting and part-time professorships at the University of Utrecht, the University of Amsterdam and the Delft University of Technology. Since 1991 he has also been project leader of CWI's Multimedia Kernel System Group. His current research interests include operating systems and architecture support for multimedia systems and multimedia document modeling and authoring systems. Dr. Bulterman is a member of the ACM, the IEEE Computer Society and Sigma Xi.