Specification and Support of Adaptable Networked Multimedia

Dick C.A. Bulterman

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

E-mail: `dcab@cwi.nl`

Abstract

Accessing multimedia information in a networked environment introduces problems that don't exist when the same information is accessed locally. These problems include: competing for network resources within and across applications, synchronizing data arrivals from various sources within an application, and supporting multiple data representations across heterogeneous hosts. Often, special-purpose algorithms can be defined to deal with these problems, but these solutions usually are 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.

1 Problem Overview

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: the network provides a convenient means of distributing information to other sites, it provides access to compute servers where special-purpose processing of multimedia data can take place, and 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 when compared to accessing and manipulating multimedia data locally: the data delivery characteristics of the network are difficult to predict and control, the contention for critical system and data resources across the network makes balanced data access difficult to achieve, and differences among network hosts may make data objects difficult to share.

To be published in *Multimedia Systems*

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 [DBL92,RV91] and in operating systems by supporting *quality of service* guarantees for delivery of (possibly) complex data types [ABL92,D91,GA91,HKN91,LMM92,TNP90]. At the network level, support for predictable multimedia is provided by, among others, admission control techniques that regulate use of resources and by technologies that provide deterministic network/data access [CSZ92, HM91,JST92,LG91,VF90,T90]. 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. Here, an application running on node *a* requests multimedia data from four sources located on three separate servers. The application must know the resource requirements of each *stream* of data it uses (where we use the term "stream" to mean either a single object or a collection of similarly-typed objects from a single server), it must coordinate the arrival and manipulation of multiple independent streams, and it must take any actions necessary if a given stream cannot be provided by the infrastructure. In general, the application software must control all content-based actions in or among the streams in the context of the application, while the infrastructure will control representation-based actions within a single stream in the context of service guarantees or network/server activity.

The content/representation division of control works well in environments where sufficient resources exist to handle an application's request fully or where insufficient resources exist to handle the requests at all. It is less effective when an application can receive only partial support of its requests. 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 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

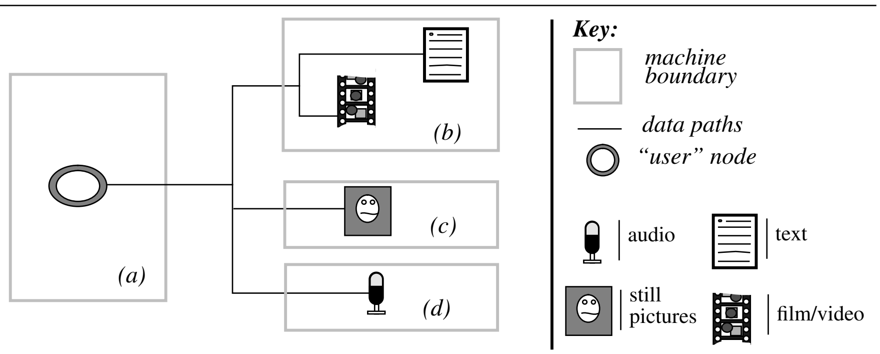


Figure 1

Simple multimedia information client/server example.
The client (a) is fed by three servers (b, c, d), one of which supplies two data types.
(The structure of the client and the client's application is not shown.)

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 do efficient resource allocation—even if this means only selecting from a set of available data streams—the application needs to know how to best make use of 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, not other streams in the infrastructure.

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 [RJM93]) can be used to generate a specification that can be accessed/executed at some 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 sections below, 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, AMF: the *Amsterdam Multimedia Framework* is presented. To put 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*.

Fig. 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 (non-embedded) 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.

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 [BZ92a]. Assuming the speci-

Figure 2 *An example multimedia application.*

The rectangles along the bottom represent navigation controls; the square in the picture is a hyper-button. The two lines of text are captions that accompany multi-lingual audio.

fication 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 [BRL91].

Creating documents using authoring systems or program-based toolkits is typically an arduous task [AC91,HSA89,M90]. One motivation for investigating adaptable networked multimedia was to provide reduce the overall effort of producing multimedia presentations by a means of reusing document structures in multiple environments once they were authored [BRL91,HBR93b].

2.2 Supporting Adaptable Documents: Data-Representation and Document-Content Issues

During analysis of a document, it is typically 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: issues associated with the physical representations of multimedia data and 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, 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:

- *intra-object 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 [D90,D91];
- *inter-object synchronization:* in general documents, data will be encoded in separate streams of objects, each of which may be located at different hosts. While inter-object synchronization is often controlled in the context an of application, the composite transfer of data may need to be coordinated to improve system efficiency. For example, synchronization of audio data and caption-text can be done by the application, but it can be done more efficiently using markers placed in the data objects and evaluated by the support software;
- *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 over 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 when allocating 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:

- *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 don't 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.)
- *presentation non-linearity:* the order in which objects are accessed and presented depends on the document structure *and* the result of user interaction at run-time. For example, 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; in Fig. 2, a small rectangle is visible over a traffic sign in the mid-right 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.
- *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 non-linearity* 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 AMF: The Amsterdam Multimedia Framework

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.

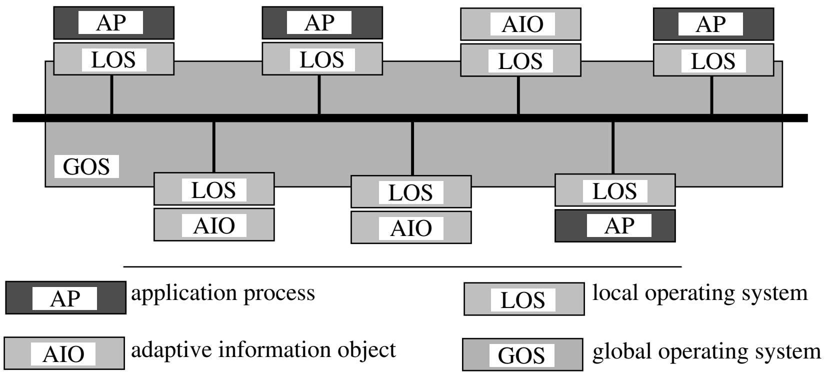


Figure 3 *AMF "active" components.*

Fig. 3 illustrates AMF. Here, many applications (AP) communicate with *adaptive information objects* (AIOs) via an infrastructure that is managed by a set of local operating systems and a global operating system. The LOSs and GOS coordinate resource allocation, while the APs and AIOs request and deliver information, respectively. Note that 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 still need to be developed that implement the general functionality of the framework.

The general structure of AMF is similar to client/server models of networked computing. The difference between AMF and these models is that within 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 AIO 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:

- each of the AIOs was aware of the other's presence,
- 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, and
- 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, 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 inter-object synchronization. A pair of components, such as an AP and a single AIO, is also insufficient, since both end-points could *think* 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 light of the needs of applications throughout the network. The scope of AMF control activity is discussed in the following paragraphs.

The application process (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 inter-object-stream synchronization requirements and a set of options that can be used in providing adaptable control (see section 3.1 for an example). The AP itself functions like the *player* described in section 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 section 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 object 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 local operating system (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 has the responsibility 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 sub-sample or pre-synchronize 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 intra-object 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 light-weight as possible; this provides the user with a responsive environment and the rest of the network with a non-intrusive element.

The global operating system (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 abstrac-

tions 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 [BL91].

Adaptive information objects (AIO).
The AIO[1] 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.

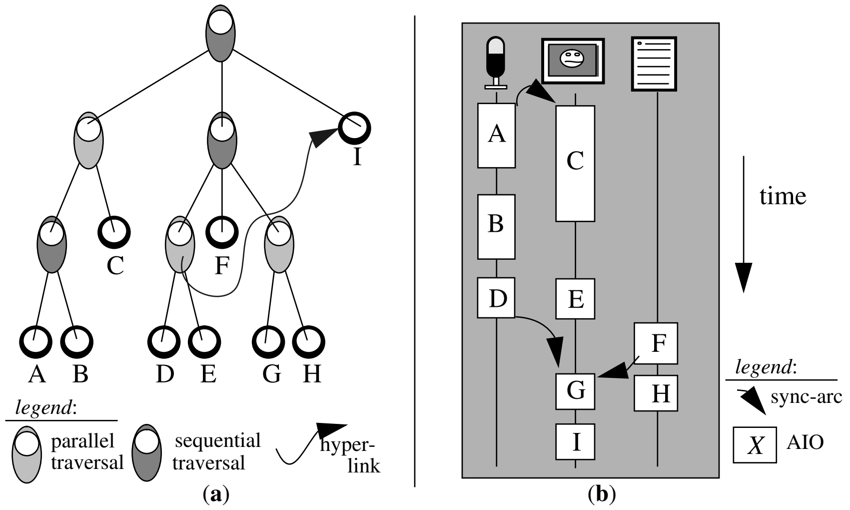
Based on the contents of an application specification, the AIO can enter a process of negotiation to provide an application with an appropriate representation of information that meets the constraints of conditions in the AP, LOS and GOS. The goal of the AMF is that individual implementation models do this negotiation transparently; the motivation for this is that by the time a user goes through the operations necessary to interactively select an alternative representation, 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 which 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 section 3.2.

3 Examples of AMF-Based Specification and Support Structures for Adaptable Multimedia

3.1 CMIF: The CWI Multimedia Interchange Format

CMIF: the CWI Multimedia Interchange Format [BRL91,RJM93,HBR93b], 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.

1. Earlier version of AMF used the notation IIO (or, Intelligent Information Object) to label this component. The term 'intelligent' was misleading in that control decisions were not based on rule-sets or other automated means.



a: 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 hyperlinks and other user interactions such as *fast forward, reverse, replay*, can change the order of tree traversal dynamically.

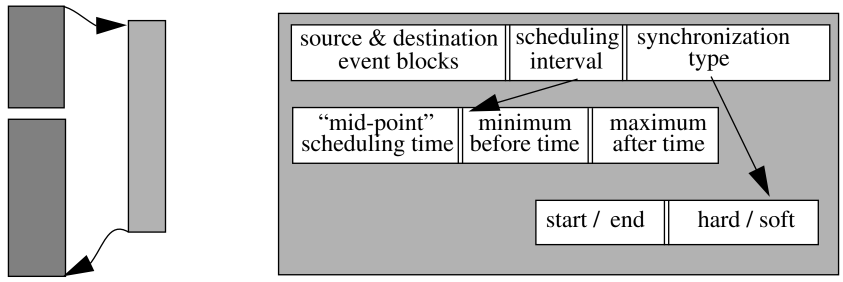
b: The hierarchy in (a) is mapped to a set of *channels,* which relate the logical components to (virtual) output devices. Each logical block is placed on a channel and is associated with an AIO. Fine-grain synchronization can be specified using *synchronization arcs*, shown between nodes A/C, D/G and F/G. At run-time, channels may be active or inactive, influencing document synchronization.

Figure 4 *CMIF in a nutshell.*

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-grained synchronization within the document. A hyperstructure is superimposed on top of the tree to provide non-linear navigation control [HBR93a].

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 similarly-typed information 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 [BW93]). Each channel consists of a collection of *event blocks*, where each such block is an instance of an AIO.[2] Objects placed on a

2. Event blocks actually point to *data descriptors*; these descriptors 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.



(a) graphical (b) tabular

Figure 5 *The Synchronization Arc*

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-grain synchronization of objects defined in the hierarchy; this placement is done automatically by the authoring environment [RJM93]. Explicit (fine-grained) 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:

- *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;
- *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;
- *two types of synchronization relationship*: one specifying if the synchronization relationship is relative to the beginning or end of an event and the other specifying if the event is *hard* or *soft.* A hard synchronization arc is something that must be supported exactly within the synchronization interval, while a soft arc is one in which 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 Section 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 [FSM91] 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 transparently adapt the data transfers in an application. In this respect, CMIF also differs from systems in which constraints are specified that are then used to derive exact timing relationships in a specification for a particular platform [BZ93a,BZ93b]. 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 Adaptive Information Objects

The AIO is an active component that manages information. As with the generic AIO in AMF, the primary value-added benefit 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 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:

- *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;
- *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.
- *representation control*—this interface allows the application to negotiate with the required data representation to another. The representation can be stored or it can be generated from a baseline representation.

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 are an explicit set of interfaces that allow negotiated access to take place across a range of control models. These interfaces

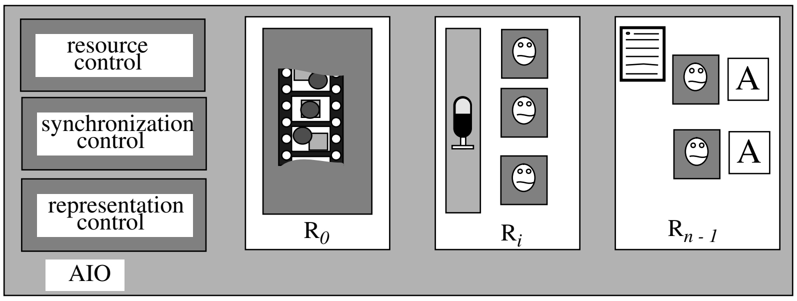


Figure 6 *The adaptive information object (AIO).*

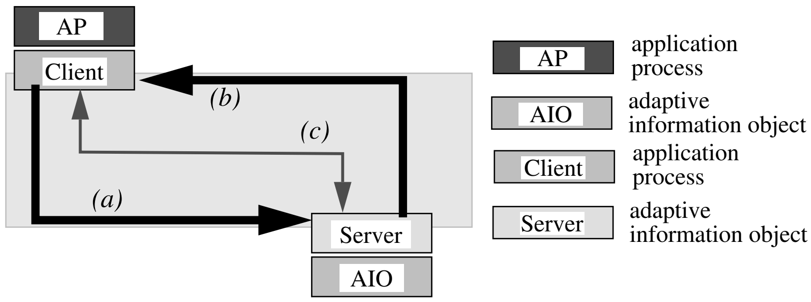


Figure 7 *The AP, AIO and surrogate client and server components.*

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 [BW93].

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.) Client code on the application's hosts sends a message to the server *(a)* requesting a particular object, plus 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. If the AIO is able to respond with the requested data object, this object is immediately returned to the application *(b)*. (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 *(c)*; 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 (temporarily) be replaced 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, all 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 local operating system, the global (distributed) operation system 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, AMF also attempts to limit the scope of any one layer by giving each layer a specific set of concerns to process.

Support for AMF, the CMIF specification and the AIO are on-going 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 at 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 [BW93], but it is too early to draw any conclusions on 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 that 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 an opportunity for the system 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 get passed among 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 [B92].

Acknowledgments

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 through the Euromath Foundation.

References

[ABL92] Anderson, T.E., B.N. Bershad, E.D. Lazowska, and H.M. Levy, "Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism," *ACM Transactions on Computer Systems*, 10(1), February 1992.

[AC91] Anderson, D.P. and P. Chan, "Toolkit Support for Multiuser Audio/Video Applications," *Proc. 2nd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* Heidelberg, Nov. 1991.

[B92] Bulterman, D.C.A., "Synchronization of Multi-Sourced Multimedia Data for Heterogeneous Target Systems," *Proc. 3rd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* San Diego, Nov. 1992.

[BL91] Bulterman, D.C.A. and R. van Liere: "Multimedia Synchronization and Unix," *Proc. 2nd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* Heidelberg, Nov. 1991.

[BRL91] Bulterman, D.C.A., G. van Rossum and R. van Liere: "A Structure for Transportable, Dynamic Multimedia Documents," *Proc. Summer 1991 Usenix Conference*, Nashville TN, June 1991.

[BW93] Bulterman, D.C.A. and D. T. Winter, "A Distributed Approach to Retrieving JPEG Pictures in Portable Hypermedia Documents," *Proc. IEEE Symp. on Multimedia Technologies and Future Applications*, Southampton, UK, April 1993.

[BZ92a] Buchanan, M.C. and P. T. Zellweger, "Scheduling Multimedia Documents Using Temporal Constraints", *Proc. 3rd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* San Diego, CA, Nov. 1992.

[BZ92b] Buchanan, M.C. and P. T Zellweger, "Specifying Temporal Behavior in Hypermedia Documents", *Proceedings of the ACM ECHT'92 Conference on Hypertext*, Milano, Italy Nov 30 - Dec 4 1992.

[CSZ92] Clark, D.D., S. Shenker, and L. Zhang. "Supporting Real-Time Applications in an Integrated Services Packet Network", in *Proceedings of ACM SIGCOMM'92*, 1992.

[D90] Ferrari, D. "Client Requirements for Real-Time Communication Services", *IEEE Communications Magazine*, November 1990. See also RFC 1193, November, 1990.

[D91] Ferrari, D, "Design and Implementation of a Delay Jitter Control Scheme for Packet-Switching Internetworks," *Proc. 2nd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* Heidelberg, Nov. 1991.

[DBL92] Danthin, A., Y. Baguette, G. Leduc, and L. Leonard, "The OSI 95 Connection-mode Transport Service: The Enhanced QoS," in *Proceedings of 4th IFIP Conference on High Speed networking*, Dec. 1992, Liege, Belgium.

[FSM91] Fujikawa, K., S. Shimojo, T. Matsuura, S. Nishio and H. Miyahara, "Multimedia Presentation System 'Harmony' with Temporal and Active Media", *Proc. USENIX Multimedia Conference*, June 1991 Nashville TN.

[GA91] Govindan, R. and D.P. Anderson. "Scheduling and IPC Mechanisms for Continuous Media," in *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, October 1991.

[HBR93a] Hardman, L., D.C.A. Bulterman, and G. van Rossum, "The Amsterdam Hypermedia Model: Extending Hypertext to Real Multimedia," *Hypermedia Journal*, Vol 5(1), May 1993

[HBR93b] Hardman, L., D.C.A Bulterman and G. van Rossum, "Structured Multimedia Authoring," *Proc. ACM Multimedia '93*, Anaheim CA, August 1993.

[HKN91] Hanko, Kuerner, Northcutt, Wall, "Workstation Support for Time-Critical Applications," *Proc. 2nd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* Heidelberg, Nov. 1991.

[HM91] Hayter, M. and D. McAuley, "The Desk Area Network," *Technical Report No. 228*, Cambridge University Computing Laboratory, 1991.

[HSA89] Hodges, Sasnett, and Ackerman, "A Construction Set for Multimedia Applications," *IEEE Software,* 6 (1), Jan. 1989.

[JST92] Jeffay, K. D.L. Stone T. Talley, and F.D. Smith, "Adaptive, Best-Effort Delivery of Digital Audio and Video Across Packet-Switched Networks," *Proc. 3rd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* San Diego, CA, Nov. 1992.

[LG91] Little, T.D.C and A. Ghafoor, "Scheduling of Bandwidth-Constrained Multimedia Traffic," *Proc. 2nd Int. Workshop on Network and Operating Systems Support for Digital Audio and Video,* Heidelberg, Nov. 1991.

[LMM92] Lesley, I.M., D. McAuley, and S.J. Mullender, "PEGASUS - Operating Systems Support for Distributed Multimedia Systems," *ACM Operating Systems Review* 1(27), January, 1992

[M90] Director version 2.0, MacroMind 1990 (authoring tool for the Apple Macintosh).

[RV91] Rangan, P.V. and H. Vin, "Designing File Systems for Digital Video and Audio," *ACM Operating Systems Review*, 25 (5), 1991.

[RJM93] van Rossum, G., J. Jansen, K.S. Mullender and D.C.A. Bulterman, "CMIFed: A Presentation Environment for Portable Hypermedia Documents," *Proc. ACM Multimedia '93*, Anaheim CA, August 1993.

[T90] Topolcic, C., "Experimental Internet Stream Protocol, Version 2 (ST-II), *Internet RFC 1190*, Oct. 1990.

[TNP90] Tokuda, H., T. Nakajima, and P. Rao, "Real-time Mach: Towards a Predictable Real-Time System," in *Proceedings of USENIX Mach Workshop*, October 1990.

[VF90] Verma, D. and D. Ferrari. "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", *IEEE JSAC*, Vol. 8, No. 3, April 1990.