Synchronization of Multi-Sourced Multimedia Data for Heterogeneous Target Systems

Dick C.A. Bulterman

CWI: Centrum voor Wiskunde en Informatica
Postbus 1009, 1009 AB Amsterdam, The Netherlands

Abstract. Accessing multimedia information in a networked environment introduces problems to an application designer that don't exist when the same information is fetched locally. These problems include "competing" for the allocation of network resources across applications, synchronizing data arrivals from various sources within an application, and supporting multiple data representations across heterogeneous hosts. In this paper, we present a general framework for addressing these problems that is based on the assumption that time-sensitive data can only be controlled by having the application, the operating system(s) and a set of active, intelligent information object coordinate their activities based on an explicit specification of resource, synchronization, and representation information. After presenting the general framework, we describe a document specification structure and two active system components that cooperatively provide support for synchronization and data-transformation problems in a networked multimedia environment.

1 Problem Overview

The focus of much of the activity associated with multimedia computing has been on the development of interfaces and software that allow various types of time-based data to be manipulated on a PC or workstation. An obvious evolution of this work is to extend it across a network infrastructure. Such *networked multimedia* offers a number of immediate practical advantages to users: 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 last (but not least!) it provides access to file servers that can be used to store the often vast amounts of data required to represent even the most simple multimedia information fragment. Unfortunately, networking multimedia activity also introduces three fundamental problems that don't exist for local-host multimedia; these are network-related *synchronization control*, *resource allocation* and *heterogeneous systems* support. Synchronization control problems are caused by network interfaces, network infrastructures, and remote hosts, all of which introduce various delays while fetching data that are not always obvious to the application requesting that data. Resource allocation problems stem from the fact that the network and its servers are resources that must be shared among various independent applications that have no idea of global network activity. Finally, heterogeneity problems result from a network in which member systems use different underlying data representations or—even worse— on which different underlying multimedia support facilities exist. Taken together, these problems mean that applications which run effectively locally will probably 'break' as soon as they are networked.

While the benefits and problems associated with networked multimedia are not new, the time- and space-sensitive nature of multimedia information provides a new set of constraints that existing infrastructures are not able to handle. This is because the traditional network and operating systems support mechanisms that manage data synchronization and resource allocation do not get enough information from standard I/O requests to intelligently support the resource and data-dependent synchronization needs of multimedia data.

In this paper, we consider the partitioning of control concerns for supporting network-wide synchronization of data in a heterogeneous environment. Synchronization is an important issue because it lies at the heart of allocating resources across the network. Heterogeneous networks (that is, networks in which resource allocation depends not only on the characteristics of the multimedia data, but also on the nature of the systems involved in a multimedia data exchange) are also important because they force a consideration of general techniques rather than any special-purpose solutions that can result from using the particular characteristics of any one support platform.

Our discussion of synchronization control is divided into two parts. We begin by describing a network-wide framework for managing multimedia data manipulation. The purpose of this framework is to clearly partition the synchronization control responsibilities among the "active components" in a multimedia system. We then present a specific synchronization control-sharing model that is based on this framework. This model uses an application-generated multimedia activity specification that is shared by the application, the operating system(s) and a set of intelligent data objects to coordinate activity across the network. As will be shown, the key to our approach is the assertion that the manipulation of multimedia data needs to be controlled by intelligent cooperation among components involved in defining, requesting, supplying and transmitting data; as a result, control issues can not be concentrated within a single layer (as is the case in current operating systems), but it must be shared across layers.

In the sections below, we start by presenting CWI's Amsterdam Multimedia Framework (AMF). We then present our current approaches to partitioning information sharing control among the application, the operating system(s) and an intelligent data storage component. We conclude with a list of problems that need to be solved before true heterogeneous support can be provided for a wide class of multimedia applications.

2 The Amsterdam Multimedia Framework

AMF was developed to organize our study of the problems associated with resource allocation and data synchronization in heterogeneous multimedia networks. This clarification is required because multimedia support has developed on a more-or-less ad hoc basis. For example, since multimedia support has partially evolved from application-based uses of relatively standard I/O devices (such as an audio interface), the applications layer has played a major role in selecting, manipulating and controlling data transfers. On the other hand, operating systems have had the historical responsibility of controlling I/O flow in a computing system. The OS allocates transfer buffers, schedules I/O operations (assigning priorities among applications and devices) and monitors the actual transfer across the system's resources. In addition to the application and the OS, I/O device controllers also play a major control role in managing data transfers. These controllers can consist of multimedia device interfaces or interfaces that control intermediate data transfer channels, such as interconnection networks.

Rather than assign control responsibility to any one layer, AMF consists of a set of active components that cooperate in the control of networked data/information flow. It also defines the "scope of responsibility" that each component can exert before and during the transfer of multimedia information. AMF gives us a foundation upon which to build individual models that address particular synchronization and/or resource control issues.

2.1 Active Multimedia Control Components in AMF

There are four active control components defined within AMF. These are: the application process, a local operating system environment (including I/O controllers), a global operating system and a collection of intelligent information objects. (See figure 1.) The nature of each of these components and their roles with respect to the control of information in a multimedia network is considered in the following paragraphs.

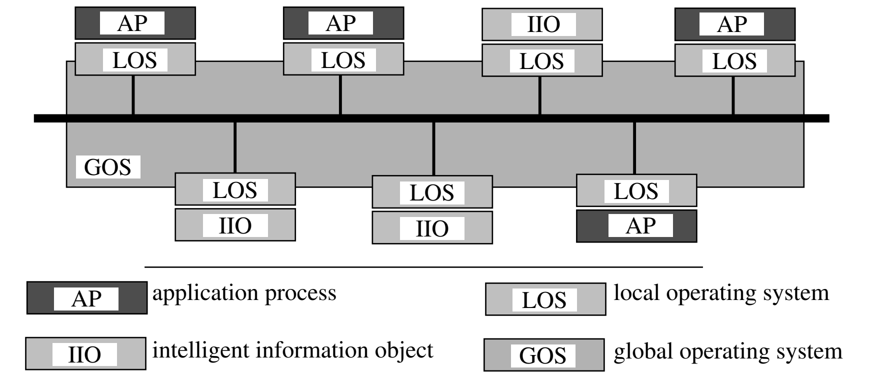


Fig. 1. AMF "active" components.

The application process. The application generates a description of the information objects that will be used (including their location), as well as a definition of how these objects are to be synchronized from the application's perspective. It also 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.—the type of operations that one might expect on, say, a CD-player or home video control panel.) The main control role of the application is to specify the relationships among the information elements and to control the user's production and consumption of information. Given the high performance cost of involving the application at run-time, however, it has only a limited role in *implementing* any dynamic control operations.

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 locally based on its knowledge of the local operating environment and an applications specification defined by the application. The LOS has the responsibility for controlling the flow of information into and out of the local environment—including presenting information to and receiving information from the network controller(s)—but it does not participate in global scheduling decisions.

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 application or the LOS, since it can coordinate activity among independent applications that use the central network but which do interact with all local workstations. As we will describe below, we feel that this is a particularly interesting domain to study distributed operating systems support. 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.

Intelligent Information Objects (IIO). The IIO is an entity that provides information to applications. It includes the information plus a series of operations on that information. In supporting access requests, it separates the notions of *multimedia information* and *multimedia information representation*. The IIO presents an abstract interface that is used to control access to one of several representations that may exist to implement a block of 'information.' This interface can be used by the application, the LOS or the GOS to select an appropriate representation based on static or dynamic needs. Note that the IIO 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. Note also that there is nothing in the IIO model that requires information to be persistent or static; it may represent a program that generates information on demand, or it may represent an interactive user.

2.2 Interactions Among Components

Within AMF, the control of multimedia is a cooperative process that requires coordination among all components. Fig. 2 illustrates this interaction. At the applications level,

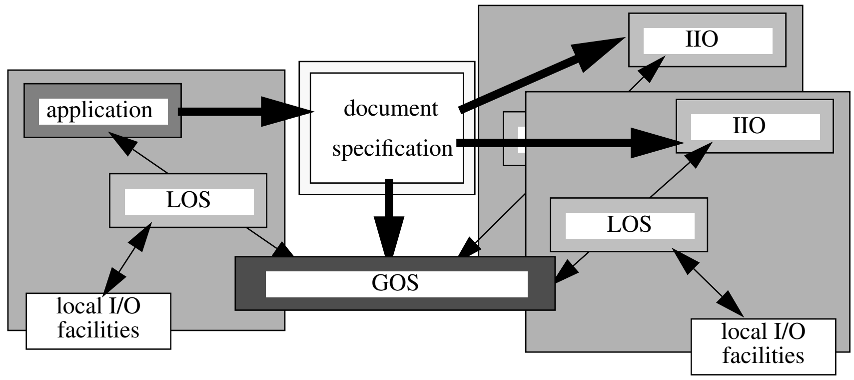


Fig. 2. The document specification.

a *document specification* is generated that describes the logical and physical interactions that are required by the application. The nature of this specification depends on the underlying support model, but its essence is that each application must be able to define the relationships that exist among components with respect to their synchronization needs, their expected resource needs, and their priority in the application. At the applications level, the document specification functions like a programming language that can be used to specify the behavior of the application. At the LOS and GOS levels, it can be used as an I/O specification and as a resource map. (Note that the GOS will have one specification available for each concurrently running application; it can use these for global resource allocation.) At the IIO, the specification can be used to determine when information objects need to be activated, as well as a guide to how the flow of information can be controlled at run-time. In general, the document specification will describe component interactions, but not component data. (This is because data is an active, not passive, component.) Clearly, the more information that is placed in the document specification, the more extensive the level of support that can be provided.

One way of using the document specification is as a guide to resource allocation during the lifetime of an application. Another way is to determine how (and if) the synchronization needs of the application are to be supported at run-time. In both cases, the processing of the document specification could lead to one of three support decisions being made:

- If the support request does not conflict with the present state of the system, a positive support action could be taken by the appropriate level (either the application, the LOS, the GOS or the IIO). This type of support is termed *trivial accept*.
- If the support request produces a fundamental conflict with the present state of the system, a negative support action could be taken by the appropriate level (either the application, the LOS, the GOS or the IIO). This type of support is termed *trivial reject.* (Trivial reject allows for either partial or full rejection.)
- If the support request results in a conflict that is not fundamental, a (light-weight!) process of negotiation could be started to see if support could be offered at a reduced level of service. This type of support is termed *negotiated accept.*

The support for *trivial accept*, *trivial reject* and *negotiated accept* is a consistent theme throughout the AMF. Note that the framework does not dictate any particular means of specifying the activity within a document or the way in which resource decisions are made. Instead, it presents a number of guiding principles that allow individual implementation models to be constructed.

3 Specification of Synchronization in a Heterogeneous Network

In the sections below, we illustrate how synchronization of multi-sourced multimedia information can be supported in a heterogeneous environment. The approach we use is based on AMF; it illustrates how the application, the LOS, the GOS and various IIOs can be architected together to partition and coordinate control activity. In order to put our approach in context, we preface this discussion with a description of the environment we base our research on and its impact on the types of models we investigate in supporting a network of heterogeneous hosts.

3.1 Environmental Impact Statement

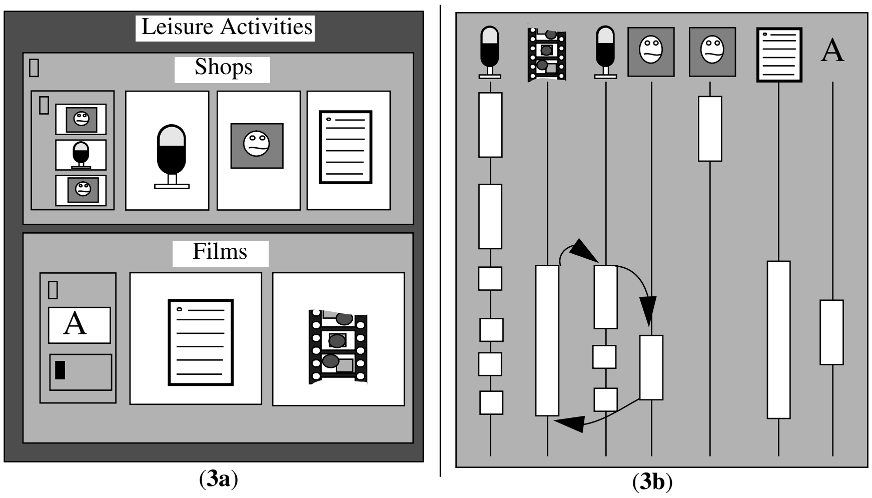
The environment used to drive our research consists of a network of heterogeneous host systems that (simultaneously) share one or more multimedia documents with each other. This environment provides the following problems that must be addressed for effective multimedia information sharing:

a) The interconnect bandwidth is a critical resource that must be allocated in an effective manner.
b) Each application is described in terms of a collection of data objects that must be implicitly or explicitly synchronized with other objects. (Synchronization information within an object is specified by the IIO implementing the object.) Each object may consist of composite or single-media data. The information associated with any one object always exists at a single source host, but the collection of objects in a document may be spread across the network. (This means that there may be multiple delays in implementing a document due to the different delay characteristics of each participating source host.)
c) At present, we restrict ourselves to a single destination host for each document, but not a single destination host type. That is, we assume that each destination processes its own copy of a document that is independent of any other documents (or any other instances of the same document) in the network.
d) At present, we restrict ourselves to "static" multimedia documents. That is, we consider only documents whose synchronization specifications have been predefined. (For example, we consider "multimedia mail" applications, in which a message is preformatted, but not "multimedia talk" applications, where data is generated on the fly.)

As a consequence of characteristic (a), we use a GOS model that is based on a distributed operating system that is functionally separate from the each LOS at a user or server site. The GOS can be tailored to provide speciality services for resource allocation and synchronization support. As a consequence of (b), the GOS schedules activity in terms of a collection of applications that each may (independently) request one or more data objects simultaneously. As a consequence of (c), each IIO must be able to support data translation operations for the data it "owns" and it must be able to communicate with the application, LOS and GOS to implement demand scheduling requirements. As a consequence of (d), the LOS and GOS can make scheduling decisions that "only" need to adapt to the data and transfer characteristics of the application and the environment; these decisions need to be adaptive, but they can be based on prior knowledge of the document set being processed. Other dynamic concerns (such as instantaneous production of data or the processing of hyperdata jumps) are not considered here.

3.2 CMIF: The CWI Synchronization Specification

The approach we take to defining synchronization is to provide a user with a general document specification model (called CMIF—the CWI Multimedia Interchange Format [2]) that provides a method for collecting data objects into a multimedia document. A CMIF specification consists of two major views of a document: a *hierarchical view* and the *virtual I/O channel view* (or, more simply, the *channel view*). The hierarchial view is used to define the content-based relationship among data objects (that is, it defines which objects are to be presented in parallel and which serially); this relationship defines the implicit synchronization of objects in the document. The channel view maps objects in the hierarchy onto a collection of virtual channels, where each channel represents a collection of similarly-typed information that shares a common resource allocation policy. (See Fig.3.) The channel view consists of a collection of



(**3a**) (**3b**)

3a: The *hierarchy view* shows the logical components of a document and how they interact. One block (Leisure Activities) is composed of two sub-blocks; nested blocks are shown by 'nesting rectangles' in the upper corner. This view shows presentation structure, which supplies implicit synchronization information.

3b: The *channel view* maps the logical components to virtual output devices. Each logical block is placed on a channel and is associated with an IIO. The white boxes are self-synchronizing event descriptors. (Note: see figure 5 for a discussion of *synchronization arcs*.) At run-time, channels may be active or inactive, influencing document synchronization.

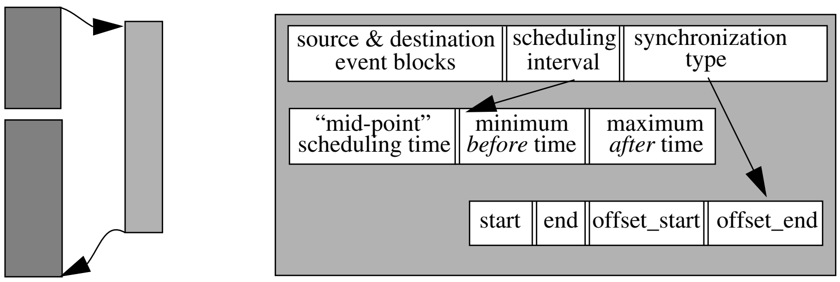
Fig. 3. CMIF in a nutshell.

event blocks, where each such block is an instance of a data object. Synchronization within a channel can be specified implicitly or explicitly. Implicit synchronization is typically defined by the relationship in the hierarchy view. Explicit synchronization is usually used between event blocks in separate channels; it allows a fine-grained relationship among events in the specification to be defined. Information within an event is usually self-synchronizing, as a result of the underlying data characteristics in the IIO. Explicit inter-event synchronization is required to match content relationships across events or to support presentation on heterogeneous hosts.

When supporting implicit synchronization, the general properties of channels can be used by the LOS for scheduling local I/O activity and by the GOS to schedule interaction among channels. They can also be used by the IIO to determine the filtering (if any) that needs to occur in presenting the data.

When using explicit synchronization, all of the ‘clues’ available for implicit synchronization exist plus information from explicit synchronization primitives called *synchronization arcs*. (The name comes from the graphical representation used to define relationships among events in different channels; see Fig. 4a.) Each synchronization arc can encode a tuple of information that is used to schedule events in the network. The elements of the tuple, shown in Fig. 4b, are:

a) the source and destination event blocks of the arc;
b) the allowable scheduling interval of the timing arc, containing:
   - the “mid-point” scheduling time as an offset from a starting time;
   - the minimum acceptable start time *before* the mid-point time;
   - the minimum acceptable start time *after* the mid-point time;
c) the type of synchronization relationship.



(a) graphical (b) tabular

Fig. 4. The Synchronization Arc

Information in the synchronization arc is used at each level for different purposes. The GOS, for example, when sensing a synchronization relationship between two events that cannot be supported, can notify the relevant IIOs to stop or retard data transmission. It could also interact with the application, although the cost of doing so may be too high (in terms of scheduling and communications time) to justify this for short-term transient problems.

In addition to synchronization, the channel view also serves as a guide to LOS and GOS resource control. Since each channel is a virtual device, not a real one, similar data types may be specified several times. (For example, there may be two audio channels defined, even if there is only one speaker.) Each time a channel is defined, a different resource allocation policy can be specified for all of its event blocks. This policy can then be passed to the IIO when the corresponding data object is accessed. (If desired, override policies could also be specified with each event block.)

CMIF can be used to provide the application’s view of how information should be gathered and presented. These rules are defined in a manner that gives the underlying layers such as the LOS and the GOS a basis for allocating resources and for fetching information out of objects. The CMIF document could be precompiled for static resource allocation, or it could be interactively interpreted by a *player* to provide an application with run-time control.

3.3 Interactive Intelligent Information Objects

The IIO is an active component that manages information. The general structure of the IIO is shown in figure 5. Here we see a single object containing three alternative repre-

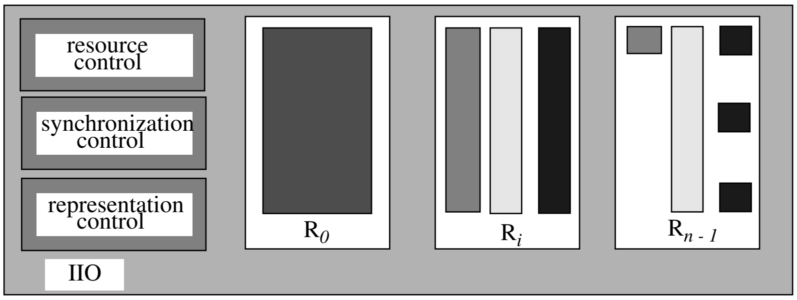


Fig. 5. The intelligent information object (IIO).

sentations; 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 three diagrams and a caption. In all cases, each representation provides the same information, albeit in a different form.

In addition to data, the IIO also contains access control information. Examples of access control may be to support resource allocation (that is, if there is only limited bandwidth available on the interconnect, the IIO select a lean information representation), or heterogeneous mappings (from one display format to another), or synchronization control (as explained above). The control operations provided by the IIO can be divided into three groups:

(1) *resource control*—this interface allows the IIO to control 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), sub-sampling, and compression, or it may consist of operations such as *aliasing* (where an alternative set of representations may be selected to reduce resource use), *preemption* (where an object can be temporarily suspended), or *optional* (where an object can be totally excluded if necessary)

(2) *synchronization control*—this interface allows the IIO to select a data-dependent synchronization action to meets the type of constraint specified by an explicit or implicit CMIF synchronization request. Policies may include *skipping*, *sub-sampling* and *forced-delay.*

(3) *representation control*—this interface allows the IIO to map data from one representation to another. It is primarily used for supporting heterogeneous activity in the network. (The use of alternative representations for resource control is discussed above.)

As was noted above, there is nothing inherent in the IIO that would restrict it to supporting static data: the IIO could be an interface into a suite of programs to dynamically generate information representations. The primary value-added benefit of the IIO is that it localizes the policy aspects of supporting synchronization and heterogeneous presentation policies. In providing this support, we delegate any higher-level decision making on the interaction among IIOs to the GOS, LOS or even application. This promotes the notion of a light-weight information serving object that can be tailored to the content of the information it manages.

The IIO shares many aspects of a (distributed) database object model. While the database metaphor is useful, the IIO can also be defined in terms of a more conventional file system architecture. In both cases, the interface into the IIO should be fast enough to allow efficient communication with the GOS (and LOS) layer(s).

4 Current Status and Summary

The AMF and the various support models described, all are based on the assumption that resource control in a multimedia network should be adaptive, and that the adaptive process should be distributed over the application, the local operating system, the global (distributed) operation system and the IIOs 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 IIO are on-going research activities at CWI. At present, we have developed an authoring environment and a run-time player to construct and manipulate CMIF-based multimedia documents in a single-host environment. The CMIF player relies on a commercially-supported LOS, which is a trend that we plan to continue. (The workstation also has to be used for general-purpose work!)

As was reported in [1], we are involved in an on-going project to investigate the development of distributed operating systems at the GOS level that are modelled on a *multimedia co-processor* concept (MmCP). The MmCP acts like an intelligent device controller that can provide distributed resource control operations. It uses the CMIF specification as input, and interacts with other active components to implement the transfer needs of simultaneous, independent multimedia activity.

Support for the IIO is at an early stage. Our present concern is the analysis of the semantic facilities that can be provided to support a wide range of resource, synchronization, and representation control operations. Support for an implementation environment for the IIO will be phased in during the coming year.

All of the 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. While the main thrust of this work has been sketched in this article, we are also interested in understanding the impact of interactive generation of CMIF documents for particular applications areas. Our work in this area has begun with the study of user-level hyperdata structures on multimedia systems [3].

References

[1] Bulterman, D.C.A. and R. van Liere: "Multimedia Synchronization and Unix," Proceedings of the 2nd International Workshop on Network and Operating Systems Support for Digital Audio and Video, Heidelberg, November 1991.

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

[3] Hardman, L., D.C.A. Bulterman, and G. van Rossum, "The Amsterdam Hypermedia Model: Extending Hypertext to Real Multimedia," CWI Technical Report (July 10, 1992).