Technical Report

UCAM-CL-TR-295
ISSN 1476-2986

Number 295

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

OPERA

Storage and presentation support for multimedia applications in a distributed, ATM network environment

Jean Bacon, John Bates, Sai Lai Lo, Ken Moody

April 1993

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

OPERA

Storage and Presentation Support for Multimedia Applications in a Distributed, ATM Network Environment

Jean Bacon, John Bates, Sai Lai Lo, Ken Moody

University of Cambridge Computer Laboratory

Abstract

We are building a display platform for multimedia applications above a multi-service storage architecture (MSSA). This style of application needs high speed ATM networks and suitable protocols and operating systems; an infrastructure which exists at the University of Cambridge Computer Laboratory.

An **open** storage architecture gives flexibility and extensibility. Conventional files, audio, video and structured objects are supported within a common architectural framework and composite objects, such as a display representation, may have components of any of these storage types. The two-level hierarchy of servers provides storage media and a byte segment abstraction at the low level and a variety of abstractions at the high level. Quality of service guarantees, which are essential for continuous media file types, are supported by *sessions* and *tickets*. These are arranged via the high level servers and used directly with the low level servers.

A platform for the creation and interactive display of multimedia presentations (IMP) is being developed. A script language allows a multimedia presentation to be specified in terms of objects, the relationships between them and the (composite) events that drive it. Presentation data is stored on the structured data service of MSSA and component objects are stored on appropriate servers, and accepted and retrieved at guaranteed rates. The presentation requirements of an application are managed by applying a script to the data representing the presentation to create a display.

1. Introduction

We have designed, and are implementing, a quality of service architecture to support multimedia applications. This style of application needs high speed, ATM networks and suitable protocols and operating systems. Our work has built on such an infrastructure, see Chapter 22 of [Bacon 93], to provide storage and presentation support for multimedia applications.

We have built storage services which are capable of storing and delivering objects of various media types (such as video, audio and text) but also structured objects. On these services we have built a minimal, active, multimedia database. At the highest level is a general and flexible platform to support the composition and display of

multimedia presentations. A related project at the Laboratory has developed support for reliable programming of distributed, composite objects in this environment [Wu et al., 93]. An overview of the project is given in [Moody et al. 93].

In this paper we outline our storage service architecture, emphasising its support for presentation and multimedia data. The IMP system is then described in some detail; data representation, IMP language features and the services and mechanisms which are needed to make the system components work together. A description of our current implementation follows and we conclude with a discussion of the issues highlighted by this ongoing research.

2. The storage architecture

Our storage architecture comprises an open, two-level hierarchy of servers.

- The servers at the low level, the *byte segment custodes* (BSCs), manage storage media of any type, support a common byte segment abstraction and provide quality of service guarantees for acceptance and delivery of data. The architecture is such that the special purpose servers at the high level, for example, those for video, audio, structured objects or conventional files, all employ the low level BSCs for storage.
- The servers at the high level provide storage abstractions for various types of stored object; for example, we have a *byte stream file custode* for conventional files, a *structured file custode* (SFC) and a *continuous media file custode.* There may be multiple instances of each type of custode.
- A server at the low level (a BSC) can be involved directly with data transfers from or to a client (a workstation window, for example). Such transfers take place within a *session* which is organised through one of the high level custodes. The client acquires a number of *tickets* for use with the appropriate BSCs to achieve delivery of the files needed in a display.

Each stored object has a storage service identifier (SSID) which is a principal specific capability [Gong 89]. Location of objects in the distributed system is managed through container abstractions at two levels. A given logical container is managed by a specific file custode instance. A given physical container is managed by a specific byte segment custode. Details are given in [Bacon et al. 91, Lo 93].

2.1 The byte segment custodes (BSCs)

BSCs manage a variety of storage media which might range from non-volatile RAM to a terabyte optical jukebox. They present a common byte segment abstraction to the higher level custodes which may be used in any way that is appropriate. This also gives extensibility in that new media may be added within the architecture. An example of how the low level might be used is that a video file may be striped across a number of devices as a number of byte segments in a number of containers managed by different BSCs. A traditional, byte sequence file is likely to map onto a single byte segment. The fact that the low level imposes no specific structure means that arbitrary standard structures, such as MPEG for video or interleaved video and audio, can be supported at the high level.

We use extent-based storage allocation within a byte segment. A storage-block size may be associated with a byte segment, as a hint to the BSC, so that different media may be treated appropriately. We use NVRAM to store data and metadata in the implementation of the BSCs. This allows a fast acknowledgement that a secure write has taken place without the need for synchronous disk access. It also provides a convenient basis for implementing atomic write operations.

2.2 Quality of service for media storage

If an object of a continuous medium type such as video is to be shown as part of a display then a certain rate of delivery must be guaranteed (at some probability) between the custode containing the stored object and the workstation. The mechanism we use to support this is the *session* which is controlled by a *ticket* which guarantees a certain rate of delivery for a specified time. A server can ensure that it does not issue more tickets than it can honour.

We take the view that the storage service should take care only of storage and its responsibility ends when the data reaches the client-end. The purpose of having sessions and tickets is to ensure that continuous media data are delivered in a way predictable by the client. For instance, we have to take account of the fact that data may be variable instead of fixed rate and might be lost in transmission [Lo 93].

2.3 Structured objects

We represent structured objects on *structured file custodes* (SFCs) [Thomson 90]. An SFC is lightweight compared with a full typed-object manager. It supports only byte sequences and storage service identifiers (SSIDs) as primitive storage types but the constructors *sequence*, *record* and *union* are used to create tree structures of arbitrary depth. Since the SFC can locate SSIDs within structured objects it can provide an **existence control** service by traversing them. The representation of structure means that locks can be associated with components of stored objects.

This simple data object store has been used to represent complex, multi-object presentations. An active multimedia database has been created on the SFC to support event driven display, see below.

2.4 Continuous media objects

Clients will use the continuous media file abstraction to store all data that is continuous in nature, whether it is audio, video or interleaved audio and video and whether it is encoded in MPEG, H.121 or any proprietary format. We may have multiple instances of the continuous media file custode (CMFC) running at the same time.

We can claim that the CMFC supports multiple formats because it supports none! It is up to the client at the receiving end to decide how to handle a given stream. The stub at the client end is a "translator". It interprets the byte stream, identifies frame boundaries and extracts timing information. It then sends the data to other processing elements down the pipeline from server to user. For example, in the MPEG standard, the role of this translator is the "Medium Specific Decoder" (which is not part of the standard). Its role is to translate from the digital storage format to

an ISO-11172 stream which is fed into an MPEG codec. Different data with different encoding schemes will have different translators.

3. Composition and display of multimedia presentations

The Pandora project [Hopper 90] gave us insight into the requirements of multimedia applications and created some prototype applications, such as video mail, which are in everyday use at the Laboratory. Additional support is needed if users are to create and interact with multimedia applications as opposed to just using those provided. Our development platform will allow applications to be created without the cumbersome, ad hoc engineering of the lower levels which is necessary in current systems.

- A user must be able to compose a multimedia presentation, as part of an application, and express requirements and policies for its dynamic, interactive display.

  The IMP (Interactive Multimedia Presentation) system [Bates 93] provides a general and flexible interface for use by individuals and groups. Two aspects are involved: the creation of structured **presentation data** and the expression of a **script** to drive a presentation.

- To support the creation of presentation data and meet the requirements expressed in the script we use the MSSA to provide **storage** of presentation structures and their components of various media types and **delivery** of these components at the right rate and at the right time (triggered by possibly composite events). In addition, we require **synchronisation** between components.

Figure 1 gives an overview of the components involved in achieving a display.

We register event expressions of interest with a service which is notified of event occurrences and thus is able to control a presentation. This service drives the delivery of components of a presentation from storage (in a minimal **active, object oriented, multimedia database** built on the SFC). A **synchronisation service**, independent of the storage services and the display manages streams of various types (to remove jitter from a video stream for example) and the synchronisation between them.

3.1 Representation and storage of presentation data

We recognize that applications have certain general presentation requirements and have attempted to provide flexible support for them. The IMP system provides an object oriented framework and a language for the expression of multimedia presentations. Applying an IMP language script to presentation data creates a dynamic display, see Figure 1.

Object classes and instances

We have built a *presentation object store* on the SFC. All objects which may be presented are registered here. Each presentation object is defined to be of one media class and inherits methods for that class. For example, if the medium is *stored video*

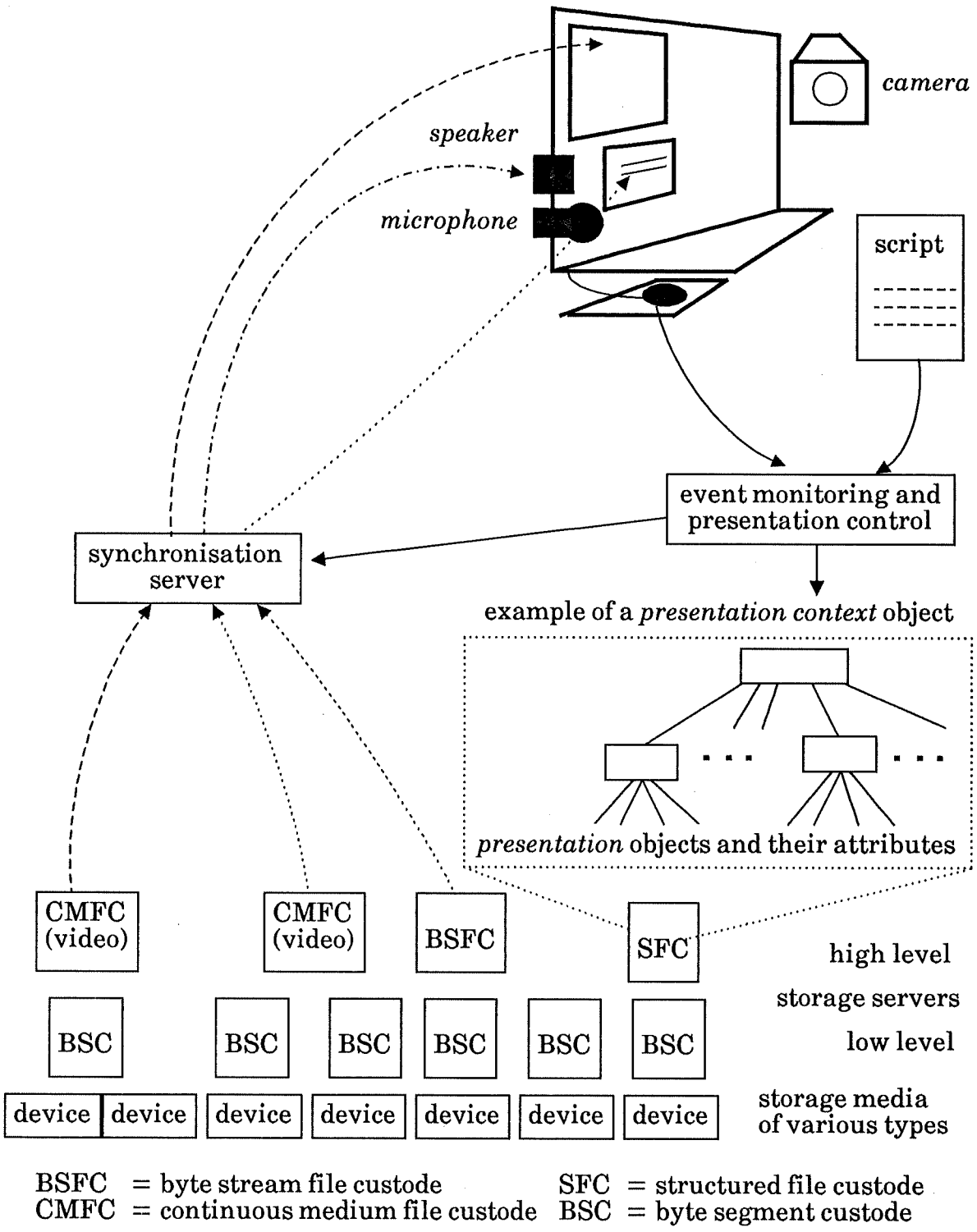


Figure 1 A multimedia presentation driven from storage by a script

the inherited methods are {*play, stop, rewind, fast-forward, pause, resume* ....}. Such a video object would also have a *format class* (e.g. *pandora*).

Each presentation object has a *storage class* and associated object-location information; for example, for storage class MSSA, this would be a storage service identifier (SSID, see Section 2.3) to locate the data on the appropriate storage custode.

Users are able to create their own instances of these objects and tailor them to their requirements by adding attributes (such as descriptions meaningful to the user) and marking events.

A presentation (a *presentation context* object)

Individual presentation objects are grouped into a *presentation context* which represents the data for an entire presentation. This is a structured object, stored on the SFC, which contains the identifiers of the component presentation objects.

Multimedia applications are event-driven: some happening associated with a presentable object has a consequence in the application. It is therefore useful to be able to mark and recognize events. Examples are a user clicking on a certain word of a text document (as in a hypertext link) or a certain frame of video being presented (with which one might associate a subtitle). Objects may be annotated with events and an application is notified of any occurrences during a presentation.

3.2 A script language to express how a presentation is to be displayed

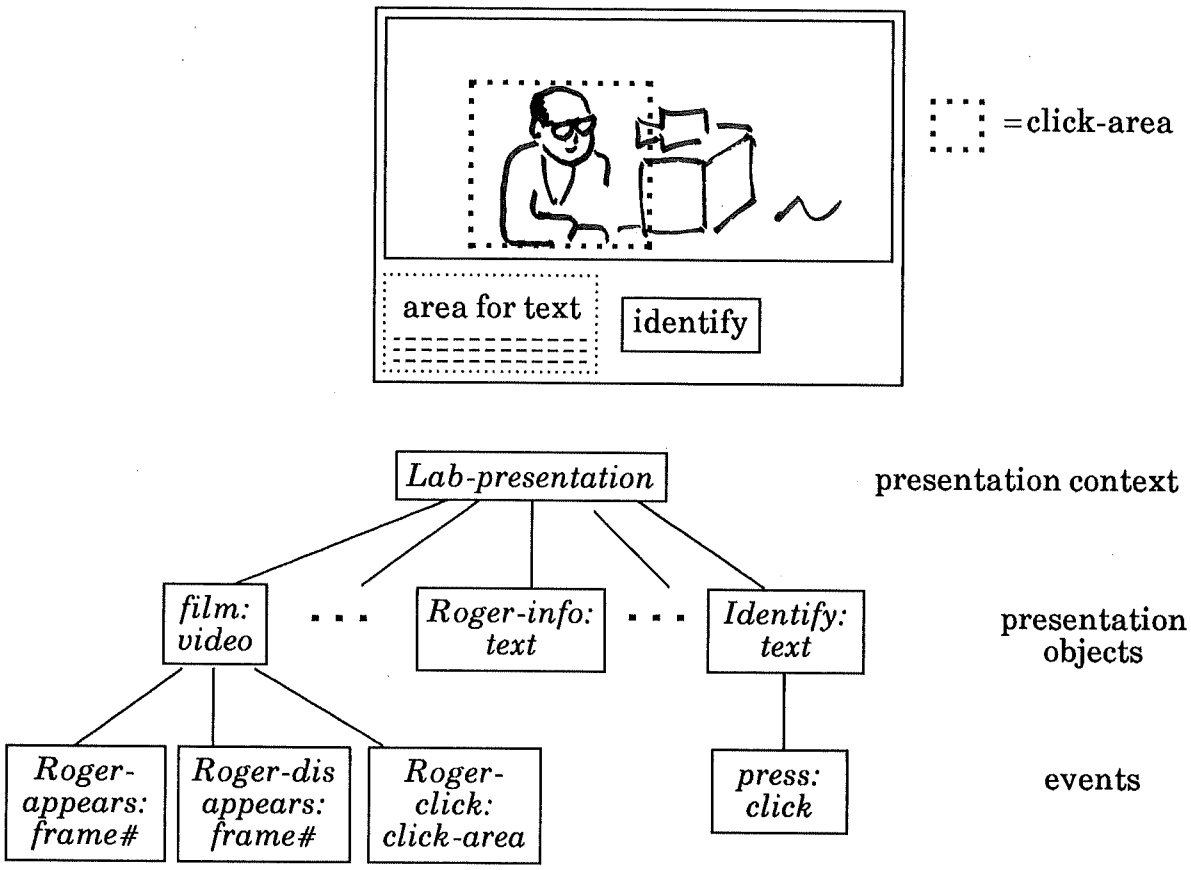
IMP's script language has been designed to express the presentation requirements of an application. It allows a multimedia display to be specified in terms of the presentation objects described above, the relationships between them and the events by which a display is driven.

An application builder may wish to have complex situations involving several of these "primitive events" detected. Time offsets or time-based events may be involved. For example, suppose that we have a video of the Computer Lab. It contains a sequence of the head of department working at his desk. We are interested in whether the user clicked on Roger while he was "on screen" and then clicked on an "identify" button which is also part of the presentation context. Figure 2 shows some presentation data that might be created for this example and extracts from the corresponding script.

To achieve this behaviour, IMP allows primitive events to be combined using logical operators to form **composite event expressions.** [Gehani et al. 92] give a minimal operator set for expressing arbitrarily complex event scenarios with the same expressive power as regular expressions. Event expressions allow the user to express certain temporal orderings, conjunctions and disjunctions over events in a presentation and then monitor for the occurrence of such a situation. The approach adopted constructs finite state machines to recognize such expressions.

IMP provides **invocation sets** for group invocation of methods on a set of related objects. This allows the application to specify, for example, how "the set of video objects" in a presentation should be handled. It also has **synchronisation sets** which support, for example, "lip-synch" between a video clip and the corresponding audio.

It is also important to take into account the notion of **quality of service** (QoS). In the event of demanded resources exceeding available resources, IMP allows the user to specify how a presentation should behave. For example, should it turn off all audio streams or cut the resolution and frame rate of video?



an extract from the script for the presentation might be:

```
context Lab-presentation
        :
rule identify-Roger=
    prior  (
    prior  (Film.Roger-appears, Film.Roger-click)
           and not (happened (Film.Roger-disappears)),
           identify.press
           )
    -->
    {Roger-info.show-window (default-text-style, x,y,height, width)
     Roger-info.play ()
    }  :
```

where the rule syntax is: rule <name> = <event expression> --> <actions>

Figure 2 Example of an IMP presentation and script

3.3 Achieving a dynamic, interactive display

A script is applied to a *presentation context* to achieve a display, see Figure 1. A number of services and mechanisms are needed to achieve this:

An event-monitoring service

Event rules are registered with this service. They are composed of a named event expression and an action to perform if it fires. Every event occurring in a presentation is notified to this service which then updates its state. If an event expression enters an accepting state, it is said to "fire". Actions, such as methods being invoked on objects or callbacks to applications, may be set up to occur when an expression fires.

We envisage that for applications such as Computer Supported Cooperative Working (CSCW), events generated within one user's presentation may cause display to occur at others'.

An active presentation database

The presentation database is active. Events are marked on objects stored in the database. A presentation script may combine these events within event rules. A presentation is driven by the firing of such event rules. The action part of the rule may cause the database to retrieve objects not yet in use or the manipulation of current objects.

Delivery mechanisms

Component objects of a presentation are retrieved from the appropriate BSC with guaranteed rates of delivery using the session and ticket mechanisms described in Section 2. As discussed above, an application should be able to express how its presentation should perform in the event of resource demand exceeding resource availability. The MSSA allows experimentation with the specification and implementation of support for QoS.

A synchronisation service

[Sreenan 92] investigates the problem of network synchronisation for continuous media streams. He suggests that jitter-removal, inter-stream synchronisation and monitoring of frame events (when requested) should be provided as a service. We have adopted this approach and will investigate further how the characteristics of certain stream format types could be passed to this service and how connections of a certain stream type can be requested. Such a service will provide an ideal bridge between the MSSA and the presentation world without either the client or server having to perform explicit synchronisation or know details of the construction of the stream formats.

4. Implementation

The storage service architecture is designed and implementations of the byte segment custode (BSC) and the structured file custode (SFC) are in everyday use. The presentation object store and active database mechanisms have been built above the SFC. The continuous medium file custode (CMFC) and the ticket mechanism are at the stage (March 93) of experimental testing.

A multimedia workstation, a Pandora box, is being used to test our ideas on representation of and support for multimedia presentations. Prototype versions of IMP and the active database are in use. A network based synchronisation server has been built but has yet to be integrated with the storage and display systems.

5. Summary, further work and conclusions

The open storage architecture has given us flexibility and extensibility. Arbitrary file types at the high level are built on a common low level. Quality of service is supported in the form of guaranteed rates of acceptance and delivery of data which take compression into account. A two level architecture could be inefficient but this is avoided by the session and ticket mechanisms which allow the low level to be used directly but securely.

Our experience to date is that the object data model we use is supported well by the minimal structured data service, the SFC. We believe that the concept of an active database is appropriate for multimedia applications and we will evaluate this when we have more experience. We achieve a dynamic interactive display by applying an IMP script to a presentation context which is stored in a lightweight active database.

Certain tools would prove useful in this environment. One example area is the marking of events. Our presentation object store scontains metadata about events associated with instances of objects. It would be beneficial if we could mark specific event types graphically. We envisage graphically marking an area for clicking in a video window or graphically marking a word in a text document for a hypertext link. A video editor should allow visual marking of a specific frame. Tools to assist the design of the spatial layout of the display (the positioning of windows) would also be useful.

The prototype IMP system aims to show that generalised classes of multimedia applications can be expressed using the methods provided. We intend to build "real" applications from several multimedia application domains. Currently multimedia applications fall into classes such as computer-aided learning (CAL), hypermedia, multimedia documents or computer-supported cooperative working (CSCW). So far, such applications have made quite simple demands on existing technology, perhaps because of a lack of good, high level support. We intend to investigate whether a more sophisticated form of support might facilitate emerging application areas. Figures 3 and 4 outline how our system might be applied in two application domains, CSCW and CAL.

In summary, we have all the levels of a quality of service architecture to support multimedia applications designed and implemented, albeit in early prototype form. We will proceed to exploit, develop and evaluate its various components.

Acknowledgements

To other members of the project, past and present: Noha Adly, Mohammed Afshar, Huang Feng, Richard Hayton, Scarlet Schwiderski, Robert Sultana, Sue Thomson, Tim Wilson, Zhixue Wu. To Heather Brown for many helpful discussions.
To SERC for grant GR/H 13666 which supports the storage service aspects of the work. To Olivetti Research Laboratories, Cambridge, for network support and use of a Pandora box. To John Wilkes of HP Research Laboratories Palo Alto, CA, for support and continued interest.

References

[Bacon et al. 91]
Bacon J M, Moody K, Thomson S E and Wilson T D, "A Multi Service Storage Architecture" *ACM Operating Systems Review* 25(4), October 1991

[Bacon 93]
Bacon J M "Concurrent Systems" Chapter 22, Addison Wesley 1993

[Bacon and Moody 93]
Bacon J M and Moody K, "Objects and Transactions for Modelling Distributed Applications: Concurrency Control and Commitment" Computer Laboratory TR 293, 1993

[Bates 93]
Bates J O, "Support for Real-time Interactive Presentation of Distributed Multimedia" Computer Laboratory PhD thesis in preparation 1993

[Gehani et al. 92]
Gehani N H, Jagadish H V and Shmueli O, "Composite Event Specification in Active Databases: Model and Implementation"
Proceedings of the 18th VLDB Conference, Vancouver, August 1992

[Hopper 90]
Hopper A, "Pandora, an experimental system for multimedia applications"
ACM Operating Systems Review 24(2), 19-34, April 1990

[Lo 93]
Lo S L, "Multi-service Storage Architecture" Computer Laboratory PhD thesis in preparation 1993

[Moody et al. 93]
Moody K, Bacon J M, Bates J O, Hayton R, Lo S L, Schwiderski S, Sultana R and Wu Z "OPERA: Storage, Programming and Display of Multimedia Objects" submitted to IEEE Fourth Workshop on Future Trends in Distributed Computing Systems, Lisboa 93 and Computer Laboratory TR 294

[Sreenan 92]
Sreenan CJ "Synchronisation Services for Digital Continuous Media" Computer Laboratory PhD thesis 1992

[Thomson 90]
Thomson S E, "A Storage Service for Structured Data" Computer Laboratory PhD thesis 1990

[Wu et al. 93]
Wu Z, Moody K M and Bacon J M "A Persistent Programming Language for Multimedia Databases" submitted to the Fourth Workshop on Database Programming Languages, New York, Aug-Sept 93,
and Computer Laboratory TR 296

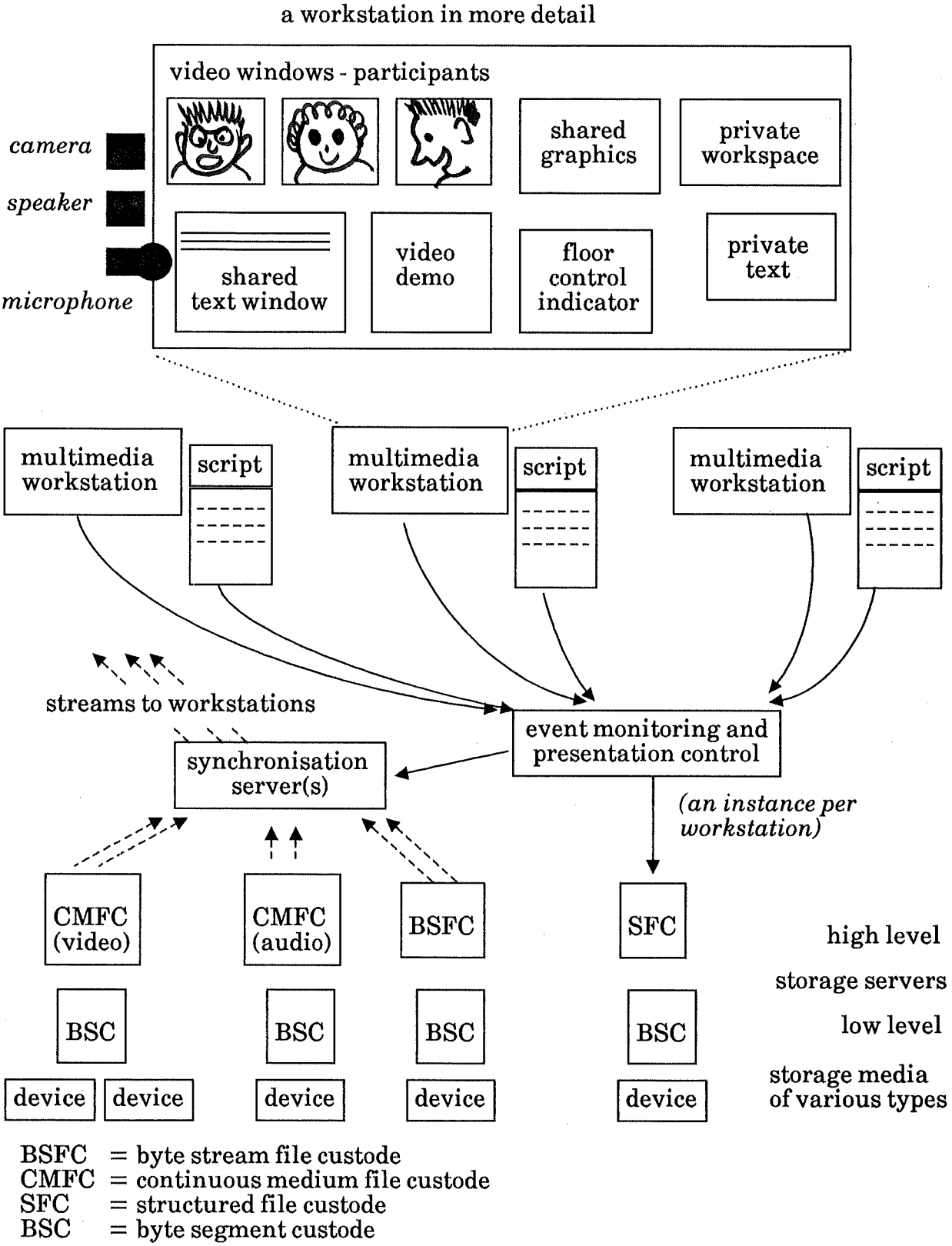


Figure 3 A CSCW conferencing application

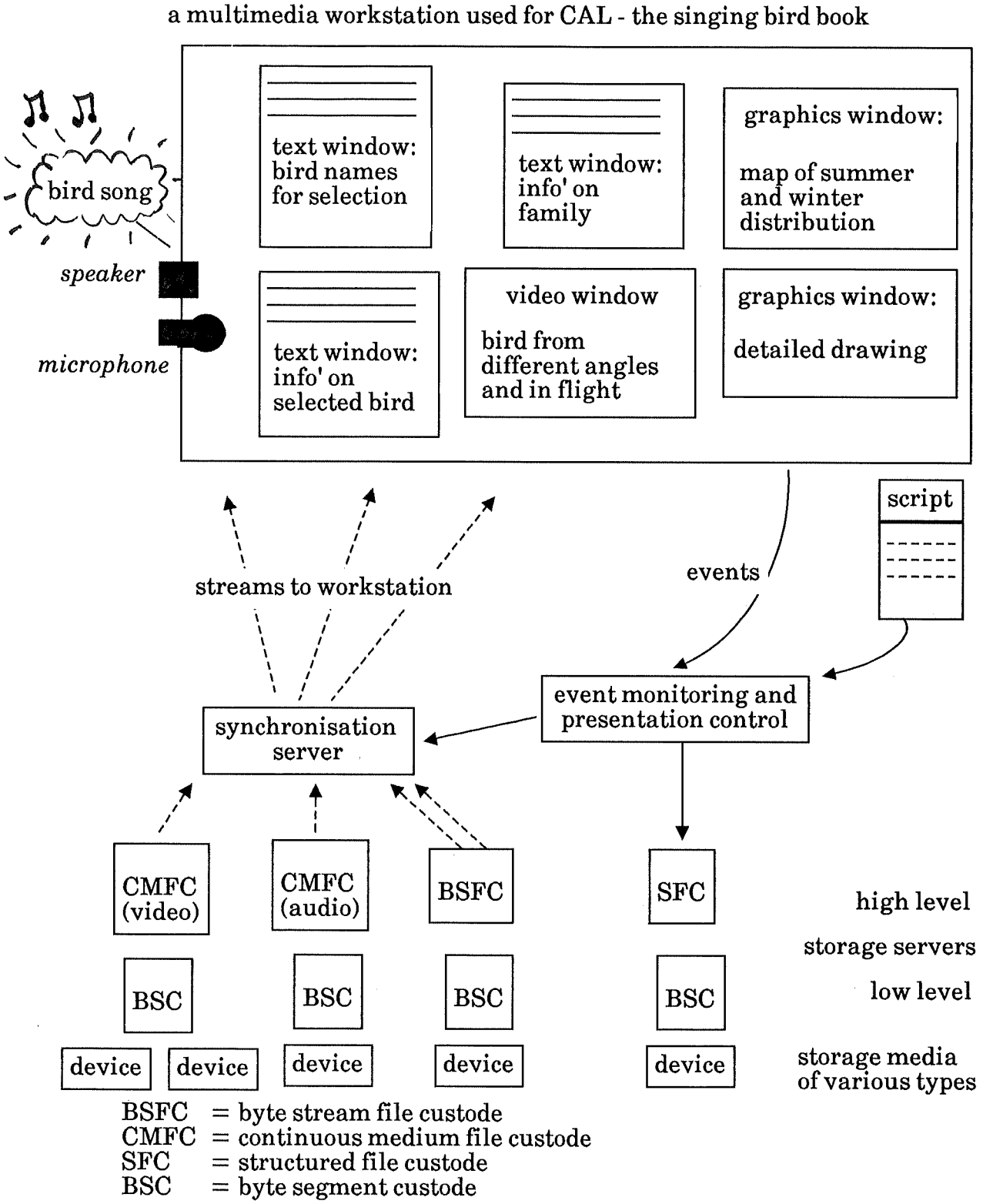


Figure 4 A CAL application, a "singing bird book"