

ment. Interception and presentation of audio/video data streams is thus local matter of each endsystem's operating/window system. An Audio/Video Communication Service provides platform-independent functions for transfer and presentation of audio/video data streams. Its export/import concept, which is based on the notion of video rectangles and audio ports, hides the aspects of coding conversion and presentation synchronization.

With JVTOS version 1.0, which was demonstrated during the Interop Paris fair in October 1993, we can share a wide range of applications among Sun Sparcstations and Macintosh computers. However, it is still asymmetric in that shared applications must execute on a Sun Sparcstation, i.e. Macintosh applications cannot be shared yet. Application sharing capabilities are restricted to non-continuous media, i.e. text/graphics, still pictures, and animation.

Audio/video sharing capabilities will be included in the next version of JVTOS, which will realize the concept presented in this paper. Current work is devoted to the implementation of the Picturephone, the Multimedia Application Sharing Services, and of the AVCS. The AVCS will be based on a transport system developed by another CIO work package. We plan to demonstrate that JVTOS version to a larger audience during the Third International Conference on Broadband Islands (Hamburg, June 1994).

Acknowledgements

The work discussed in this paper was performed in the context of the RACE II project CIO (R2060) and specifically within the work package 4.2, in which many of the ideas put forward were born. We would like to acknowledge the contributions of all involved partners that were left unmentioned in this text.

References

- [1] Gabriel Dermler, Konrad Froitzheim: "JVTOS - A Reference Model for a New Multimedia Service". *Proceedings, 4th IFIP Conference on High Performance Networking (hpn '92)*, pp. D3/1 - D3/15. Edited by A. Danthine, O. Spaniol. Liège, 1992.
- [2] Thomas Gutekunst, Thomas Schmidt, Günter Schulze, Jean Schweitzer, Michael Weber: "A Distributed Multimedia Joint Viewing and Tele-Operation Service for Heterogeneous Workstation Environments". *Proceedings, GI/ITG Workshop on Distributed Multimedia Systems*, pp. 145 - 159. Edited by W. Effelsberg, K. Rothermel. Stuttgart, 1993.
- [3] Gabriel Dermler, Thomas Gutekunst, Bernhard Plattner, Edgar Ostrowski, Frank Ruge, Michael Weber: "Constructing a Distributed Multimedia Joint Viewing and Tele-Operation Service for Heterogeneous Workstation Environments". *Fourth IEEE Workshop on Future Trends of Distributed Computing Systems*. Lisboa, 1993.
- [4] Wulfdieter Bauerfeld: "RACE-Project CIO (R2060): Coordination, Implementation and Operation of Multimedia Tele-Services on Top of a Common Communication Platform". *Proceedings, International Workshop on Advanced Communications and Applications for High Speed Networks (IWACA '92)*. München, 1992.
- [5] Konrad Froitzheim, Edgar Ostrowski, Nelson Pires: "Multimedia Applications and how they Interact with Workstation Hardware and Operating Systems". *Internal Report of RACE/CIO WP 4.2*. Ulm, 1992.
- [6] Michael Altenhofen: "Erweiterung eines Fenstersystems für Tutoring-Funktionen". *Diploma Thesis at Universität Karlsruhe*. Karlsruhe, 1990.
- [7] Greg McFarlane: "Xmux - A system for computer supported collaborative work". *Proceedings, 1st Australian Multimedia Communications, Applications & Technology Workshop*. Sydney, 1991.
- [8] Hussein M. Abdel-Wahab, Mark A. Feit: "XTV: A Framework for Sharing X Window System Clients in Remote Synchronous Collaboration". *Proceedings, IEEE Conference on Communications Software: Communications for Distributed Applications and Systems*, pp. 159 - 167. Chapel Hill, 1991.
- [9] John Eric Baldeschwieler, Thomas Gutekunst, Bernhard Plattner: "A Survey of X Protocol Multiplexors". *ACM Computer Communication Review*, Vol. 23, No. 2, pp. 13 - 22. New York, 1993.
- [10] Thomas Gutekunst, Bernhard Plattner: "Sharing Multimedia Applications Among Heterogeneous Workstations". *Proceedings, Second International Conference on Broadband Islands*. Edited by O. Spaniol, F. Williams. Athens, 1993.
- [11] Oliver Jones: "Introduction to the X Window System". *Prentice-Hall*. Englewood Cliffs, 1989.
- [12] Miroslav Vodslon: "Feasibility of Translating Native Windowing Systems to X Window". *Internal Report of RACE/CIO WP 4.2*. Berlin, 1992.
- [13] Andreas Rozek, Ilka Miloucheva, Paul Christ: "Multimedia Communication Platform: Transport Service Programmers Interface (TPI)". *Public Deliverable of RACE/CIO WP 3 (R2060.US.CIO.DS.P.003.b2)*. Stuttgart, 1993.

pressed images. Each compressed image is intercepted by the Multimedia Application Sharing Service and handed over to the AVCS for distribution. The alternative of displaying the images locally and instructing the AVCS to retrieve them from the video rectangle is obviously less efficient since it implies a decompression and a recompression step.

Control Interface	ExportVideoFromWindow
	ExportAudioFromPort
	ImportVideoToWindow
	ImportVideoToPort
	CloseExport
	CloseImport
	QueryTransferQualities
Data Interface	PutVideoFrame
	PutAudioSlice

Table 1: AVCS Interface Commands

5 Audio Sharing Example

Consider the situation where several users are running a JVTOS session. One user initiates the sharing of an audio application, i.e. audio output of the application is to be distributed and presented to all session participants.

As far as audio is concerned, the following sequence of actions is performed at the workstation of the user that initiated the sharing of the audio application:

1. The Multimedia Application Sharing Service (MASS) intercepts the audio output initiation command that is generated by the shared application.
2. The MASS queries the possible transfer qualities for audio export from the AVCS.
3. The MASS negotiates the transfer quality for audio with its peer entities.
4. The MASS requests audio export with a given transfer quality. Note that this step may be repeated several times for different transfer qualities.
5. The MASS notifies its peer entities about the export id(s) of the audio source.
6. The AVCS establishes a transport connection according to the negotiated transfer quality as soon as it receives an import request from a remote AVCS entity.
7. The AVCS gets the audio data periodically from the audio server and sends it to the peer AVCS entity.

On the other users' workstations, the sequence is as follows:

1. The MASS receives an indication that audio is to be shared.

2. The MASS queries the possible transfer qualities for audio import from the AVCS.
3. The MASS negotiates the transfer quality for audio with the peer entity that initiated the audio sharing.
4. The MASS requests audio import with the negotiated transfer quality.
5. The AVCS relays the import request to the exporting AVCS entity and waits for the transport connection to be established.
6. The AVCS gets the audio data periodically from the exporting AVCS entity and hands it over to the audio server for presentation.

Now, audio is distributed and presented to all session participants. As soon as the initiating user stops audio sharing or the applications indicates end of audio output, the Multimedia Application Sharing Service indicates end of audio sharing to its peers and requests the AVCS to close the export. The AVCS then takes appropriate action.

6 Implementation

Sun Sparcstations. The decision of whether to implement pseudo interfaces either in kernel or in user space was driven by the existing Sparcstation software environment. The window system and the Xplx extension were implemented in user space. So, the respective pseudo interfaces were implemented in user space as well. The standard Sun audio interface is realized as a stream-oriented device and can only be accessed via system calls. Therefore the audio pseudo interface was implemented in the kernel as a loadable kernel module.

Since audio is not related to a window as video is, at present, we cannot distinguish between audio stream associated with a given shared application and audio streams of local relevance only. Therefore, we only allow the user to decide whether or not to share audio as a whole.

Macintosh Computers. Interception of control information is achieved by changing the entries referring to start addresses of routines that are executed when calls to the Macintosh Toolbox occur. Instead of the initial system routines, new routines are provided that intercept control information, perform appropriate sharing action, and then call the initial system routine.

7 Summary

This paper has presented our approach for sharing audio/video applications among heterogeneous platforms. Sharing audio/video applications requires audio/video data streams to be multiplexed and presented on remote endsystems.

There are no unifying concepts for audio/video handling established yet that can be used in a heterogeneous environ-

An AVCS entity is able to send audio/video data from the same source in more than one format. For this purpose, it employs modules converting between the format provided by the shared application and the format used for data transfer. As a result, receivers of the same information stream may be provided with different data formats, e.g. 11 khz Audio or 8 khz Audio. It is known that maintaining good audio quality with such conversions is hard to achieve in software. Instead, our intention is to provide a realtime conversion and tolerate possible loss in quality, rather than having no audio conversion at all.

The negotiation ensures that each receiver gets data in a format that can be processed by its machine. Furthermore, it tries to reduce the number of transfer formats for the AVCS, thus relieving the sender of unnecessary conversions. Since the AVCS has to transfer data for each specific format through a dedicated transport channel, the reduction also implies that the number of multicast channels to be set up is reduced.

Data Transfer. After setting up transport connections, the AVCS is in charge of taking local audio and video data and transferring it to remote AVCS entities. A remote AVCS entity delivers video data directly into a video rectangle and audio data to an audio server port, respectively. Depending on the shared application, the sender AVCS gets the data in two different ways. For *analogue video sources*, where the data is not handled by the application itself, the data is read in periodically by the AVCS directly from the I/O device, e.g. the video frame grabber.

For applications that handle audio/video data internally, the data is handed over to the AVCS by the audio/video sharing module of the Multimedia Application Sharing Service. Recall that the pseudo interface intercepts all I/O calls generated by a shared application to audio and video devices. This concept covers in particular the class of applications that retrieve *stored audio/video* data for (local) presentation.

Synchronization. During data transfer, the AVCS ensures that the temporal relationship within and between audio/video streams, as observed at the source location, is restored at the sink locations before presentation. For interstream synchronization, the AVCS relies on an interstream synchronization service, which is able to calculate the artificial delay that has to be inserted by the AVCS for a specific stream before presentation. The AVCS is responsible for performing all synchronization: intrastream synchronization on a per-channel basis and interstream synchronization between two or more channels. For each of several correlated streams, the synchronization service calculates the artificial delay that has to be inserted before presentation. Thus, synchrony between the streams is achieved at presentation time.

Encapsulation and Generality. The presented architecture encapsulates the described aspects of dealing with audio/video transfer in the AVCS. This has the advantage that a tuned implementation of AVCS can be provided without affecting other parts of JVTOS. Such an implementation is expected to outperform most of the “ad-hoc” approaches taken for transferring multimedia data across networks. Also, with rapidly changing techniques, multimedia sharing may soon be outdated if it is too closely intertwined with specific ways of audio/video handling. In our approach, the main parts to be adjusted are concentrated in the AVCS.

Though the AVCS was introduced in the context of multimedia application sharing, its design is such that it may be used by arbitrary multimedia applications. In other words, the AVCS can be viewed as a building block that can be advantageously integrated into new distributed multimedia applications. Within the JVTOS working group, it is currently foreseen to provide a multi-party picturephone application implemented on top of the AVCS.

4 The AVCS Interface

Export/Import Concept. The AVCS interface is based on an export/import concept. An AVCS client (e.g. the Multimedia Application Sharing Service) exports a source (video rectangle or audio port). With each exported audio/video source, the AVCS client associates exactly one transfer format. Thus, if a client wishes to export the same source in different formats, it has to request a corresponding number of exports. The transfer format is fixed for each initiated export. If the format has to be changed for some reason, the AVCS client has to cancel the initial export and to initiate a new export.

Each audio/video export is identified by a unique export id. This export id is required by remote AVCS clients to import the audio/video source. When importing a source, both a sink (video rectangle or audio port) and a transfer quality have to be specified. The setup of a connection between the exporting and importing endsystem is performed by the corresponding AVCS entities.

Control and Data Interface. The AVCS interface consists of a control part and a data part. The control interface allows an AVCS client to control the behavior of the AVCS for any export/import, while the data interface allows to hand over data of a previously exported source. The data is then transmitted to all locations that have imported the specific source and delivered to the respective sinks (video rectangles or audio ports).

The usage of the data interface is recommended when the shared application itself feeds the framebuffer with compressed video data. An example for this is the presentation sharing of a movie that comes in a sequence of com-

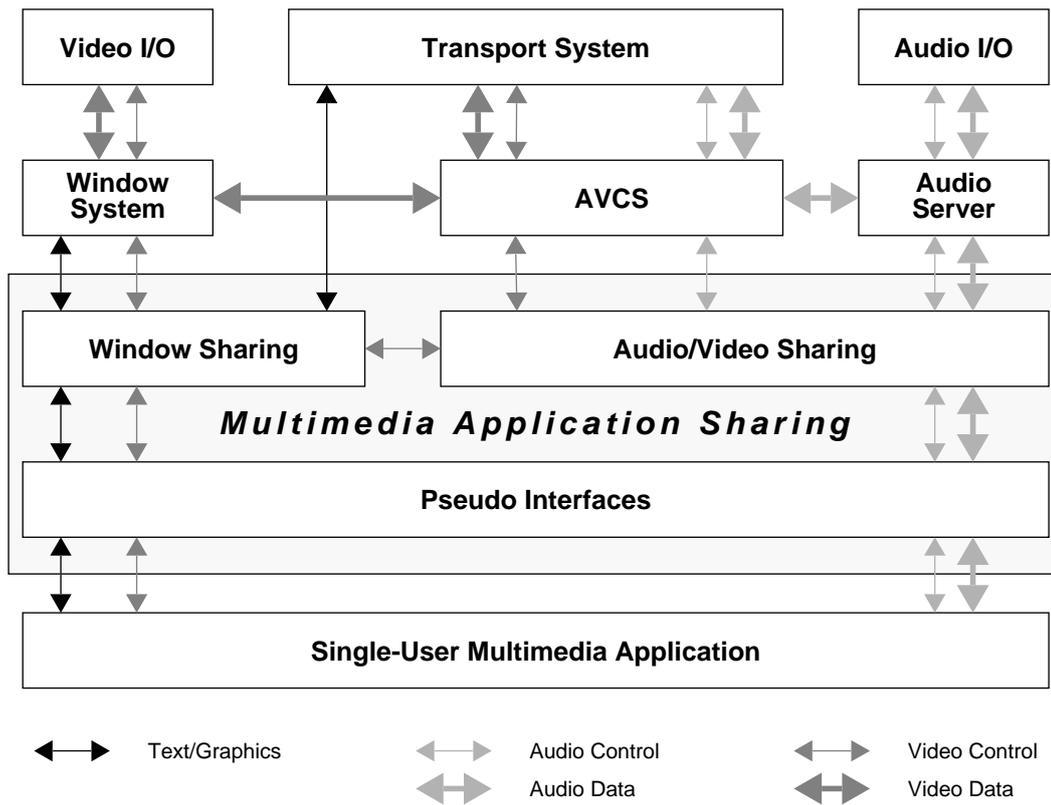


Figure 1: Application Sharing Data Flow

is connected to multiple output ports, multiplexing is done. Coding conversions mediate between different audio codings.

Synchronization. The *interstream synchronization service* coordinates presentation of correlated data streams. At source locations, audio/video samples are timestamped. At sink locations, the earliest possible point in time for synchronized presentation is calculated from the timestamps and the estimated presentation delay of each data stream.

Transport System. The AVCS supports connection attributes related to the QoS of audio and video. From these, QoS parameters for the underlying transport connections are calculated. The AVCS is designed to make use of an advanced transport system [13] that provides services suitable for multimedia. However, the data protocol used within the AVCS is flexible enough to be used on top of other transport systems as well.

3 The Audio/Video Communication Service

Abstractions. The main benefit of the Audio/Video Communication Service (AVCS) is that it allows operations on audio and video streams by using two simple ab-

stractions, namely the *video rectangle* of a window and the *audio server port*. For these, two corresponding connection types are defined. A video connection associates a source video rectangle with a set of sink video rectangles, while an audio connection does the same for audio server ports. For sharing audio and video streams, connection setup and termination is initiated by the audio/video sharing module of the Multimedia Application Sharing Service.

Transfer across networks as well as presentation of a data stream is encapsulated in the AVCS. Thus, applications are relieved of any conversions applied to either control or stream data, which are necessary in order to deal with transfer codings for interconnected AVCS entities, and of any timing mechanisms necessary for the presentation of time-dependent data such as audio and video. Both aspects considerably simplify the development of multimedia applications across heterogeneous platforms.

Negotiation of Transfer Formats. The negotiation of the transfer formats to be used between the source and the sinks is in the responsibility of the AVCS clients, i.e. of the applications that make use of the AVCS. In JVTOS, this is the Multimedia Application Sharing Service and the Picturephone.

(what you see is what I see) not only for text/graphics application output as it is done in Timbuktu or the known X multiplexers (e.g. SharedX, Xmux, XTV) [6, 7, 8, 9], but also for multimedia applications such as multimedia authoring systems or film editors.

The Picturephone allows audiovisual conversation between the participants of a JVTOS session such that they are able to discuss the contents of shared applications. Telepointing provides pointing tools used for drawing attention to particular objects in the windows of shared applications.

A key aim of JVTOS is to *support and bridge heterogeneous platforms*. It supports Sun Sparcstations and Macintosh computers. The inclusion of PCs and Silicon Graphics Indigo workstations are currently being considered. A second aim is the *integration of audio and video* in order to have multimedia applications shared. The third goal is aimed at *dynamic user participation*, i.e. participants should be allowed to enter and leave JVTOS during a session. The fourth goal is to *symmetrically distribute each service component* on all supported platforms in order to achieve a scalable and well-balanced solution.

2 Our Approach

The Sharing Metaphor. Application sharing means associating multiple users with single-user applications. The set of a user's input/output facilities comprises a screen (framebuffer) for text/graphics and video output and a loudspeaker for audio output. A microphone is used for audio input. Keyboard and mouse serve for text/graphics input. Video input is possible by grabbing a region of the framebuffer.

Multiplexing Data Streams. The Multimedia Application Sharing Service is based on a distributed X multiplexer [10] that was extended for audio/video support and heterogeneous platforms. X [11] was chosen as the interchange protocol for text/graphics data as it is a network-transparent, device-independent windowing and graphics system currently supported by most leading workstation manufacturers. For non-X windowing systems, *translators* [12] map non-X protocols to the X protocol. Commercially available X servers are used for X to non-X conversion. On the Macintosh platform, application output is translated by intercepting calls to the Macintosh Toolbox and translating them into X requests. X events (application input) are translated into Macintosh events, which are subsequently read from the event queue by the applications as their input.

Sharing multimedia applications requires several data streams to be multiplexed and presented on remote endsystems. Sharing audio/video data streams is strongly related to the characteristics of the local operating/window system. The Multimedia Application Sharing Service interacts with

the local operating/window system to intercept the required control information and the data samples to be shared. To present received audio/video correctly on a remote endsystem, again, interaction with the remote endsystem's operating/window system is necessary. Control information for video comprises position and size of video windows as well as overlay mode and compression parameters. Intercepted control parameters for audio concern encoding format and volume.

Audio/Video Communication Service. There are no unifying concepts for audio/video handling established yet that can be used among heterogeneous platforms. In practice, we find many different interfaces. Interception and presentation of audio/video data streams is thus local matter of each endsystem's operating/window system. The *Audio/Video Communication Service (AVCS)* hides the complexity of audio/video processing in the endsystem behind an interface that provides platform-independent functions for transferring audio/video data streams from a local source to a remote sink and vice versa. Although the AVCS might be directly used by any application, the design focus was on the support for sharing multimedia applications.

Supported Audio/Video Interfaces. Multimedia applications have to use well-defined interfaces in order to be sharable. On the Sparc platform, we support the Sun audio device for audio and the Xplx extension (Parallax XVideo) for video. On Macintosh computers, selected components of QuickTime are supported. As these interfaces do not provide enough support for CSCW, the AVCS provides *audio and video stream handlers* supporting intrastream and interstream synchronization as well as an *audio server*. With the AVCS, applications to be shared still use the specific interfaces of the operating/window system for local operations. The AVCS solves the heterogeneity problems for the communication between the endsystems.

Interception and Presentation. Pseudo interfaces that replace the standard interfaces handle the supported audio/video sharing access points for the specific workstations. They offer unchanged interface functionality as expected by an application. Application requests related to audio/video data streams are converted into commands for the AVCS. The AVCS handles the transfer of audio and video data between the endsystems. Video is always associated with a window such that source AVCS entities know where to retrieve video data, and sink AVCS entities know where to display it. For audio, the procedure is similar: here, an audio server port is the equivalent to a window.

Audio Multiplexing/Mixing. JVTOS contains an *audio server* that allows simultaneous output of audio streams originating from various sources. It can be understood as a switch panel where an input source can be connected to one or multiple output sinks. Whenever two or more streams meet at an output port, mixing takes place. If an input port

Sharing Audio/Video Applications Among Heterogeneous Platforms

Gabriel Dermler¹, Thomas Gutekunst², Edgar Ostrowski³, Frank Ruge³

University of Stuttgart¹
IPVR
Breitwiesenstraße 20-22
D-70565 Stuttgart, Germany

☎ +49-711-7816423
Fax +49-711-7816424
<dermler@informatik.uni-stuttgart.de>

Swiss Federal Institute of Technology (ETH)²
Computer Engineering and Networks Laboratory (TIK)
Gloriastrasse 35, ETH-Zentrum
CH-8092 Zürich, Switzerland

☎ +41-1-6327008
Fax +41-1-6321035
<gutekunst@tik.ethz.ch>

Technical University of Berlin³
PRZ, Sekretariat MA 073
Straße des 17. Juni 136
D-10623 Berlin, Germany

☎ +49-30-31425985
Fax +49-30-31421114
<{ostrowski, ruge}@prz.tu-berlin.d400.de>

Abstract

JVTOS (Joint Viewing and Tele-Operation Service), which is being implemented within CEC/RACE II project CIO (R2060), is a new teleservice for high-speed networks enabling distributed users to work in a collaborative fashion with multimedia. Core functionality of JVTOS is provided by an application sharing service that allows multimedia applications to be displayed and interacted with on multiple users' workstations simultaneously.

The focus of this paper is on the audio/video elements of shared applications. We discuss the problems of sharing multimedia applications and describe our conception that supports interoperability between heterogeneous platforms.

1 Introduction and Motivation

A computer-supported cooperative work (CSCW) environment requires *joint viewing*. This allows multiple users, each on his own computer workstation, to view and interact with a single application. A possible solution is to build a new set of *cooperation-aware applications* that explicitly support this requirement. Such an approach has several problems. Perhaps the most critical of these is that users would be limited to the use of only special cooperation-aware applications. Considering the diversity of computer applications, this requirement appears very limiting.

Application sharing is another solution to the joint viewing problem. It exploits properties of the operating/window system to allow joint viewing with unmodified ap-

plications. Such an approach has several advantages. Firstly, users are not required to use new applications, they can share their existing applications. Secondly, the joint viewing system does not need to be modified to support new applications or changes to existing applications. Finally, the task of developing a CSCW environment is greatly reduced. Instead of reimplementing many existing programs, the developers only have to implement an application sharing service.

JVTOS (Joint Viewing and Tele-Operation Service) [1, 2, 3] is a new teleservice for high-speed networks enabling distributed users to work in a collaborative fashion with multimedia. It was developed by work package 4.2 of CEC/RACE II project CIO (R2060) [4]. JVTOS is structured into a set of user-level and low-level services. The user-level services are (1) *Session Management*, (2) *Multimedia Application Sharing*, (3) *Picturephone*, and (4) *Telepointing*. The two low-level services, the *Audio/Video Communication Service* and the *Interstream Synchronization Service*, are hidden from the user. They target the specific requirement of distributing multimedia information.

JVTOS uses a session-based model. The *Multimedia Application Sharing Service* allows *cooperation-unaware single-user multimedia applications* to be displayed and interacted with on multiple users' workstations simultaneously. The terms "cooperation-unaware" and "single-user" denote that the applications were actually constructed for a single user only and hence are not aware of being run in a group context. Multimedia applications may handle text and graphical information, still pictures, moving pictures (video and animation), and sound [5]. It is an outstanding feature of JVTOS that it realizes the WYSIWIS concept