JVTOS
-
Multimedia Telecooperation Interconnecting Heterogeneous Platforms

Gabriel Dermler[1], Thomas Gutekunst[2], Edgar Ostrowski[3],

Nelson Pires[4], Thomas Schmidt[5], Michael Weber[6], Heiner Wolf[7]

University of Stuttgart[1]
IPVR
Breitwiesenstraße 20-22
D-70565 Stuttgart, Germany
<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
<gutekunst@tik.ethz.ch>

Technical University of Berlin[3]
PRZ, Sekretariat MA 073
Straße des 17. Juni 136
D-10623 Berlin, Germany
<ostrowski@prz.tu-berlin.d400.de>

INTERSIS Automaçáo[4]
Estrada de Paço de Arcos, 48
PT-2780 Oeiras, Portugal
<M4240@eurokom.ie>

Siemens AG[5]
ZFE ST SN 21
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
<schmidt@dfki.uni-sb.de>

German Research Center for Artificial Intelligence (DFKI)[6]
KIK-TEAMKOM
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
<mweber@dfki.uni-sb.de>

University of Ulm[7]
Distributed Systems
Oberer Eselsberg
D-89069 Ulm, Germany
<wolf@informatik.uni-ulm.de>

Abstract

JVTOS, the “Joint Viewing and Tele-Operation Service”, is an advanced teleservice allowing distributed users to work in a collaborative fashion with multimedia. JVTOS offers services for multimedia collaboration across high-speed networks and is primarily aimed at running in heterogeneous workstation environments comprising different hardware platforms and also different operating and window systems.
JVTOS comprises facilities for session management, floor control, multimedia application sharing, telepointing, and audio/video communication.
This paper describes the design and implementation of JVTOS on different platforms.

1 Introduction and Motivation

Information processing and data transmission have made the world smaller since the first images were drawn on cave walls. Today, the technical environment for high-speed data interchange enables humans around the world to collaborate with each other without a need for traveling.
Synchronous collaboration usually takes place with all participants being in the same room. Participants are engaged in a face-to-face dialog in which black- or whiteboards, overhead projectors and other equipment are frequently used. Many telecooperation tools mirror such meeting rooms with the users sitting in the same room in front of their computers. High-speed networks make a distributed version of this approach possible, both technically and economically. Multimedia information may be transferred via the network offering a joint usage of applications, as well as audiovisual communication facilities. Users may thus attend a “virtual” meeting without even leaving their office, being able to retrieve information from their personal or corporate databases.
Several of such telecooperation tools are being brought to the market designed for specific operating systems. These tools face a world of heterogeneity and so the feasibility of cooperative working strongly depends on the system the user has access to.
This paper presents a system, its design and its implementation, competing to bridge these gaps and to overcome the frontiers of proprietary hard- and software: JVTOS. This “joint viewing and tele-operation service” enables synchronous joint working in heterogeneous workstation environments comprising different hardware platforms and also different operating and window systems. Specifically, JVTOS supports SUN Sparcstations with SunOS/Solaris, Siemens-Nixdorf workstations RW420 with IRIX 4.x, the Apple Macintosh with MacOS, and the IBM PC with MS-Windows.
JVTOS is issue of work package 4.2 within RACE 2060 project CIO [4].

2 Service Description

JVTOS is a new telecooperation service for high-speed networks [5], [7], [9]. It is structured into a set of four user-level services: *Session Management*, *Application Sharing*, *Picturephone* and *Telepointing*.
The *Session Management Service* is the major control of the entire JVTOS teleservice. It administrates and runs sessions. Sessions are the frame in which a collaboration takes place. The Session Management Service offers a variety of admission and floor control policies to accommodate different ways of cooperative work.
The *Application Sharing Service* allows cooperation-unaware single-user multimedia applications to be shared among several heterogeneous workstations. 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 [11]. To maintain the single-user behavior of shared applications, floor control is used to determine which user is allowed to direct input to a shared application.
The *Telepointer Service* allows a session participant to move a telepointer in shared windows being visible to all other session participants. The Telepointer Service also distributes the floor holder’s mouse pointer.
The *Picturephone* offers desktop video conferencing and thus allows the session participants to communicate audiovisually to each other.
The key aims of JVTOS are to *support and bridge heterogeneous platforms* and the *integration of audio and video* in order to have multimedia applications shared. It is an outstanding feature of JVTOS that it realizes the WISIWYS concept (what I see is what you see) not only for text/graphics application output as it is done in e.g. Timbuktu or the known X multiplexers (e.g. SharedX, Xmux, XTV) [1], [2], [3], [10], but also for multimedia applications such as multimedia authoring systems or film editors. JVTOS is also aimed at *dynamic user participation*, i.e. participants should be allowed to enter and leave JVTOS during a session.

2.1 Session Management

In the context of a session, users may be invited by the session chairman, or they may request to join or leave an existing session, the chairman can assign and revoke the floor for shared applications. The *Session Management Service* (SMS) coordinates all these operations necessary from the start of a session until its end. It acts as the mediator between the users and the involved services by providing a set of operations which are grouped into core session management (e.g. open/close the session), participant management (e.g. invite a participant), floor control (e.g. assign/revoke the floor) and service management (start/terminate a service).

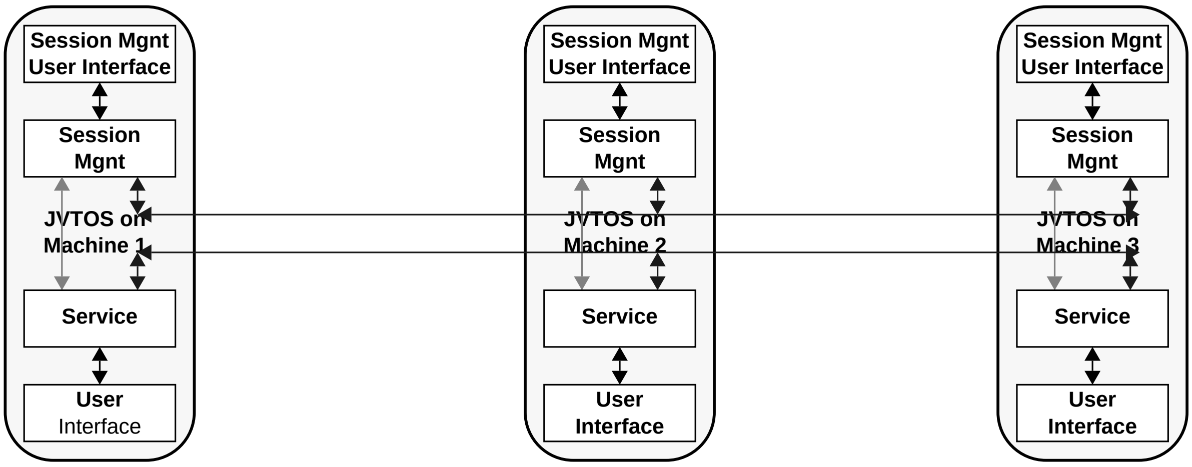

Figure 1: Distributed Session Management

The SMS also includes the *Management Information Base (MIB)* which holds session-static as well as session-dynamic information. Static information relates to JVTOS users, such as their names and addresses. Dynamic information relates to session-specific data, such as the current set of participants and their roles (e.g. the current floor holder), and the services active in a session at a given time. The MIB can be accessed by all services to get relevant information. Since the MIB was conceived as a unique global store in a session, it can in particular be used by distributed parts of a service for non time-critical, intraservice communication across machine boundaries. The MIB provides the stored data in a network transparent manner, i.e. the clients do not know where the data is actually stored.
Since the MIB should not and cannot necessarily “know” the semantics of data to be stored in it, a generic storage concept was conceived. Data is stored in the MIB as so-called attributes [13]. Each attribute has a header and can have an arbitrary amount of attribute elements. It is possible to create, change and delete attributes.
Like the other services, the SMS is realized in physical distributed way (Figure 1). On each machine the SMS resides as a component of the whole JVTOS system. If a user calls a service the session management service module on his machine invokes the affected service module.

2.2 Multimedia Application Sharing

A telecooperation 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* which 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 applications. 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 telecooperation environment will be greatly reduced. Instead of reimplementing many existing programs, the developers only have to implement an application sharing service.
Applications and terminals are the basic entities in the application sharing service. *Application* means a cooperation-unaware single-user multimedia application running on a computer, *terminal* denotes the set of a user’s input/output facilities such as display, keyboard, mouse and audio/video input/output devices. The *Multimedia Application Sharing Service* (MASS) allows all session participants the joint viewing of an application and directing input to the application. The participants may use different hardware platforms and window systems, namely:

- *X11R5* on *SUN/SNI* workstations
- *QuickDraw* on *Mac*
- *MS-Windows* on *PC*

The support for heterogeneous platforms comprises three items:

- *Application output:*
  Application output may be presented on any platform.
- *Application input:*
  Application input (which is subject to floor control) may come from any platform.
- *Application execution:*
  A shared application may run on any platform.

The Multimedia Application Sharing Service is divided in an Application Sharing Service (ApSS) dealing with text/graphic data and an Audio/Video Sharing Service (AVSS) dealing with audio/video data.

2.2.1 Text/Graphic Sharing

The ApSS is realized by a distributed X multiplexor (X wedge) [8] which is mainly formed by the pseudo server, the pseudo client and the translators (The components are called pseudo server and pseudo clients since they behave like an X server or X clients from the point of view of the application and the terminals, respectively).
The pseudo server is the central component of the application sharing service. It is responsible for

establishing connections to the pseudo clients and for keeping track of the X resources created during the session. When a new participant is joining, the archived information is used to set up a series of X requests that brings the new participant's X server into the required current state. The requests are sent to the pseudo server which forwards them to the new pseudo client and to the associated X server respectively.
Pseudo clients reside on terminal's site. There is one pseudo client for each terminal. Pseudo clients are dynamically created upon request by a pseudo server and destroyed when not being used any longer.
X was chosen as the basic interchange protocol for application data because X is a network-transparent, device-independent windowing and graphics system, which is currently supported by most leading workstation manufacturers. For non-X windowing systems (e.g. Macintosh QuickDraw), translators map the non-X protocol to the X protocol and vice versa. Implementing non-X to X translators is a complex but feasible task [12]. For X to non-X translation, commercially available X servers are used. There are commercially available tools which provide for non-X to X translation (e.g. XGator for Macintosh). However, we do not know a tool which translates single applications. They rather translate the whole of a Macintosh screen. Also, they do not have any mechanisms to filter input or control the floor.
For the Macintosh and the PC platform, the required functionality for QuickDraw/QuickTime resp. MS-Windows to X translation is the following:

- *Output translation:*
  The translator has to intercept calls to the Macintosh Toolbox/MS Windows (output from an application) in order to translate them into X requests and send them to the pseudo server.
- *Input translation:*
  X events (input to an application) coming from the pseudo client have to be translated into Macintosh/MS Window events which subsequently are read from the event queue by the application as its input.

2.2.2 Audio and Video Sharing

In addition to text and graphical information, the JVTOS multimedia application sharing service allows the sharing of continous media. Multimedia applications dealing with continuous media for input or output are based on system specific interfaces enabling the access to the multimedia extension of the operating system [11]. For JVTOS, the continuous media interfaces shown in Table are supported. JVTOS exploits functionality offered by these interface to realize the sharing of multimedia data.

Sun Sparcstation	Xplx (Parallax XVideo Extension), audio device ("/dev/audio")
Siemens-Nixdorf RW 420	SGI Video extension, audio device ("/dev/audio")
Apple Macintosh	QuickTime (selected components)
IBM PC	MME (Multimedia Extension to Windows), MCI (Media Control Interface)

Table 1: Supported multimedia extensions

In order to provide multimedia application sharing, the standard continuous media access points have to be modified such that they convert and distribute the data stream to the real interface among the machines of the other session members and collect and convert the data streams produced by these machines [5].

2.3 Telepointing

Communication between JVTOS users which is based on sharing information generated by an application, is enhanced by providing globally visible pointing tools, termed telepointers or telemarkers. Its purpose is to monitor movements of pointers owned by a user and to display these movements locally and at remote user sites. In addition, the service tracks the position of the mouse pointer of the floor holder and mirrors it on remote displays in the corresponding shared windows. Each user may own a set of telepointers and move these on his display. A user's telepointer becomes visible to others as soon as it enters a shared window, i.e. as soon as it could point to an object which is jointly viewed and possibly referred to in an audio/video conversation.

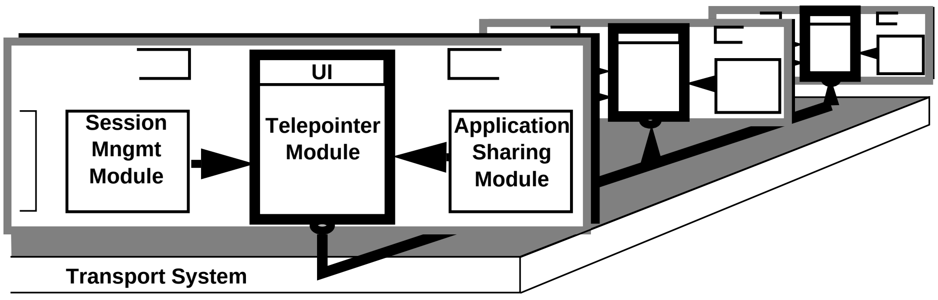

Figure 2: Telepointer Architecture

The Telepointer Service relies on support from other JVTOS services. From the session management it receives address information of session participants and the identity of the current floor holder. From the sharing service it receives the list of shared windows. Thus, the service is able to determine when a telepointer is to be "shared" and when so, with whom. A JVTOS user controls telepointing through a user interface offered by the Telepointer Service. Each user is allowed to create one or more telepointers. Besides telemarkers having a rectangular shape containing an arrow-like pattern, further telepointer shapes with a transparent background are provided. However, their use is restricted to the windowing systems that support transparency.
The Telepointer Service is implemented in a distributed way in order to reduce the end-to-end delays incurred by various processing layers (Figure 2).

2.4 Picturephone

JVTOS includes a Picturephone, which provides interpersonal audiovisual communication among JVTOS users. The Picturephone supports two-way audio and two-way video streams between all participants of a JVTOS session.
During a session, the Picturephone probably runs most of the time. The Picturephone must be smart enough to allow other multimedia application to run in parallel without degrading their quality.

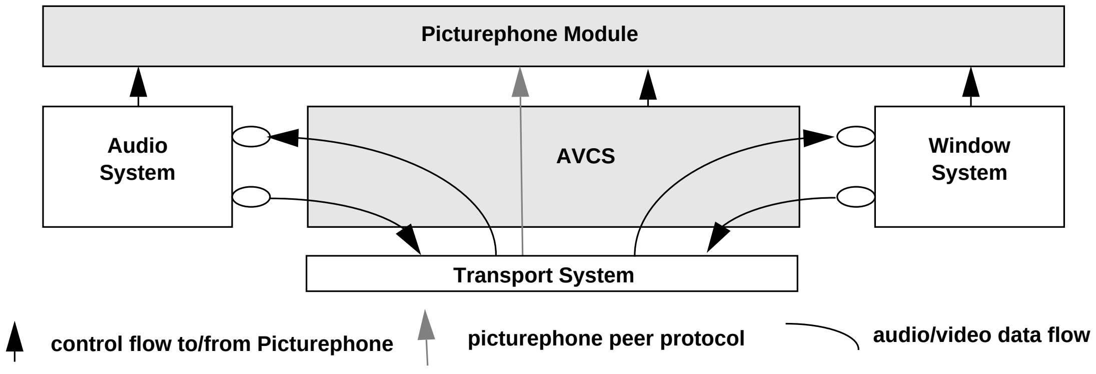

Figure 3: Picturephone Architecture

On each platform, the user interface is adapted to the native look-and-feel of the target window system, but the functions offered are identical. The Session Management Service supplies information about the session participants enabling the Picturephone to adjust itself according to changes during a session. For all session participants, status information is shown indicating connectivity, visibility, and audibility. There is a gain control for each audio channel and also a gain control for the

sum of all audio channels. Each video window also displays an indication to which session participant it belongs. If the underlying hardware supports adjustments for contrast, hue, and brightness, the JVTOS user is able to modify these values via the user interface.
The Picturephone is implemented in a distributed way (Figure 3). On each participant's machine there is a Picturephone module. Rather than transferring data between sources and sinks, the Picturephone module is merely responsible for organizing the data transfer between remote locations. At the same time it offers the user interface to the local user allowing it to control this transfer. The data transfer itself is done by the AVCS.

3 Implementation Aspects

3.1 Overview on the Implementation Structure

UNIX. The UNIX operating system provides a pre-emptive multitasking environment for the active applications. Thus the JVTOS services can be implemented as a set of concurrently executing UNIX processes. So each of the blocks in Figure 4 is implemented as at least one process. These processes communicate using available UNIX interprocess communication facilities.

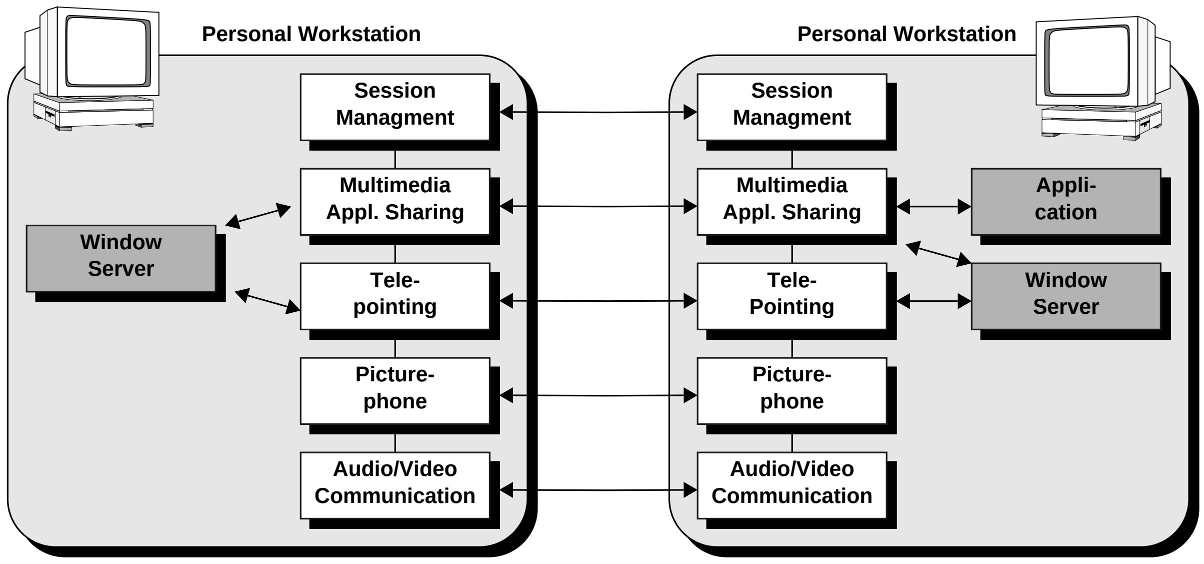

Figure 4: Implementation Structure on the UNIX Platforms

MAC. The Macintosh operating system provides a cooperative multitasking environment for the active applications. This means that applications determine on their own how much time they use and how much processing time remains for other applications and background processes. Application do not get a guaranteed part of the processing time.

JVTOS as a telecooperation tool is used like a system level service and not like an application. Thus in a typical application scenario, JVTOS runs in the background while a shared application is in the foreground. In this case the cooperative multitasking system does not guarantee processing time for JVTOS and its services. In order to make enough processing time available for JVTOS, the JVTOS services are implemented as interrupt driven modules (Figure 5).
In general, JVTOS services are accessed by subroutine calls. In the case of drivers they are accessed by driver control calls. Messages of JVTOS services to higher levels of the hierarchy are delivered asynchronously by use of callback routines. In the same way the low-level services react asynchronously to events that are produced by the transport system in case of data arrival from remote instances of JVTOS services.

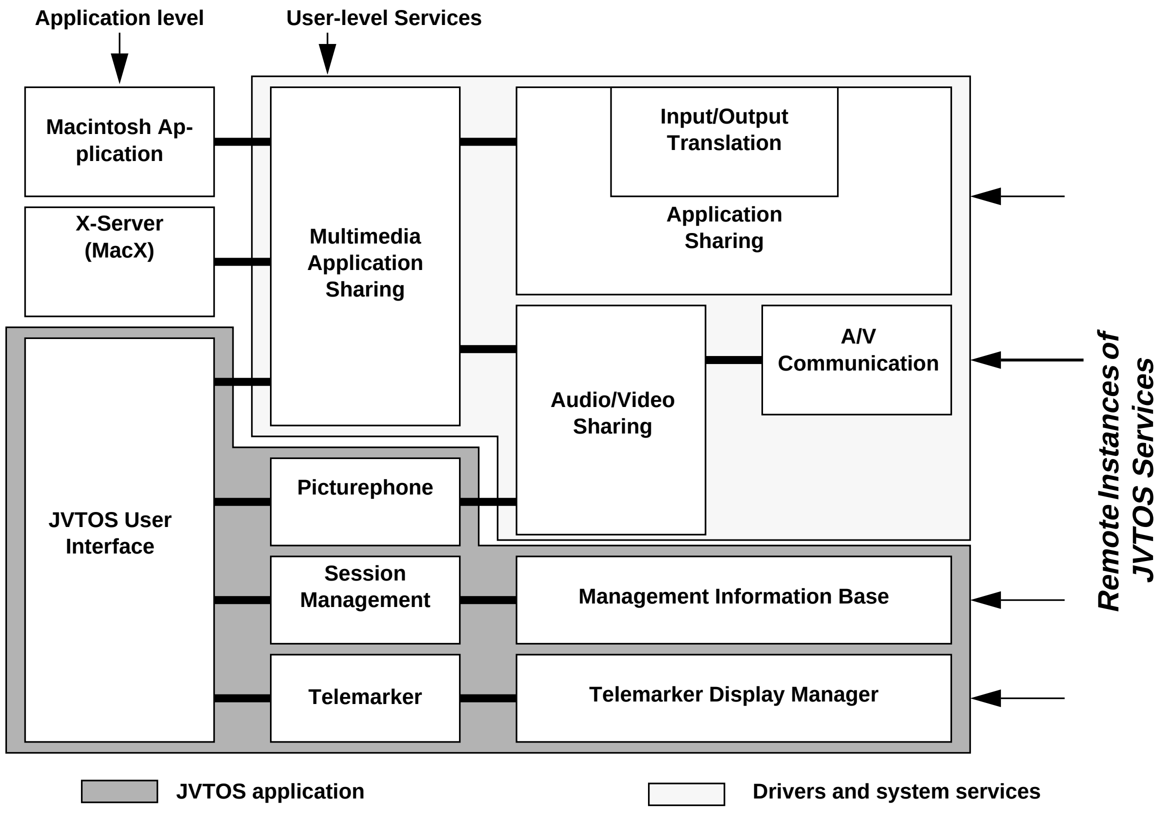

Figure 5: Implementation Structure on the MAC platform

PC. On the PC platform, there are four different processes running. The Picturephone, the session management module, and WinX which has two processes, the interceptor and the translator. These processes communicate using the standard MS Windows interprocess communication protocol Dynamic Data Exchange DDE which allows applications to communicate and "share" memory objects. The Session Management contains the participant/chairman User Interface, the telepointer which is implemented as a Dynamic Link Library DLL, as well the MIB for Floor Control purposes. The SM Module sets up a daemon to listen for incoming calls or to initiate a session when using the chairman capability, then checks for the existence of the Picturephone "atoms" to establish DDE communication. If they are not present it starts the Picturephone and then sets internally the Multimedia capabilities accordingly. The application brings up the session control panel. The telepointer user interface is triggered when an application is being shared as well as the shared application control part is added to the SM User Interface.
The WinX module consists of two parts. The interceptor is implemented as DLL that intercepts from the MS Windows system and filters the messages for the shared applications. The translator is an application that (1) translates the messages received from the interceptor into X messages and (2) sends them to the X wedge as well as (3) filters the remote input messages to the shared applications. The communication between these processes is achieved by using one "atom" to identify the *JVTOS Interceptor* and another "atom" for the translator called *JVTOS Translator*. The JVTOS translator sends a DDE message initiating the communication, afterwards DDE messages flow back and forth between these modules.
The Picturephone is a stand-alone application that sets up a daemon waiting for incoming calls and opens a local video window (if the hardware is present) and creates the "Atom" *JVTOS Video* which is used as the "Topic" for DDE communication with other JVTOS applications, the same procedure applies to the local audio channel and the "Atom" is *JVTOS Audio*. If none of the hardware requirements are satisfied, the application will not start, meaning of course that the PC Workstation is not able to use the multimedia capabilities.

3.2 Multi-Media Application Sharing

3.2.1 Text/Graphic Sharing

UNIX. On Unix machines the main components of the X wedge are realized as different processes running simultaneously (Figure 6).
An AS daemon runs on every site potentially taking part in JVTOS. The AS daemon listens for incoming requests to initiate local pseudo server or pseudo client processes.

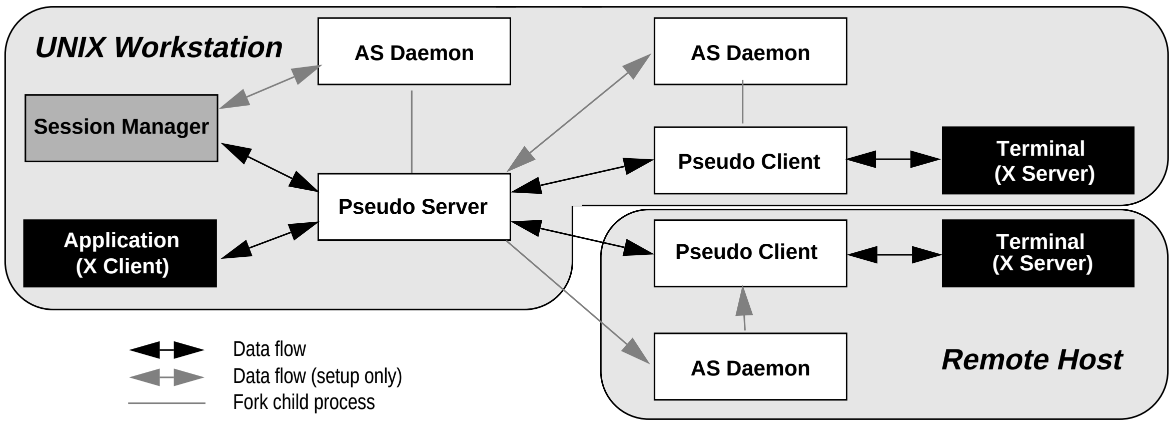

Figure 6: Application sharing in the UNIX environment

MAC. Quickdraw is not network transparent like the X Window System. It is not even network aware and does not use any network transport system. This functionality has to be added in order to make Macintosh applications sharable. Therefore the graphics output of Quickdraw is intercepted, analyzed and translated to the X protocol (Figure 7).

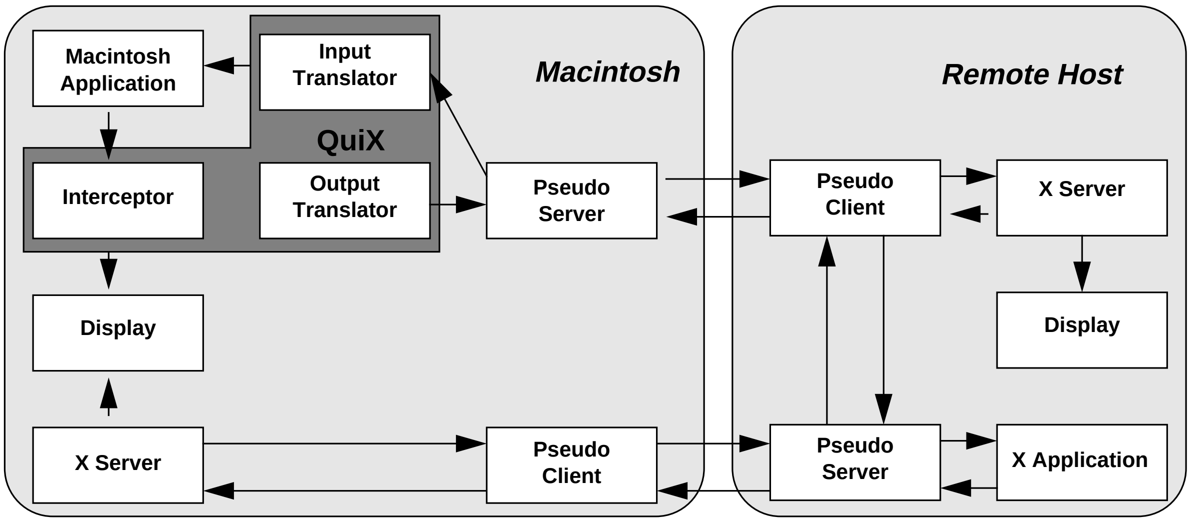

Figure 7: Sharing Macintosh Applications in an X environment

Quickdraw like all other graphics systems consists of a large number of graphics requests. But the lower interface to the graphics system consists of some bottlenecks where all graphics operations pass through before they are finally executed. At this point the graphics output may be intercepted and redirected to a translator. The Quickdraw to X translator (QuiX) analyses all graphics requests and translates them to the X protocol. In this process it has to be considered, that not all graphics

operations may directly be translated to X protocol requests. Some operations have to be converted to a sequence of X requests. On the other hand in some cases sequences of Quickdraw operations have to be combined in order to translate them to X Protocol requests. The resulting stream of X Protocol requests is sent to the pseudo server in order to be distributed to other participants of the session.
Input of remote users of shared applications like mouse movement and keyboard input, which the translator receives from the application sharing service are converted to Macintosh specific event records and sent to the local Macintosh applications

PC. MS Windows is intended for *single machine use*, it does not provide any means for simple communication between computers over a network. The method JVTOS uses to share MS Windows applications is similar to the method used by the Mac when making its applications sharable. Since MS Windows does not require the use of an underlying network, it assumes that all its applications are running on the local machine. Hence, there is no quick and effective way to tell the windowing system to display the application on a remote host. To overcome this setback, applications which are to be shared, require the use of an *interceptor*. This interceptor has the ability to capture requests, sent by the application to MS Windows, before they arrive at their final destination.

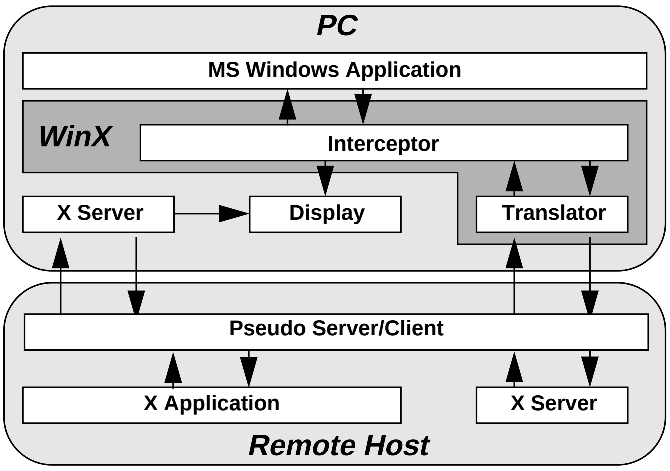

Figure 8: Sharing MS Windows applications in a X environment

Using the interceptor alone, it is possible to have all application requests captured and sent to other systems running MS Windows. The remote systems would then accept these request, as if they were generated locally, and act on the display accordingly. However, it is a well known fact that JVTOS is a multi-platform service. Hence, there is a possibility of having countless different architectures involved in a single session. This demands that the sharing of applications is centered around the X Protocol.
It is at this point where WinX (Windows to X translator) enters the scene. WinX works together with the interceptor to produce this X Protocol. This is done by having the interceptor relay all application requests to both MS Windows and WinX. By passing these requests to the local windowing system, the application is displayed on the local machine as if nothing had happened. However, when WinX gets hold of these requests it translates them into their *equivalent* in X Protocol and pass them along to the Pseudo-Server running on a remote host (Figure 8).
It should be noted that there are over 1000 different MS Windows calls, and most of them do not have an equivalent in X Protocol. Therefore, a scheme had to be derived to “mix and match” these two protocols. This means that a single MS Window message may give way to a set of X Protocol requests or a single X Protocol request may lead to several MS Windows messages. At the moment, WinX only makes use of a few of these messages.

3.2.2 Audio/Video Sharing

UNIX. The distribution of the audio data stream and the required simultaneous running of the Picturephone application and an audio application simultaneously leads to a modification of the Sun audio device in such a way that it may be opened by several applications at once. This implies also that mixing of different simultaneous write accesses to the audio device and the multiplexing of read operations is handled somewhere behind this interface (Figure 9).

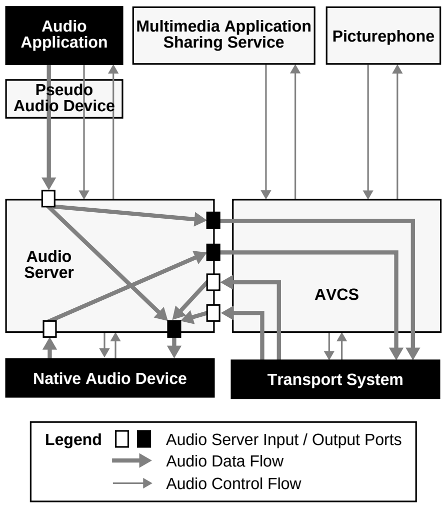

Figure 9: Scenario of using an audio application and the Picturephone simultaneously

The Audio Server 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 is connected to multiple output ports, multiplexing is done. Coding conversions mediate between different audio codings. In order to differentiate between the ports, a local input/output port (original audio device) and application input/output ports are used.
The *Audio/Video Communication Service (AVCS)* hides the complexity of audio/video processing in the endsystem behind an application programming interface (API) which provides simple functions to transfer continuous media 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. The AVCS is capable of multipeer communication. It negotiates codings and sets up transport connections to remote AVCS entities.

The X wedge informs the AVSS (Figure 10) about the application which should be shared. By receipt of the application top-level window ID the AVSS is able to resolve any child windows and it can check, whether video is shown in one of these windows. The AVCS transports a video data stream from a local video window to a remote instance of on AVCS, which forwards it to the remote instance of an AVSS. Feeding the data pipe and displaying the video are local tasks at the two endsystems. The remote AVSS is responsible to define a sink address for the video data stream (a window ID).

MAC. Audio and video output of applications is not presented by Quickdraw, but by a multimedia extension called Quicktime. Quicktime works with audio and video data. The structure of Quicktime is very modular. Quicktime contains a large number of software components, which deal with the data streams.As Quickdraw, Quicktime is not network aware. Video and audio data are presented directly on the local hardware. It is neither possible to specify an alternative device nor another host. In order to distribute video and audio the output has to be intercepted and redirected as with the distribution of standard graphics (Figure 11). There are only a small number of components which output audio and video data. These software components are bottlenecks for all audio and video streams. They can be replaced by pseudo components which call the original system components and additionally distribute the data streams via the AVSS to other participants.

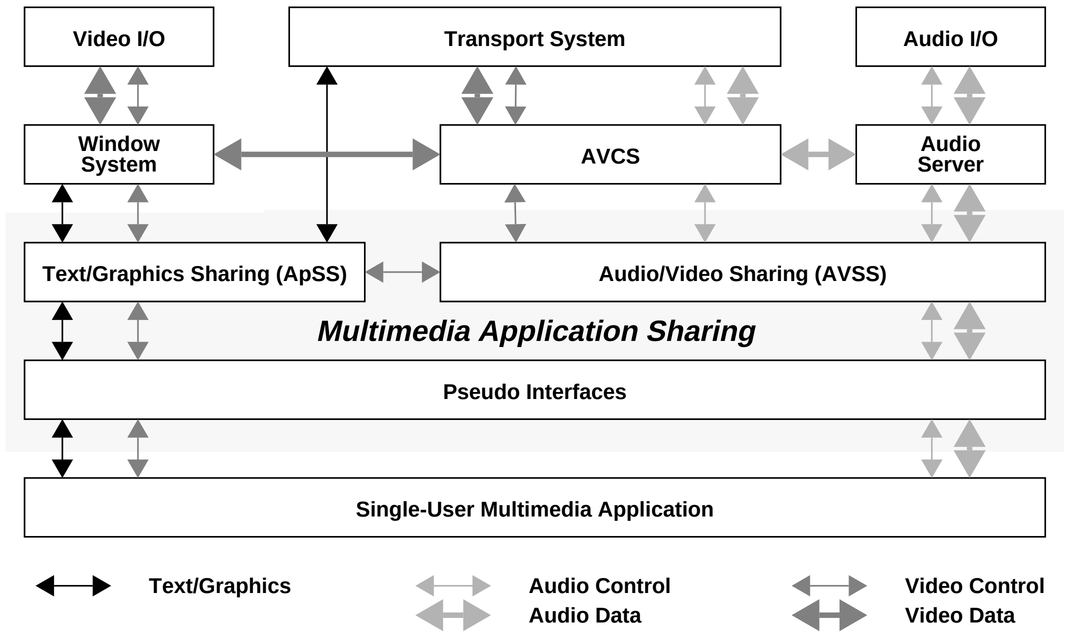

Figure 10: Audio/Video sharing data flow

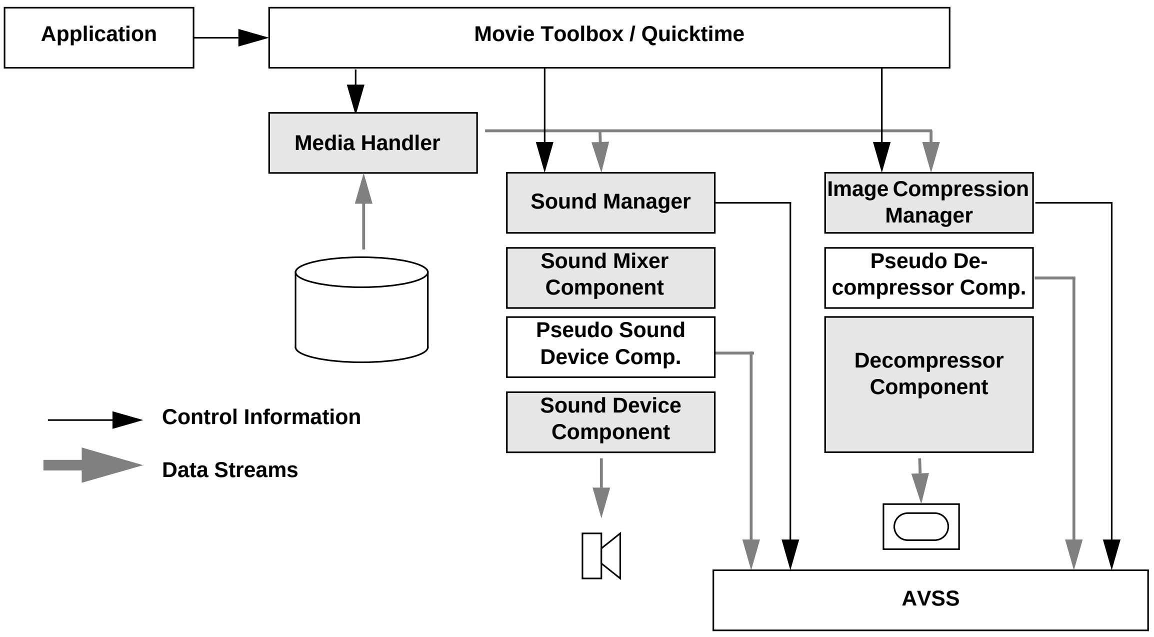

Figure 11: A shared Quicktime application plays a compressed video with sound

PC. Similarly to what was done for WinX, an additional interceptor has to be developed for audio/video data. However, this interceptor does not intercept MS Windows messages but instead calls to and from the actual device drivers. To accomplish this, the interceptor has to listen to two different communication paths; one being from the MS Windows System Software and the other being the much less used 'direct' path from the actual application (Figure 12).

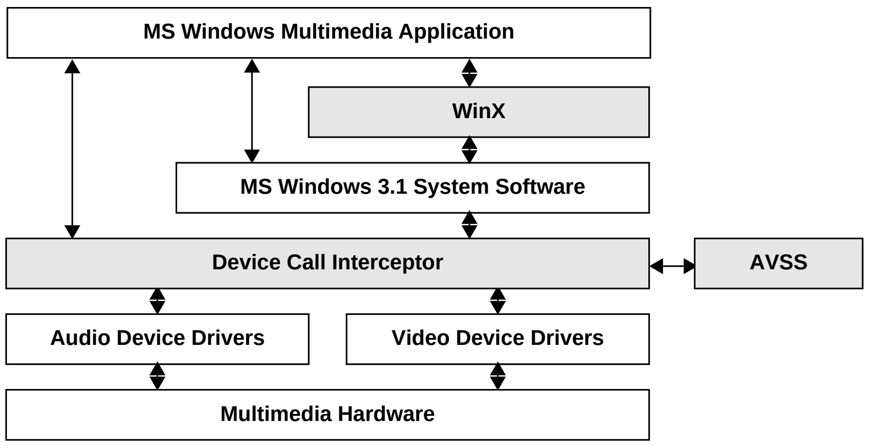

Figure 12: Path of multimedia data within a JVTOS session in a PC

As mentioned, this interceptor captures raw data instead of MS Windows messages. For example, it intercepts the JPEG encoded video right before it reaches the driver and corresponding hardware decompression. In addition, since the data is coded according to well established standards, there is no need to make use of a translator. It is then the task of the audio/video sharing service to send this data to the other participating JVTOS users.

4 Summary and Conclusions

This paper describes the implementation of a system supporting cooperative work. Unlike similar available systems, JVTOS does this support in a heterogenous and multimedia environment. Heterogeneity is present in the flavor of the used platform and applications.
JVTOS runs on four different hardware platforms using different operating and window systems: SUN/SNI (OSF/Motif), PC (MS-Windows) and MAC(QuickDraw).
Shared applications may be initiated on a workstation of any type. Such applications use the window system of the workstation they run on, but it still is possible to interact with such applications using a workstation of any of the other types. The key to this functionality is our use of the X protocol with video extensions as an interchange protocol and the concept of translators.
Multimedia support is not restricted to the provision of just a Picturephone. Instead, sharing of multimedia applications is also possible, with the implication that another element of heterogeneity has to be taken care of, the handling of different kinds of multimedia toolboxes associated with the four platforms.
Since JVTOS integrates different endsystems over different high-performance network technologies the domain of telecooperation is opened to a wider group of potential users and thus potential collaborators.

5 References

[1] 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.

[2] Michael Altenhofen: “Erweiterung eines Fenstersystems für Tutoring-Funktionen”. *Diploma Thesis at Universität Karlsruhe.* Karlsruhe, 1990.

[3] 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.

[4] Bauerfeld W.: RACE-Project CIO (R2060): Coordination, Implementation and Operation of Multimedia Tele-Services on Top of a Common Communication Platform; *International Workshop on Advanced Communications and Applications for High Speed Networks ‘92,* pp. 401 - 405, 1992.

[5] Gabriel Dermler, Thomas Gutekunst, Edgar Ostrowski, Frank Ruge: “Sharing Audio/Video Applications among Heterogeneous Platforms” accepted at the 5th IEEE COMSOC Workshop Multimedia ‘94, Kyoto, Japan, May 1994.

[6] Gabriel Dermler, Thomas Gutekunst, Bernhard Plattner, Edgar Ostrowski, Frank Ruge, Michael Weber: “Constructing a Distributed Joint Viewing and Teleoperation Service in a Heterogeneous Workstation Environment”. *Proceedings, 4th IEEE Workshop on Future Trend of Distributed Systems in the 1990’s, Lisboa, Sept. 93*

[7] 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.

[8] Thomas Gutekunst, Bernhard Plattner: “Sharing Multimedia Applications among Heterogeneous Workstation”. *Proceedings, Second International Conference on Broadband Islands.* Athens, 1993.

[9] 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.

[10] Greg McFarlane: “Xmux - A system for computer supported collaborative work”. *Proceedings, 1st Australian Multi-Media Communications, Applications & Technology Workshop.* Sydney, 1991

[11] 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.

[12] Miroslav Vodslon: “Feasibility of Translating Native Windowing Systems to X Window”. *Internal Report of RACE/CIO WP 4.2.* Berlin, 1992.

[13] Weber, M., Schmidt T., Luna L.: “Programmers Guide to the JVTOS Management Information Base”. *Internal Report of RACE/CIO WP* 4.2. Saarbrücken, 1993.