JVTOS - A Reference Model for a New Multimedia Service

G. Dermler[a] and K. Froitzheim[b]

[a]Institute of Parallel and Distributed High-Performance Systems, University of Stuttgart, Breitwiesenstr. 20-22, 7000 Stuttgart 80, Germany

[b]Department of Distributed Systems, University of Ulm, Oberer Eselsberg, 7900 Ulm, Germany

Abstract

Multimedia Joint Viewing and Teleoperation is a new service for highspeed networks. It allows physically separated users to work in a collaborative fashion with documents and applications. It is however not a cooperative application but a system level service exporting the functionality of applications to many workstations.

Keyword Codes: C.2.4; D.4.4; H.5.1
Keywords: Distributed Systems; Communications Management; Multimedia Information Systems

1. Introduction

Advances in communications and computer technology provide the technical foundation to integrate digital audio and video into today's distributed information processing systems. This is of great advantage for many application areas, especially for those, where users communicate and cooperate to work on a common task. Joint Viewing (JV) systems are considered to be the basis for many applications in the field of real-time computer-supported cooperative work [ElGi91], such as desktop conferencing [Cr90], joint-editing [FiKr88] and electronic classroom applications [ScCe87]. They provide the basic mechanisms for jointly viewing and manipulating multimedia information, user synchronization and user interaction.

Proposed or implemented JV-based systems differ in a number of ways. Some of them only support a restricted number of specific cooperation tools with dedicated computer applications [SaGr85], [StBo87], while others allow the sharing of a wide range of cooperation-unaware applications among multiple users [Fa91], [WaFe91]. In both cases interaction between users and applications is usually restricted to the exchange of text and graphics data only. In those cases where audio and video are incorporated as well, dedicated solutions have been implemented [Cr90]. Furthermore, solutions usually imply the existence of a homogeneous implementation platform, for instance with respect to the operating system used.

“Joint Viewing and Teleoperation” (JVTOS) is a multimedia collaborative service developed within the RACE II project “Coordination, Implementation and Operation of Multimedia Services” (CIO). Its goal is to provide an open multimedia communications platform as a system-level service which allows to realize a wide range of CSCW scenarios. It provides open communications in that it interconnects computers with different hardware and operating systems. Moreover, it allows the use of existing (cooperation-unaware) applications in a JV fashion. At the same time, JVTOS can be employed as a toolkit providing multimedia communication functions to future applications realizing new cooperation schemes. The integration of these capabilities into one service is the main feature and benefit of JVTOS. Its implementation can be expected to facilitate both the development and the use of multimedia cooperation across heterogeneous distributed systems.

In this paper we present a reference model describing the functional structure of JVTOS which will serve as the basis for implementing our prototype [Cio1692]. In Section 2 we first introduce the concept of JVTOS by describing a typical JV scenario and a basic architecture comprising the functional components of a JV environment. The JVTOS functionality as provided by the core JVTOS component is covered in Section 3. Then, in Section 4 an architectural model is presented which defines the functional structure of a JVTOS design.

2. Concepts of JVTOS

2.1. JVTOS Scenario

JVTOS was conceived to support the computer-based equivalent of a personal meeting, where people convene to jointly work on a specific topic, for example by exchanging visual and audible information. JVTOS allows these users to stay physically at their workstation and travel electronically.

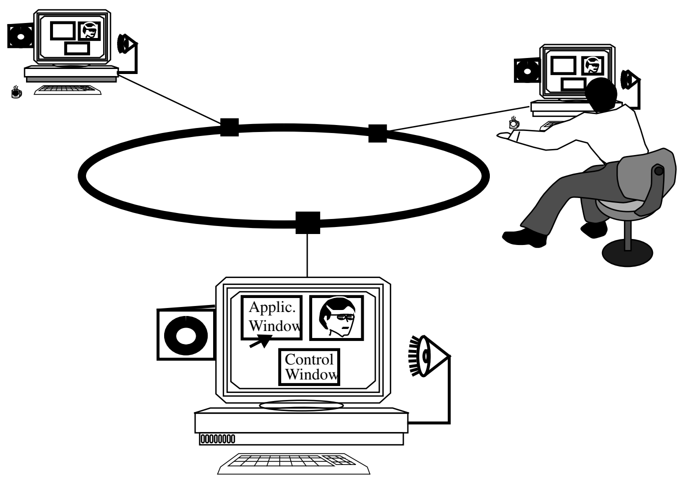

Figure 2.1: JVTOS scenario

JVTOS offers these features by providing a distributed computer-based environment, in which users can jointly view the output of multimedia applications and control and feed the applications by generating input to them. Users communicate among each other by employing corresponding applications providing audio and video links and by using globally visible pointing tools to mark objects jointly viewed.

A typical JVTOS scenario is shown in Figure 2.1. It involves joint viewing of application windows, the use of a pointing tool to point within the window. User-to-user communication is enhanced via a videophone application shared through JVTOS. Each of the users is additionally provided with a control window to manipulate the state of a JVTOS session. Another scenario is joint work with multimedia applications, for example preparing a movie with video editing software. JVTOS enables artists at different sites to use the program by transmitting multimedia output and input.

The JVTOS scenario assumes that windowing systems are used as graphical user interfaces. It allows for a heterogeneous implementation on different workstation types running different operating and windowing systems. Other groups within the CIO project supply a unified transport service platform hiding the heterogeneity of the underlying network technologies. Currently it is intended to implement JVTOS on Sun workstations, Macintosh computers, PCs, and perhaps other Unix-based workstations. Accordingly, the windowing systems to be supported will be X, MacOS and MS-Windows.

2.2. Basic Structure

In order to provide the user-level functions presented above, entities representing the functional blocks of a JVTOS implementation are defined. An initial classification reveals two groups: modules incorporating the core JVTOS functionality and entities augmenting it to end-to-end interaction relationships between users and applications. The first is referred to as the JVTOS Manager and is the centerpiece of a JVTOS implementation. The latter comprises applications, multimedia terminals, and transport systems. Figure 2.2 shows the basic relation between these entities.

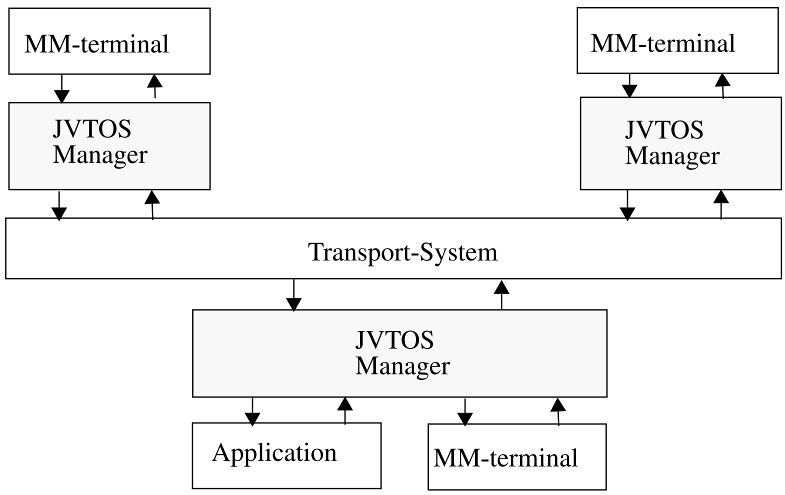

Figure 2.2: Basic architecture of JVTOS

JVTOS supports the use of cooperation aware and unaware multimedia applications. It offers its functionality to *cooperation aware* applications through an application programmers interface (API) allowing access to JVTOS functions controlling the exchange of different data streams between users and applications. Various degrees of integration allow for the development of distributed applications. To support *cooperation unaware* (standard 'off the shelf') applications JVTOS implements invisible features such as distribution of application output to a number of users and coordination of the input from the various participants. Classic single media (text/graphics) applications are included as special cases. In the case of the Macintosh this would be implemented as an operating system extension intercepting calls to the graphic system or the screen driver and transmitting these requests to other participants. In the X-Window world this is achieved through a special X-Server called Shared X Bridge, which multiplies the received application output and simulates multiple X-Clients [Fa91].

The centerpiece of the service is the JVTOS Manager implementing cooperation related functions such as session management, telepointers, floor control, synchronization, etc.. These functions are described in more detail in the next section. The JVTOS Manager in turn uses the CIO Transport Service to move the multimedia data across high speed networks like ATM, FDDI, or DQDB.

JVTOS capable terminals have to be equipped with the appropriate multimedia devices for input and output of graphics, audio, and video. Passive stations need at least a high quality graphics display (color or greyscale) and a loudspeaker. Due to the synchronous nature of the JVTOS service, the throughput of the workstation is crucial. It is essential that any high speed data, especially video, is copied only once in the computer. The workstation should have a high performance internal bus system and efficient interrupt handling. Furthermore high volume datastreams should be moved over the bus in their compressed representation. Figure 2.3 shows a typical workstation architecture suitable for multimedia JVTOS. Comprehensive descriptions of the requirements to a JVTOS multimedia workstation and to the CIO transport system can be found in [Cio1092].

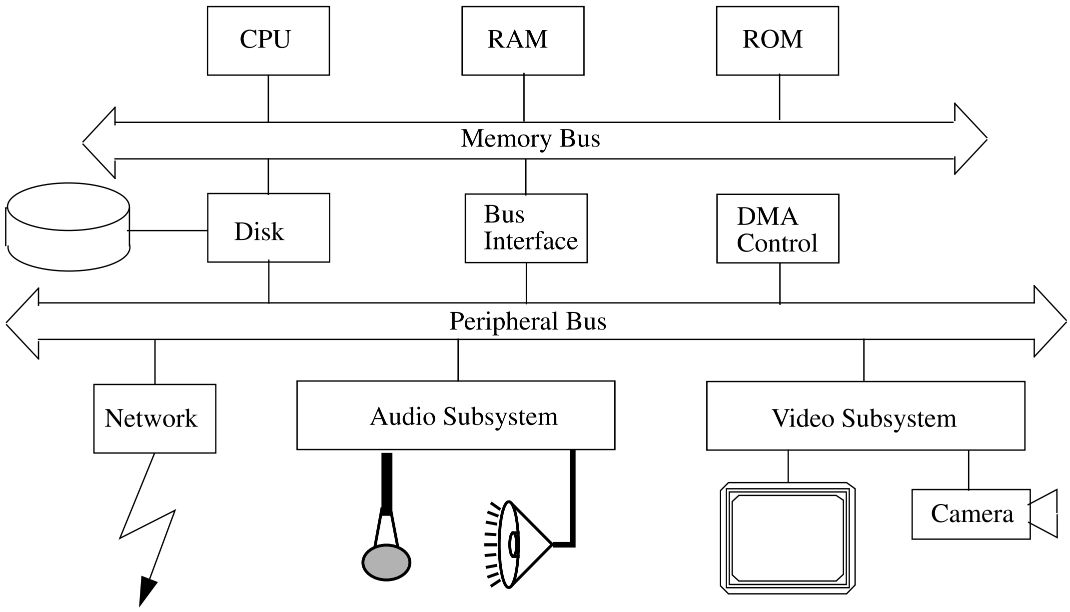

Figure 2.3: Multimedia JVTOS Workstation Architecture

3. Functions of the JVTOS Manager

3.1. Overview

The JVTOS Manager manages data streams and supplies the necessary JVTOS control functions including the user interface. It implements the structure of interaction in a JVTOS session. This structure depends on the user selected joint-viewing situation with respect to an application (who sees what, who generates which data streams, who decides on changes,...).

The choice of such a joint-viewing pattern is supported in the JVTOS Manager by control functions (Session Management, Floor Control). Other functions relate to support for handling different media and their synchronization. The following functions are provided:

- Session Management,
- Input and Floor Control,
- Shared Windows,
- Telepointer,
- Audio/Video Processing, and
- Synchronized Presentation.

3.2. Session Management

"Session" denotes a communication relationship between users and applications. This relationship evolves under user control. The set of operation users can perform on the session state is referred to as Session Management operations. The way users agree on whether or not to perform these operations is called Session Policy. We will give examples for the operations, a basic export/import paradigm and a few session policy examples.

- Initiation and termination of a session,
- Participation control (joining and leaving a session) and state queries (participating users, availability of Shared Windows),
- Export of data streams like application windows, video, and audio,
- Set-up of Shared Window instances, and
- Selection of policies (Session Policy, Floor Control policy).

The export/import paradigm divides the set-up of end-to-end communication links into two steps. This approach supports in a natural way a one source / many destinations configuration which may change dynamically during a session.

Session policy can be based on roles assigned to users or automated mechanisms. A specialized protocol is defined to support role changes.

Example1: One user is assigned the role of a chairman. Whenever a new user wants to join a session, the chairman decides whether to pull him into the session.

Example2: One user wants to initiate an application. All users decide by voting.

3.3. Input and Floor Control

Input is generated by a user and directed to an application. It can be physically local (from within the same workstation) or remote, i.e. from another workstation on the network. Since we allow for multimedia applications, the input can belong to any supported medium:

- Text/graphics input in the usual sense,
- Audio, video, etc.,
- Mouse movement and button state information,
- Shared Window resizing information, and
- other events.

In a live meeting, only one person has the right to speak at any given time or confusion occurs. In a JVTOS environment with cooperation unaware applications, it is even more desirable that only one user is generating input at a time. Hence the notion of the Floor as the right to direct input to the JVTOS application and Floor Control to determine which JVTOS participant has this right, i.e. is current Floor Holder.

There are several methods of implementing Floor Control policies. They range from simple oral agreements to elaborate transaction-syntax based protocols.

Example: One method of Floor Control is what we call Explicit Token-passing. Control of the floor is represented by the possession of a token, and participants can request the floor-token. At any point, the current holder of the token (and hence the floor) can be requested to yield the floor. The usual recovery procedures assure continued operation of the system when a workstation fails to operate properly.

3.4. Shared Windows

3.4.1 Concept

By jointly viewing Shared Windows, JVTOS users share the physical representation (e.g. display of text, graphics, video, ...) of both the application output and of related tools (Telepointers, Mouse Pointer). A Shared Window is identified by the application behind it, i.e. two users viewing instances of the same Shared Window interact with the same application. Applications may open more than one window. In this case, an appropriate set of windows might be shared as a whole.

The output of cooperation unaware applications results in identical contents in Shared Window instances on the participating workstations. This may be different for cooperation aware applications.

The Shared Window concept makes no restrictions to the content's type. The type will be determined by the shared application. Presently used textprocessing applications produce only text and graphics output, upcoming multimedia document applications will add voice annotations and videoclips to the document content. Applications like video editing, video recording, or video multiplexing for conferencing may handle recorded or real-time video.

3.4.2 Operations on Shared Windows

Each user may move his Shared Window instance to another position on his screen. The movement is of local significance, i.e. it does not have any influence on other instances of the same Shared Window located on other users' displays.

The same applies to resizing a Shared Window by a user who is not holding the floor. He may resize as he wants, but neither is the application notified nor is any other instance of the Shared Window resized. This operation might result locally in clipping some parts of the Shared Window or in having empty parts in the Shared Window instance.

The situation is different for the Floor Holder. His resizing information is considered as input to the application. Thus the effect of resizing is global, i.e. all instances of the Shared Window should be resized accordingly.

3.5. Telepointing

Telepointers are a means of communications between users. They use Telepointers to point at objects in Shared Windows. A user's Telepointer is part of a Shared Window (and visible in all instances of the Shared Window), if it enters the local instance of the Shared Window. Otherwise it is only visible on the local display and not in other instances of the Shared Window. The concept of Telepointers is separate from that of the Mouse Pointer. However, the mouse pointer could be used to select and control one of the available telepointers in order to operate with it in a Shared Window.

One user may have many Telepointers. All active Telepointers could be visible at once. Different shapes and/or colors for the Telepointers should facilitate the correlation between Telepointers and users.

A set of several Telepointers with different functionality ('Telemarkers') is even more useful. Each user, whether possessing the floor or not, has access to a palette of Telemarker tools. The tools could comprise a pointing tool, a highlighting tool, a post-it note tool, a set of different framing tools, etc. These tools would reside on an annotation overlay on top of the Shared Window. The mouse pointer is again used to select one of the available telemarkers and to operate them in a Shared Window.

3.6. Audio and Video

Audio and video streams are generated by applications controlling input (digitization) devices such as a microphone, a tape player, or a camera respectively. Thus, two way audio and video communication between users has to be provided by a corresponding application and not by the JVTOS-Manager itself. It is up to the application to configure the connectivity to user terminals. For this purpose it may interact with the Session Management module.

A user/application connection will traverse the chain terminal - JVTOS-Manager - application in one direction. The provided quality depends on each of these entities. For audio it can range from telephony quality up to Compact Disk (CD) quality sound.

Additional functionality like compression/decompression might be provided by this module or be assigned to terminal or application entities. Typical codings are G.711, G.721, G.722, MUSICAM (MPEG Audio).

According to the role of video in the context of a particular video application (e.g. videophone, one-way-links) different video codings and qualities are supported. Video communication is not necessarily symmetric (different codings/qualities per direction, different window sizes, ...). The video module in the JVTOS Manager will perform at least the conversion between different codings used by different terminals and applications. Additional functionality like compression/decompression might be provided by this module or be assigned to terminal or application entities. Possible codings are: JPEG based proprietary video protocol, H.261, MPEG I.

3.7. Synchronization

3.7.1 Application Output

In the JVTOS-scenario both users and applications act as sources of interrelated information streams destined for presentation at user sites. By presentation synchronization we understand the enforcing of a temporal relationship between information streams at the site and time of their presentation. The following is required:

- The use of an application providing a one-way audio link combined with a one-way video link requires lip-synchronization. The end-to-end delay is not restricted explicitly, it has to be the same for both media.
- Audio- or Videophone applications presume a real-time conferencing situation, meaning that what is generated somewhere is perceived at virtually the same time at the connected user sites. In this case the round-trip delay is restricted to something called non-noticeable delay.
- The use of audio/video links and of telepointers to discuss the content of a Shared Windows is consistent if every user is presented with the same Shared Window content at any given time. Larger presentation time variations than with audio and video may be tolerable, if they do not exceed a certain level.

  In a non-conferencing situation (one master talking and visible), it is required that at a slave site the content of the Shared Window instance is that of the Shared Window instance at the master site at the time when the master audio/video input currently played was generated.

- The presentation of multiple media in a Shared Window instance requires synchronization between the involved streams.
- The presentation of two or more Shared Windows, produced by either one or more applications, might also be requested to be synchronized.
- Any continuous medium requires intrastream synchronization (periodic output of samples).

3.7.2 Application input

A general JVTOS application is considered to be able to process multimedia input generated by a remote user. The synchroneity of the input media might be part of the information the application has to process. This requires JVTOS

- to present the input synchronized to the application at the site of the application or
- to provide information about the temporal relationship of input streams to the application.

3.7.3 Implicit versus Explicit Synchronization

Implicit synchronization is given if end-to-end communication is fast enough to yield the impression of things happening at the same time globally, i.e. restricting delays to be non noticeable. This kind of synchronisation imposes very strict requirements on the transport system and the multimedia components. However, it does not require any mechanisms relating to synchronisation between streams.

Explicit Synchronization relies on synchronisation functions handling explicit information about the temporal relationship between streams.

3.8. Translators

Both applications and terminals exchange data streams of different media with the underlying JVTOS Manager across interfaces using a set of data exchange primitives and data formats. This Reference Model defines neither the primitives nor the data formats. It only requires that for the exchange of data between distributed parts of the JVTOS Manager (at user or application sites) one set of primitives and data formats is to be used. This set is referred to as the JVTOS interchange format. End-to-end communication across different terminal or application interface formats (primitives + data formats) is supported by introducing the concept of a translator as a gateway between a specific format and the JVTOS interchange format.

The interchange format of JVTOS for text/graphics will be the X protocol of the X-Window system. Multimedia data will be handled in JVTOS proprietary formats to be specified. In order to give users of systems not capable of the X protocol or using proprietary windowing systems the possibility of taking part in JVTOS sessions, text/graphics translators to/from the X-protocol are required. In X-Window terminology a text/graphics translator is a protocol gateway between the X protocol and another windowing system (e.g. MS-Windows, QuickDraw). The main task of this gateway is to find matching representations between parts of these protocol elements.

4. The JVTOS Architectural Model

The architectural model consists of the functional modules onto which the concepts and functionalities described in the previous section are mapped. It embeds the functional modules into a unified structure making use of degrees of freedom left by the functional description, such as the distribution of functional modules to physical locations and the sequence of functional modules with respect to data flow.

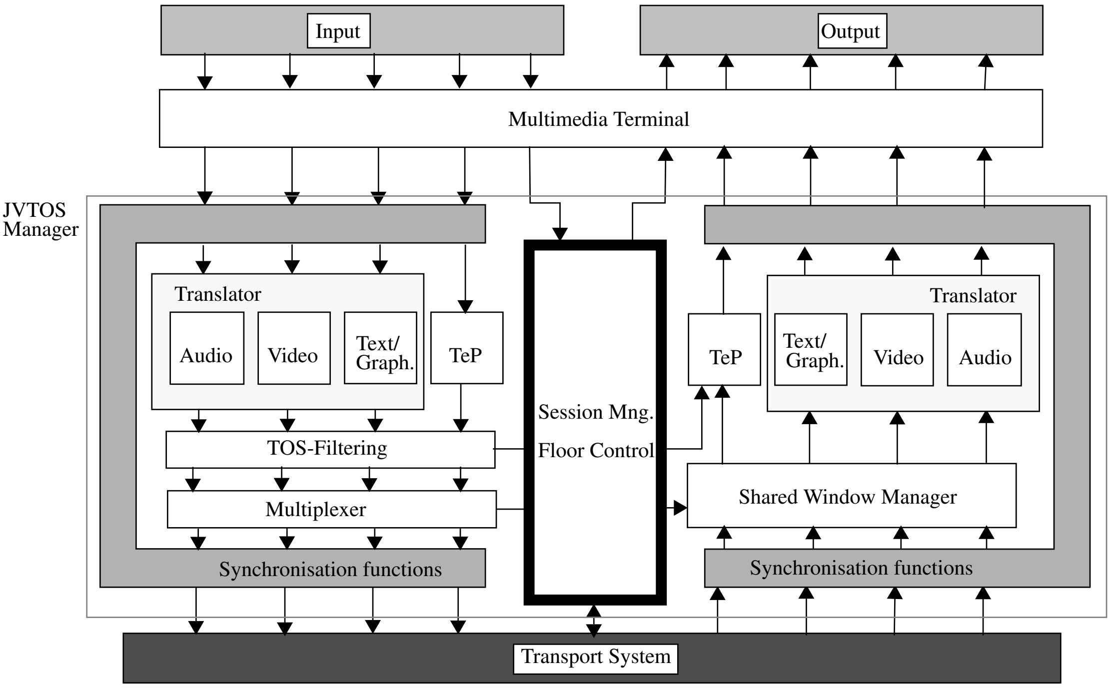

Figure 4.1: JVTOS Architectural Model - User site

4.1. Basic configuration

This configuration is based on the client/server paradigm as commonly used in distributed environments. It considers two types of sites:

- sites where users (clients) generate and consume multimedia streams by operating their multimedia terminals (figure 4.1),
- sites where applications (servers) are running (figure 4.2).

Note that no control flow between the modules of the JVTOS Manager is depicted.

This separation fits the envisaged use of JVTOS as an interconnection service between applications and users located on different machines. The case of an application running at a user site implies an architectural model which can be easily derived from the one defined for the basic configuration.

Figures 4.1 and 4.2 capture different possible mappings of functional modules onto site types. Providing a module at one site type may make the use of the module at the other site type unnecessary. The possible alternatives are mentioned in the next section in the context of the modules concerned.

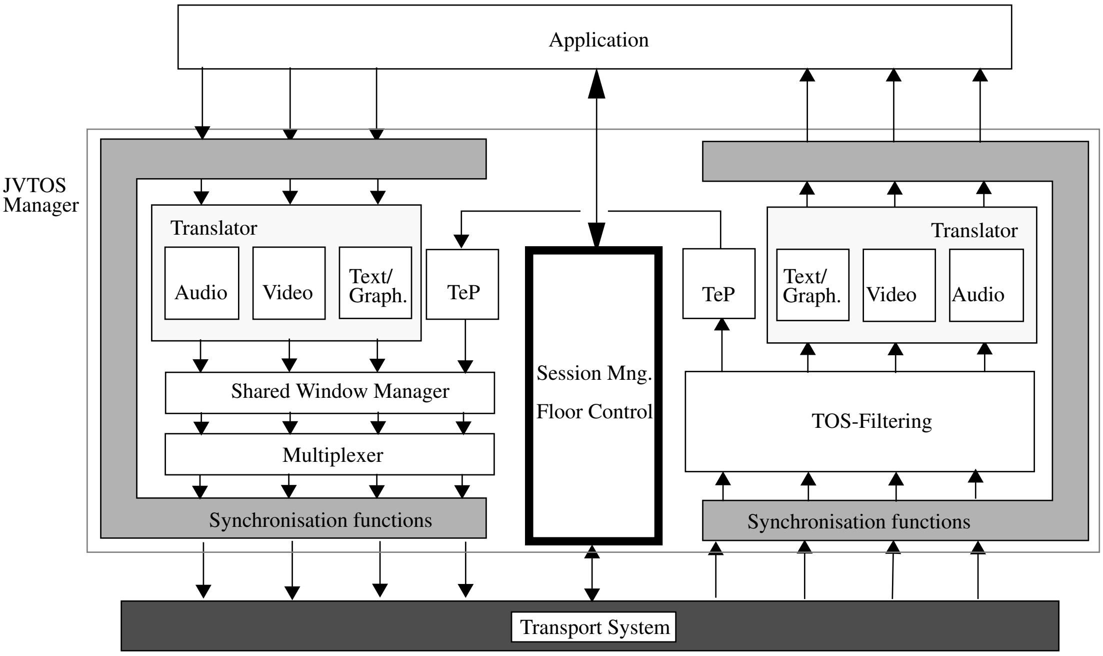

Figure 4.2: JVTOS Architectural Model - Application site

4.2. JVTOS modules

Two types of modules are to be considered. The choice of a pattern of joint-viewing (who generates what data, who sees what,...) is supported in the JVTOS Manager by *control modules* (Session Management, Floor Control). The other functional modules embody *basic mechanisms* which can be used by the control modules to implement different patterns of joint-viewing.

Among the controlled modules we differentiate as follows:

- Modules dedicated to a single communication medium (audio, video, text/graphics) handle the medium specific functionality required as support to connected terminals and applications. These modules convert between different data and primitive formats and are therefore called translators.
- Modules related to specialized presentation tools (Telepointers) take the corresponding input and generate the appropriate output on the displays of the participating stations.
- Cooperation related modules (TOS filtering, Multiplexer, Shared Window Manager,) are responsible for the distribution of data streams and the composition of single data streams representing parts of the shared window into a consistent shared window content.
- The synchronization module provides for the required temporal synchronisation between single data streams.

4.2.1 Shared Window Manager

This module collects all data streams carrying information about the elements of a Shared Window (application output, Telepointers, Mouse Pointer of Floor Holder) and correlates this information, such that all Shared Window instances present the same correct view. Alternatively, one may say it collects the elementary information and builds compound presentation objects called 'Shared Window instances'. Correlation is needed since instances of a shared window may be in a different state. For example if the shared window instance belonging to a user is iconified, the corresponding telepointer must not be displayed. Similarly, correct positioning of a Telepointer inside a window depends on the position of that specific window on the screen. The result of the correlation might be for each instance a single stream carrying all Shared Window information. It also may result in several streams, where the information in each stream accounts for the correlation.

The Shared Window Manager is depicted in the Architectural Model at both the user and the application site. The described correlation can be done at either of these sites. If correlation is to be done at the application site, the involved streams have to be available there, i.e. streams generated at user sites (Telepointers, Mouse Pointer of Floor Holder) have to be directed to the Shared Window Manager at the application site. If correlation is done at a user site, relevant elementary streams have to be available there. The correlation at a user site only relates to the local Shared Window instance.

4.2.2 Teleoperation Service Filtering (TOS Filtering)

This module decides which of the data streams generated by users are to be used only locally or are to be forwarded either as input to an application or to a Shared Window Manager module in order to become part of a Shared Window. More precisely, the module has the following tasks:

- It filters out and passes to the shared application the input of the participant who owns the floor from those other participants who do not. The information required to identify the Floor Holder is provided by the Floor Control.
- It copies the mouse pointer movement information from the input generated by the Floor Holder, if the Mouse Pointer is within the Shared Window instance. The information is then passed to all involved Shared Window Manager modules. Otherwise the mouse pointer information is passed on for local display only.
- It prevents Telepointer movement information from becoming shared, if the Telepointer is not within a Shared Window. In this case the information is passed on for local display. Otherwise the Telepointer information is forwarded to all involved Shared Window Manager module for correlation.

The TOS-filtering module is depicted at both the user and the application site. All types of filtering can be done at either site. If it is done at a user's site, it only concerns the input streams generated locally.

4.2.3 Multiplexer

This module is essential for realizing the exchange of data streams in one source/many destinations configurations. More precisely, it marks the point from where - based on an input stream - a data stream is made available for each destination. Subsequent processing is done in the context of the destination the data stream is intended for. If the Multiplexer is the last processing stage at a site, its functionality is merely that of a distributor providing for the multitude of streams to be transported.

4.2.4 Translator

Two types of translators are needed:

- translators between the windowing systems (X to Mac, Mac to X, X to Windows, Windows to X). Figures 4.1 and 4.2 shows them as text/graphics translators. They are typically implemented in software.

- Media translators for video or audio require higher processing power. Especially video compression and decompression is typically located in hardware.

4.2.5 Telepointer

This module handles the input concerning telepointer movements and provides for display of the telepointers, such that each telepointer can be easily associated with its owner. It is also responsible for displaying the Mouse Pointer of a remote Floor Holder. The functionality of this module may be provided at the user sites or in a centralized fashion at an application site.

4.2.6 Synchronization

We expect that explicit synchronisation mechanisms have to be provided in order to ensure synchronized presentation of multiple data streams. In this case synchronization information (SI) on the temporal relationship of multiple streams has to be established, converted, transferred, passed for further processing to an application or used to enforce synchronized presentation.

Establishing SI means the transformation of given synchronicity into a representation form not subject to corruption due to delays (e.g. by inserting synchronisation marks into stream). An additional synchronization 'track' might be transmitted in a separate channel of the transport system. The SI has to be transferred to the site where it is to be used (for resynchronization or as input to an application).

The following considerations led to the U-shaped area in the Architectural Model:

- Ideally the information about an existing temporal relationship is gathered as soon as possible after the generation of the data and used as close as possible to the actual presentation of the data at the destination sites, thus minimizing corruption due to processing components delay.
- The transport system is expected to be one of the major sources for corruption due to delays. In order to preserve interstream synchroneity across the transport system stream interleaving could be used.
- If data presentation is to be synchronized across different sites a distributed synchronisation control will be required.

4.2.7 Control Module

This module provides a shell to JVTOS users for Session Management and Floor Control operations.

The Session Management part executes these operations by using the control mechanisms provided by the other functional modules. It furthermore enforces the selected session policy. Operations to be performed are among others:

- initiation/termination of any other module
- the initiation/termination of interaction between local modules
- initiation/termination of communication between modules at user and application sites:
  - informing the Multiplexer modules at the user and application sites of streams to be multiplexed for multiple destinations
  - informing TOS Filtering modules of streams to be filtered (Floor Control does this only for application input).

This functionality may be also used by cooperation-aware applications in order to dynamically set up or remove links to users.

The Floor Control module informs the TOS-filtering module which streams are generated by the current Floor Holder.

4.3. Module interworking example

In figure 4.3 the data flow is shown for the case of two user sites jointly viewing an application through a Shared Window. A single medium (text/graphics) is used and both users have Telepointers, but only the Telepointer of user 2 is within the local Shared Window instance.

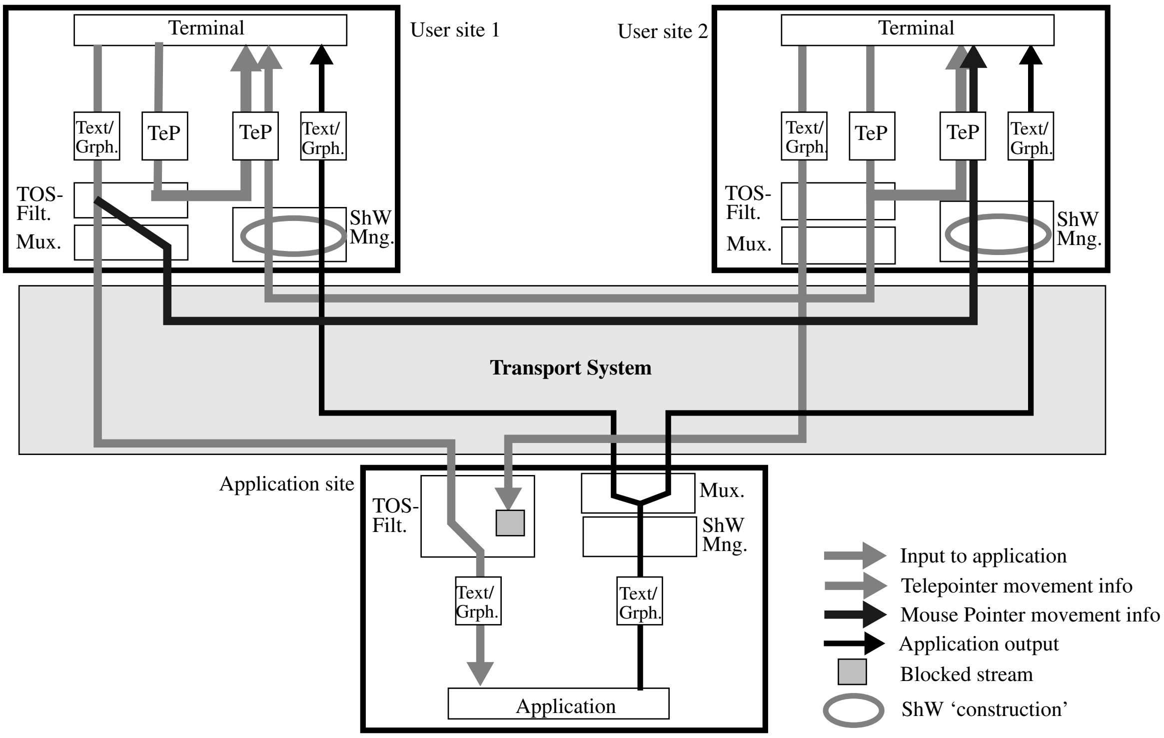

Figure 4.3: Module interworking example - decentralised Shared Window correlation

User 1 is the Floor Holder and his Mouse Pointer is within his instance of the Shared Window. No synchronisation is provided and no Telepointer modules at the application site are used. The Session Management and Floor Control modules are omitted, since they only provide for control flow.

The application input of user 1 is directed to the application, since he is the Floor Holder. The input of user 2 is filtered out (blocked), in this example by the TOS-filter at the application site.

The Telepointer information of user 1 is only used for local display. The Telepointer information of user 2 is used for local display and additionally sent to user 1. The Mouse Pointer information of user 1 is identified and copied out of the input of user 1 and transferred to the Shared Window Manager of user 2. The Mouse Pointer information of user 2 is not distributed, since user 2 is not holding the Floor. This example assumes that the display of local mouse pointers is provided automatically by the terminals.

The application output is distributed by the Multiplexer at the application site to both user sites. The correlation of the Shared Window element streams (the 'construction' of the Shared Window) is done at the user sites 1 and 2 by the corresponding Shared Window Managers, and only for the respective local Shared Window instance.

5. Conclusions and Future Work

The given reference model is flexible enough to accommodate many usage patterns allowing for a broad application of the service. Existing services for cooperative work force users to give up their favorite application (word processor, drawing tool, presentation program). JVTOS however will work with the installed base of existing software packages. We expect a much higher acceptance by the users when they can continue to use their favourite applications invisibly extended by JVTOS. The implementation of cooperation aware JVTOS application will be the second step fully exploiting the JVTOS functionality.

The CIO JVTOS working group will implement a prototype service based on this reference model and the architecture given above. The first implementation will have limited functionality and it will run on SUN Sparcstations and the Apple Macintosh. In a second step we will add more functions and port the implementation to PCs running Windows.

6. Acknowledgements

The presented Reference Model is in significant parts the result of joint work done in the CIO JVTOS working group. In particular, we wish to acknowledge the contributions provided by our colleagues Edgar Ostrowski, Bernhard Plattner, Thomas Schmidt, Guenter Schulze, Miroslav Vodslon, and Sicke Westerdijk.

7. References

[ElGi91] C.A. Ellis, S.J. Gibbs, G.L. Rein. Groupware - Some Issues and Experiences. Communications of the ACM, January 1991, 39 - 58.

[Cr90] T. Crowley, et al. MMConf: An Infrastructure for Building Shared Multimedia Applications. Proc. of the 3rd Conference on CSCW. ACM New York, 1990.

[FiKr88] R. Fish, R. Kraut, M. Leland, M. Cohen. Quilt: A Collaborative Tool for Cooperative Writing. Proc. of the Conf. on Office Information Systems. ACM New York, 1988.

[ScCe87] J.A. Scigliano, B.A. Centini, D.L. Joslyn. A real-time Unix-based Electronic Classroom. Proc. of the IEEE Southeastcon '87. IEEE, NewYork, 1987.

[SaGr85] S. Sarin, I. Greif. Computerbased Realtime Conferenceing Systems. IEEE Computer 18, 10 (October 1985), 33 - 45.

[StBo87] M. Stefik, D.G. Bobrow, G. Foster, S. Lanning, D. Tartar. WYSIWIS revised: Early Experiences with Multiuser Interfaces. ACM Transactions on Office Information Systems, 5, 2 (April 1987), 147 - 186.

[Fa91] G. McFarlane. Xmux - A System for CSCW. Proc. of the 1st Australian Multimedia Communications, Applications & Technology Workshop. Sidney, 1991.

[WaFe91] H.M. Abdel-Wahab, M.A. Feit. XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration. Proc. of the IEEE Conf. on Communications Software: Communications for Distributed Applications and

Systems. Chaple Hill, 1991.

[Cio1692] CIO WP4.2. Deliverable 16: Requirements to a Joint Viewing and Teleoperation Service. R2060. Berlin, 1992.

[Cio1092] CIO WP4.2. Deliverable 10b: Requirements of Isochronous Audio/Data/video Service on the Communication Platform. R2060. Berlin, 1992.