1st International Workshop on Community Networking: Integrated Multimedia Services to the Home, San Francisco, July 1994.

A Terminal-Based Approach to Multimedia Service Provision

Christian Blum, Refik Molva, Erich Rütsche[1]
Institut Eurécom
2229, route des Crêtes, 06904 Sophia-Antipolis, France
E-mail: {blum, molva, ruetsche}@eurecom.fr

Abstract - We describe an architecture that supports networked multimedia applications. The architecture is based on a separation of multimedia stream processing and application processing. Applications reside in *application pools* whereas media stream processing is performed in *multimedia terminals*. An application controls a set of terminals that exchange multimedia data streams directly between each other. A tele-teaching and a home shopping example demonstrate important aspects of our architecture.

I. INTRODUCTION

Current platforms for community networking are based on PCs and on simple terminals such as the French Minitel. To bring multimedia services to the home these devices must be enhanced to multimedia terminals that offer the user a window to the multimedia services and applications in the network. Services will evolve from retrieval services such as video-on-demand to interactive services such as video-conferences and games.

To provide these multimedia services an architecture is required that supports a wide variety of foreseeable services and that can be extended to support upcoming services with new features. Most of the multimedia architectures proposed in literature build on powerful workstations that communicate as peer-systems. Such architectures are not adequate for community networking because they do not support service diversity. For multimedia communication to be successful in community networking, an approach similar to the *Intelligent Network* concept [1] is required, where terminals with simple interfaces access a wide variety of services that are installed on servers within the network. Corresponding to the Global Functional Plane in the Intelligent Network architecture, there must be a high-level implementation platform that eases the introduction of new multimedia services to the community network.

Customer premises equipment must be optimized for multimedia communication and especially for the processing of digital audio and video that have high requirements on the CPU and on the I/O bandwidth. The customer equipment must be able to guarantee a quality of service (QoS) for every processed medium stream.

1. The work of Erich Rütsche was funded by the Swiss National Fund

Although the focus of a new multimedia architecture is on communication rather than processing it should not neglect traditional networked applications. A multimedia application often combines communication and processing, for which computer supported collaborative work (CSCW) may serve as an example.

We propose an architecture that supports fast service deployment and that is optimized for multimedia communication without sacrificing traditional applications. Since most multimedia applications do not process medium content it is possible to divide them into a media processing part and an application processing part. In our architecture, media processing is performed by multimedia terminals that are optimized for this purpose. Besides media acquisition, presentation and transport they also implement the user interface that allows a user to interact with applications. Applications run on central application pools within the network. They implement the services that are offered to users. Application pools provide an application programming interface to manage applications and build multiparty sessions between terminals. Media data are directly exchanged between terminals, while the control of the application and of the terminal is concentrated on the application pool. The control and data interfaces of our architecture are extensible to make it flexible towards technical evolution.

In the next section we give an overview of the architecture and its main components. The architecture of the terminal is discussed in detail in Section III. Section IV describes the control flows between terminal and application. Section V presents two examples, tele-teaching and home shopping, that demonstrate how services are deployed on top of our architecture. We conclude with a discussion of the architecture and the state of our work.

II. ARCHITECTURE OVERVIEW

Requirements of multimedia in terms of system architecture, operating system and networking are quite different from those of conventional applications. It is not trivial to come up with a system that accommodates both multimedia stream processing and application processing in an efficient way.

Most current systems are based on multimedia-enhanced workstations. Workstations are designed for a very broad range of applications; adding multimedia to such a general system is likely to compromise the quality of the multimedia services or the performance of the other applications. Multimedia will finally be done on systems that are especially conceived for this purpose and that are similar in spirit to the one proposed in the Pegasus project [2]. Such systems will integrate multimedia and traditional application processing into a single workstation and might eventually become the standard for future research and engineering environments. They will not be adequate for public multimedia service provision because they will be expensive and because they require the user to install software on her station for every service she wants to use. This approach is not acceptable for community networking where a high number of users want access to rapidly changing and evolving services provided by multiple independent parties.

We are developing an architecture that should be efficient in terms of multimedia stream processing and that supports the introduction of new multimedia applications or services. Such a system can then be deployed in both a research and a public environment.

The following provides a closer look at the nature of multimedia applications. Proceeding from this a new multimedia system architecture is proposed.

A. Towards a New Architecture

A multimedia application can be classified according to the way it treats media content. There are some applications in science and engineering that analyze media content and that take the result of such an analysis directly as control input, with possibly no human user being involved [3]. Other applications treat the various media as transparent streams whose content is only meaningful to human users. Such applications are still indirectly influenced by media content simply because the human user will be influenced by it in the way she interacts with the application. The majority of distributed multimedia applications that are known today belong to the second category and do not look at media content at all. This is especially true for applications whose purpose is human-to-human communication or interactive multimedia information retrieval, and, in our opinion, for all multimedia services that are to be offered to a greater public in the near future.

Applications that do not process media content can be divided into a media part and an application part. Application part and media part can be represented by entities that communicate by means of a control protocol. The protocol entity on the media side executes media stream related application requests as part of control protocol procedures. The introduction of a control protocol allows to completely insulate application processing from media processing and in fact perform the two at different locations. It is then conceivable to have a terminal that implements the media part and that is remotely controlled by an application whose location is hidden to the terminal user. This terminal may communicate with other terminals that are controlled by the same application. For the provision of multimedia services to the public exactly such an approach has to be taken. What is needed in addition is a programming interface on the application side that supports application development and thereby service diversity.

The set of available media is limited and applications tend to deploy media streams in a similar fashion. It is thus straightforward to have a programming interface that offers a high-level control over commonly used media streams. Such an interface is for instance presented in the Touring Machine [4]. A programming interface allows to concentrate on application design issues rather than media stream handling and will therefore speed up application development considerably.

The separation of media and application processing and a programming interface on the application side are the keys to our architecture.

B. A Terminal-Based Multimedia Architecture

In our architecture we physically separate the multimedia application from its media streams. Applications reside on *application pools* inside a network, whereas media stream processing is performed on *multimedia terminals* in the periphery. A multimedia terminal is optimized for the processing of media streams and offers a standard control and data interface to the outside. An application pool runs applications on top of an interface that implements features for the control of terminals. Two protocols provide the link between application and terminal. An *application management protocol* offers control on application level. It includes procedures for start, modification and release of applications. A *connection control protocol* allows an application to establish and modify connection structures among participating terminals and to control media acquisition and rendering in each of them.

The basic elements of our architecture are depicted in Figure 1. The central entity of a terminal is the *terminal control.* The terminal control uses the application management protocol to communicate with the central entity of the application pool, the *application manager.* The *connection control* in the terminal handles single connections or connection groups with synchronized connections and does not have knowledge about the application as a whole. It is directly accessed via the connection control protocol and handles requests of applications that are authorized by the terminal control. The connection control invokes a *stream handler* for every media connection that it establishes. A stream handler implements the complete processing path of a single stream within the terminal, i.e., it does acquisition and transmission on the source side and reception, synchronization and presentation on the sink side of a stream.

In the application pool, applications are started and released by the application manager following the corresponding procedures in the application management protocol. An applica-

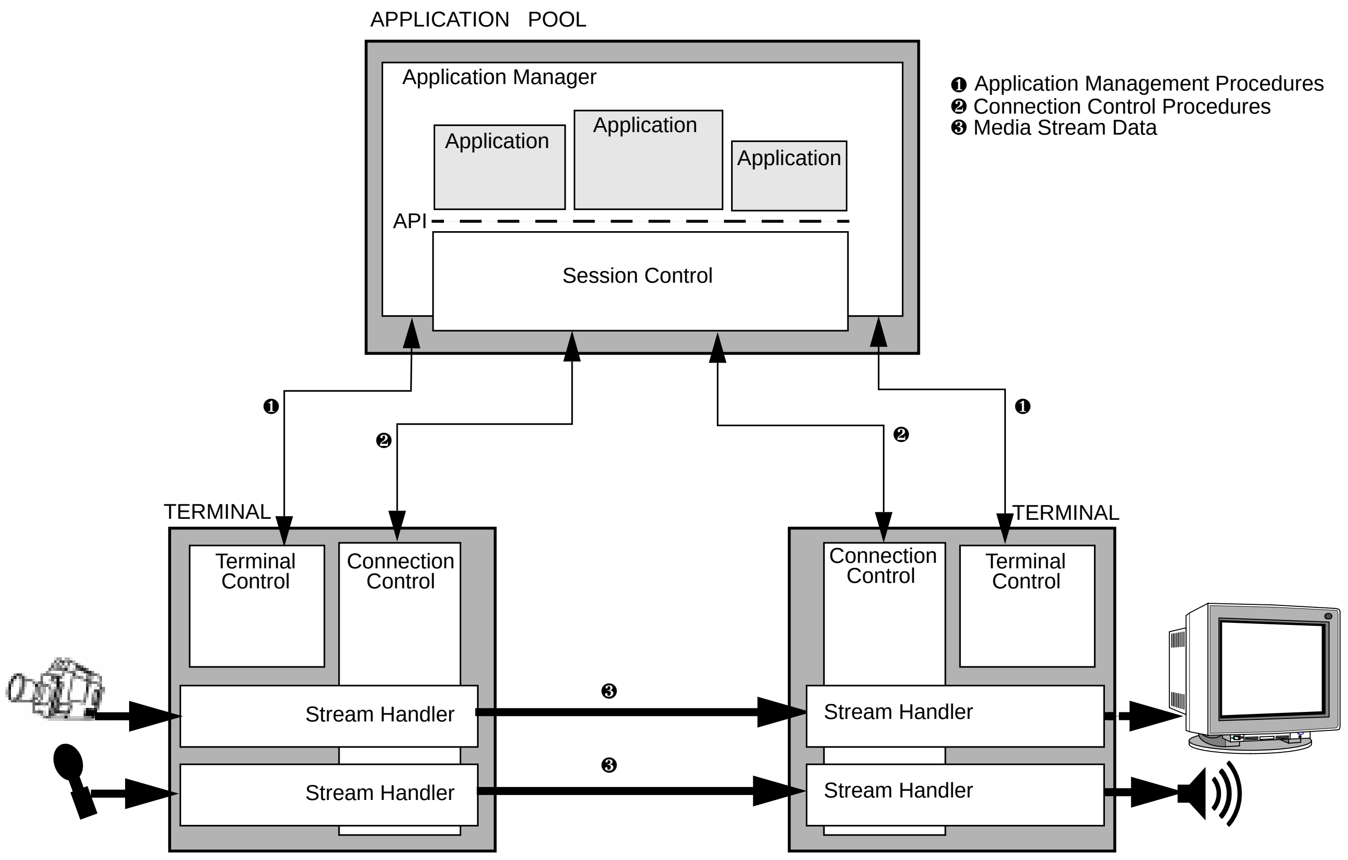

Fig. 1. The General Architecture Showing one Application Pool and two Multimedia Terminals (API = Application Programming Interface).

tion programming interface (API) provides the services of the *session control* and of the application manager to the application. The session control translates requests for the establishment of complex connection structures among participants into simple connection control primitives towards the respective terminals. In addition to connections between terminals, applications will also establish connections *with* terminals. Interactive applications will use such connections for their direct communication with terminal users, but they might also serve to convey small amounts of multimedia data to terminals. High volume multimedia data, e.g., video, on the other hand are not supposed to originate from the application pool. They bypass the application pool on connections in-between terminals, as indicated in Figure 1.

C. *The Multimedia Terminal*

The terminal control of a multimedia terminal processes user requests to start applications on remote application pools. It also receives invitations to participate in ongoing multiparty applications and forwards them to the user who may accept or reject them. A terminal may participate in various applications at a time. These applications may reside in different application pools. The terminal control maintains an application management connection to every application pool from which it has applications running. Before the terminal control starts a new application it first checks if the system has enough resources for this application. If so it reserves the resources that are requested by the new application.

The connection control processes connection establishment, modification and release requests from the application side. The basic control protocol procedure is an operation invocation for which the connection control returns a positive or negative operation result. The connection control also generates error events towards the application in case a medium connection perceives serious QoS problems.

The multimedia terminal has to be very flexible in the way it deals with media streams and their synchronization. It must export as much control over its features as possible, thus allowing applications to control all aspects of media stream processing.

The connection control protocol reflects the capabilities and resources that a terminal may have. It is designed in a way that it does not restrict the evolution of terminals, i.e., it is possible to integrate new terminal features into this protocol.

The control interface of the terminal is specified by the connection control protocol and the application management protocol. In addition to this a data interface has to be specified, which corresponds to a specification of every media stream that is handled by the terminal. The variable parameters of the media stream specification will be part of the connection control protocol.

A terminal is the source or sink of multimedia streams. Therefore, all major multimedia sources or sinks, e.g., a multimedia file server, have the same control and data interfaces as multimedia terminals. A multimedia terminal will in the general case also have storage capabilities and in some applications take the role of a file server towards other terminals.

D. *Application Pools*

The application manager in the application pool processes requests from terminals that want to start applications and from applications that want to invite terminals to participate. Running applications have the possibility to request the start of other applications which is for instance necessary in the case where a yellow page application wants to start a service that was selected by a user. Applications are only started after the negotiation procedure in the application management protocol has made clear that a terminal is compatible with an application.

Applications run on top of the session control. The session control views an application as a collection of connections and terminals and does not deal with application specific semantics. It offers a high-level interface to the applications that allows applications to establish complex application structures with single requests. The session control decomposes the request for a complex connection structure into a number of connection control requests towards concerned terminals. An application may for instance request the full interconnection of its participating terminals with video connections. The session control will consequently establish as many video multicast connections as there are terminals in the session.

Another function of the session control is error message filtering. The breakdown of a connection will result in the generation of as many error messages as there are terminals at the endpoints of this connection. The session control has to find out if an error message refers to an error condition that has already been reported to the application.

The application programming interface offers an object-oriented interface to the services of the application manager and the session control. With the protocols that are used in these services, an application can establish a multiparty session between terminals. The goal of the API is to provide an interface that allows the rapid development and introduction of applications.

Some applications might be distributed over multiple application pools. To support this the application manager and the session control interface offer mechanisms to establish communication sessions between multiple application pools and terminals.

Every application pool runs a basic directory server application that gives information on all installed applications and the connected users and terminals.

III. Architecture of a Multimedia Terminal

The multimedia terminal is a software architecture that can be implemented on a low-cost hardware platform such as a multimedia PC, a multimedia X-station, or a multimedia enhanced Minitel.

The terminal interface is completely specified by the application control protocol, the connection control protocol and the transmission protocols for media streams.

The architecture of the terminal is shown in Figure 2. The terminal is managed by the *terminal control*. The *connection control* communicates with the session control on the application pool and deals with all data communication and processing issues. Data streams are processed by dedicated *stream handlers* and synchronized by *stream coordinators.* A *resource monitor* administrates the resources of the terminal and is consulted by the terminal control in the course of application negotiation procedures, and by the connection control during the actual connection establishment. The *presentation server* is responsible for the interaction between application and user and performs functions like windowing and data in- and output. The terminal may also run *local applications*.

A. *Terminal Control*

The terminal control communicates by means of the application management protocol with the application manager in the application pool. Within the terminal it communicates with the resource monitor, the connection control and local applications. The terminal control treats all issues that concern the application as a whole, e.g., all requests to start an application. The terminal control consults the resource monitor for the reservation of the resources that an application requests during the startup phase. It rejects applications for which it cannot provide mandatory resources and supports the renegotiation of resources after the startup phase. To keep track of the local state of an application it receives indications of established connections from the connection control. Applications that experience serious problems can be aborted by the terminal control.

B. *Connection Control*

The connection control is the end-point of the connection control protocol within the terminal. It receives requests from

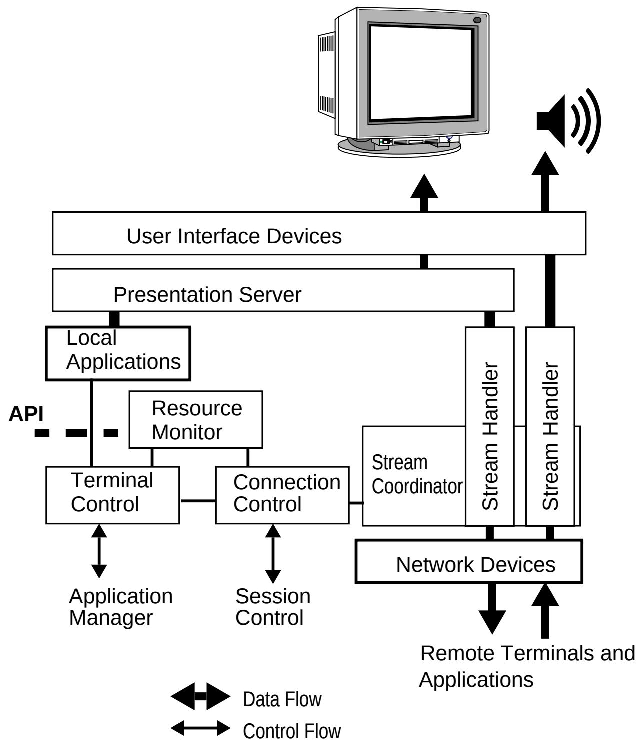

Fig. 2. Terminal Architecture

the session control in the application pool to open connections, to synchronize them with other connections, to modify them or to release them. To open a new connection the connection control requests the respective terminal resources from the resource monitor. The resource monitor grants them on the basis of the reservation that was previously effected by the terminal control. The connection control initializes a stream handler for every connection and a stream coordinator for a group of stream handlers that must be synchronized. It supervises the performance of its stream handlers and coordinators and demultiplexes connection control requests.

C. *Stream Handlers and Coordinators*

Stream handlers are the active entities that transport multimedia data from the user interface to the network and vice versa. Stream coordinators ensure the synchronization of stream handlers.

A stream handler implements the protocol stack that is required for the communication with a single medium. The control interface of a stream handler is composed of a medium-specific and a medium-independent part. The medium-independent part is common to all stream handlers and provides generic functions like the establishment of network connections, or start and stop of medium data flow. A stream handler performs the following functions:

- medium data acquisition and presentation
- compression and decompression
- transport protocol processing
- processing of a medium-specific communication protocol
- intra-stream synchronization

For a video stream the stream handler gets the video frames from a camera, compresses the frames, builds network protocol packets and writes them to the network adapter. In the sink terminal the stream handler receives the network packets from the network adapter, extracts the video data from the compressed video packets and writes them to a user interface device, e.g., a video screen. If compression/decompression hardware is available, the stream handler takes advantage of it to speed up processing.

Stream handlers collect statistical data about their performance and communicate them to the resource monitor. They report serious QoS problems to the connection control which will decide about necessary measures.

The control interface of a stream handler offers functions for presentation timing. A stream coordinator uses this part of the control interface to synchronize a group of streams. For continuous synchronization the stream coordinator establishes a common end-to-end delay among streams [5]. For event-based synchronization the stream coordinator sends presentation requests to stream handlers.

D. *Resource Monitor*

The resource monitor is the entity that controls and monitors all resources in the terminal. Resources are for instance hardware devices, I/O bandwidth and CPU time.

The resource monitor handles requests for the reservation, assignment and release of resources. The terminal control requests the reservation of resources for an application from the resource monitor. The monitor calculates whether the resources can be granted to the application and whether the additional load can be taken without limiting existing applications. The monitor can for instance give a microphone only to one application and might refuse to display two videos that each take about 50% of the available I/O bandwidth. Reserved resources are finally assigned to an application when the connection control requests them for the establishment of connections.

The resource monitor watches the average load of the terminal. Decisions are taken based on load averages, empirical QoS data, and statistical data collected by the stream handlers. Short term load fluctuations and QoS problems are handled by the stream handlers and the stream coordinator.

E. *Presentation Server*

The presentation server presents applications to the user. It controls the appearance of the application and offers the user the control interfaces to interact with the application. The presentation server provides the functionality of an X-server but supports also non-visual interfaces such as audio in- and output. The presentation server is controlled in a X-like protocol.

The user interface of an application is composed of a set of functions with local or remote significance. The presentation server demultiplexes user-generated events to local stream handlers or to the remote application which then takes the appropriate actions. A function with local significance would be a control panel that allows to adjust the volume of an audio output channel. Of remote significance are all functions that require some action from the application itself.

F. Local Applications

Local applications give our architecture compatibility with existing applications. Typically, these applications are single user programs for text- and data processing.

The terminal architecture provides an API for the implementation of local applications that is a subset of the API in the application protocol. The difference between the two is that the terminal API does not offer session control functions.

Every terminal runs a minimal local application that interfaces the user with the application management functions of the terminal control. This application allows the user to start and to stop both remote and local applications and to receive incoming calls. It offers access to the directory service and keeps a hotlist of favorite applications.

Before a local application can run it must obtain terminal resources from the terminal control. For some applications this may be the only interaction with the terminal API.

G. Implementation Issues

The concept of stream handlers and coordinators lends itself to an implementation in the user space of a multithreaded operating system. A stream handler can be implemented as a single task with the different processing functions being performed by a set of threads. This allows to optimize protocol processing and, since the whole processing path is within one task, to enforce QoS of a stream in the terminal. An implementation in user space allows to avoid the operating system and copying overhead of kernel based implementations [6]. The operating system must support the implementation of stream handlers and coordinators with real-time features, especially with control over scheduling [7]. Its memory management must allow to move data between network adapters, stream handlers and I/O devices with a minimum amount of copies.

IV. CONTROL INTERFACES

The terminal is conceived as an open system and is not restricted to a certain software or hardware environment [8]. It is specified by a terminal interface that is located above the network layer. Whichever end-system conforms to the terminal interface can connect to an application pool and request a service.

The terminal interface consists of a control part and a media part. The control part of the terminal interface is divided into two logical layers. The application management protocol specifies the terminal interface on the application level and provides services for invocation, modification and release of applications. The connection control protocol provides the services to setup single connections between terminals within the framework of an application. The control part requires the specification of a transport service for control protocol flows in addition to the specification of the protocols themselves.

A. Application Programming Interface (API)

The API provides an object-oriented interface to the application. An application is based on a user group. A group can consist of one, two or many users. To this group users can be invited, they can join the group and leave it. The application management protocol describes the function to create and modify a group session and to negotiate the session resources that satisfy the application requirements. Resources are seen as objects upon which the application can get control.

Once an application is negotiated the session must be established. The API provides the functions of the connection control protocol to initiate a stream handler and to control the negotiated resources.

B. Application Management Protocol

The application management protocol consists of procedures that are initiated by the terminal control and others that are initiated by the application manager in the application pool. The terminal control initiates among others the procedures

LAUNCH	procedure for the start of an application.
ABORT	procedure for the abortion of an application.

The procedure LAUNCH is solicited by a user request. This procedure includes the negotiation of the resources that are initially granted to the application. The procedure ABORT can be initiated by both a user or the terminal control following an error condition. Other procedures offer for instance access to the directory service or allow to join ongoing applications.

The application manager initiates the procedures

INVITE	procedure for the invitation of a terminal to an ongoing application.
RESOURCE	procedure for the negotiation of additional resources or the release of held resources.
RELEASE	procedure for the orderly release of an application.

The RESOURCE procedure allows an application to obtain additional terminal resources or release those that are not needed any more. The RELEASE procedure releases an application after all of its connections have been closed. A terminal does not have to initiate such a procedure since it is supposed that every application offers an exit instruction in its user interface. Having received such an instruction from its user, an application will take all necessary measures to release the respective terminal.

The following gives a list of important protocol parameters:

applicationName	is the name of an application. The name may be unique or local to an application pool.
applicationProfile	describes the application and its hard- and software requirements.
terminalProfile	describes the hard- and software configuration of the terminal.
resourceList	lists the resources that the application wants to obtain.
grantedResources	lists the resources that the terminal grants to an application.
sessionSummary	describes the state of an ongoing application in terms of connections and participants.

The use of these parameters is illustrated in the example at the end of this section.

C. Connection Control Protocol

The connection control protocol is used to establish a communication session between two or more terminals. Applications require communication sessions of a guaranteed QoS. The session control resolves this QoS and creates transport connections between the terminals with a guaranteed QoS. The QoS is handled depending on the network. For ATM networks the ATM QoS parameters are used. For other networks empirical QoS service parameters, e.g., equivalent or mean bandwidth, are used.

Multicast groups are built based on the available multicast services in the network. In the worst case a multicast group is built by grouping point-to-point connections. However, the implementation is transparent to the application which sees the session control interface.

The resources granted in the application management protocol are used to create multimedia streams by connecting the devices and resources of the terminal over the network.

The following are some protocol primitives used [9]:

CreateSH	creates a stream handler and connects it with an input or an output stream (or device).
ConnectSH	opens a network connection to one or multiple stream handlers.

Send and receive operations set the stream handler in a state where it moves data between the network connection and the input and output streams or devices. The data movements are triggered by the data source or by a timer that is controlled by the stream coordinator.

D. An Example

The establishment of an application is shown in Figure 3. The user on Terminal A wants to use a service and requests it from his terminal control. The terminal control connects to an application pool that offers this service and initiates the LAUNCH procedure (❶). In its initial message the terminal control will just give the applicationName of the application that corresponds to the requested service. The application manager will return a LAUNCHaccept message if it can offer this service. This message contains the applicationProfile and the initial resourceList. The terminal control examines if its own profile matches the applicationProfile and returns a LAUNCHstart message if it finally wants to start the application. This message contains the list of grantedResources and the terminalProfile. In the following the application manager will set up the application which will, once running, connect to the presentation server in Terminal A (❷). The application creates a user interface on Terminal A, which allows the user to specify that she wants to call the user of Terminal B. The application will prompt the application manager to invite this terminal. The application manager connects to Terminal B and initiates the INVITATION procedure (❸). The initial request of the INVITATION procedure contains the applicationName, the applicationProfile, the sessionProfile, and the initial resourceList. The connection control of Terminal B is thus informed about the kind of application that initiated the INVITATION procedure, its momentary state and the resources that the terminal is supposed to grant, if it agrees to participate. The connection control of Terminal B answers with an INVITATIONproceeding message and consults its user. After a positive answer of the user the connection control will finally send an INVITATIONstart message to accept the invitation. This message contains the terminalProfile and the grantedResources along with the terminalProfile. Now that Terminal A and Terminal B are participating the application is able to establish a medium connection between the two by means of connection control procedures (❹). Following a respective application request the session control in the application pool will send a CreateSH message to both of them, with configuring Terminal A for active-open and Terminal B for passive-open. After the connection controls of both terminals have sent a positive acknowledgment, the session control will ask the connection control in Terminal A to establish the network connection (❺).

In the following the application might establish further connections and add other participants. An application knows the terminalProfile of every participating terminal and will use this knowledge in connection related decisions. The momentary state of the application is reflected in the sessionProfile which is given to every terminal that enters the application.

V. Service Examples

The following two examples demonstrate how services are implemented on top of our architecture. The examples are

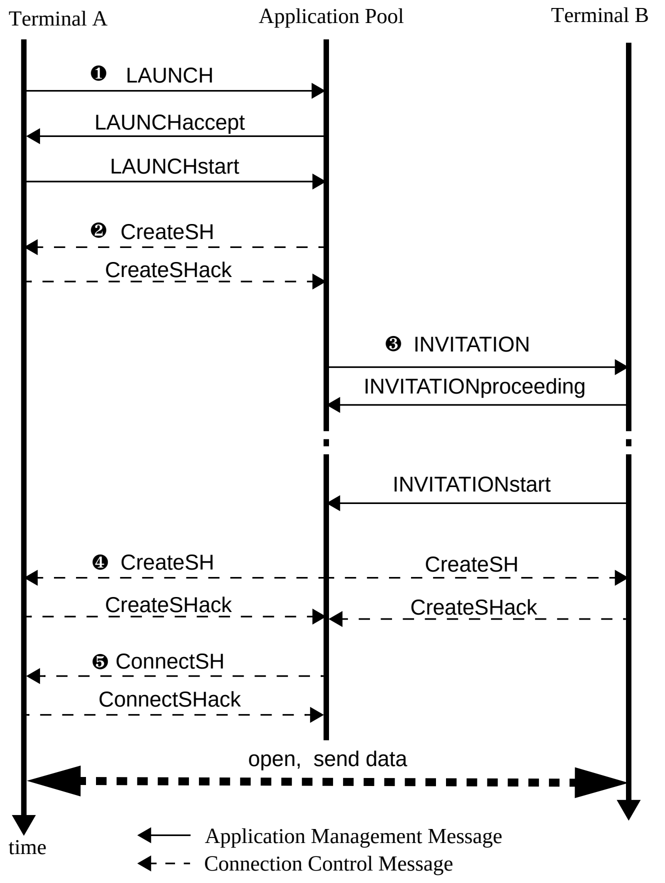

Fig. 3. Application Establishment Example.

tele-teaching and home shopping. Both services are offered by private service providers. The service provider owns the application pool whose address and services he will advertise. He may at any time install new services in its pool and remove other ones.

A. *Tele-teaching*

This example is derived from the Betel tele-teaching project in which our institute participated [10].

The service provider in this example is a school that offers services for the remote training of people on word-processors or other software products. The school itself consists basically of an application pool on which all of its tele-teaching applications are installed. Both teacher and students work on their personal multimedia terminals at home. The teacher starts a tele-teaching application remotely at a given time and waits for the students to connect to it. Initially the teacher's image and voice is distributed to all class members. The teacher will give a short introduction into the goal of the lesson and possibly demonstrate it by starting the word-processor, sharing its user interface with the students and explaining certain features directly on the program itself. If a student has a question an audio connection is established between her and the teacher. Student questions are fed back into the teacher's audio channel so that everybody can hear them. After this initial phase every student starts her own word-processor and begins to work on the assignment. The teacher is no longer visible to everybody. Instead, she connects herself to single students, asks for their progress and helps them if they have problems. In this case a bidirectional audio-visual connection is established between the teacher and the student. The student can share her word-processor with the teacher and thus demonstrate her results to the teacher or get direct help. Students may also call the teacher who will then connect to them as soon as she is free. If there are questions coming up in these private consultations that are of general interest, the teacher may choose to talk again to the whole class and answer them for everybody. Once the lesson is finished the teacher waits until all students have disconnected and releases the tele-teaching application.

On the side of the teacher and the student, the tele-teaching service requires a standard multimedia terminal with keyboard, mouse and audiovisual capabilities. Since student and teacher want to use the screen of the terminal as workspace it makes sense to have an extra screen for the video image. This will also improve the quality of the communication between teacher and student.

The tele-teaching application in the application pool of the school contains functions for the interaction with one teacher and a set of students, the taught word-processor, and a function to share the user interface of the word-processor. The application has a connection to the presentation server of every participating terminal. It generates a user interface for the teacher that allows her to control the application, to switch between a communication with the whole class and single students and to share the user interface of the word-processor. The user interface for the student allows to ask questions when the teacher is addressing the whole class, to call the teacher and to start the word-processor. Every student will run one instantiation of the word-processor in the application pool of the school. The instantiations of the program might be distributed onto multiple machines. An application-sharing component allows to connect more than one terminal presentation server to the word-processor.

Figure 4 shows the service-specific connections that may exist during a tele-teaching session. The tele-teaching application can have two control connections to the presentation server of a terminal, one for the word-processor user interface and another for the application user interface. The tele-teaching application uses three different types of media connections; an audio and video multicast connection between the professor and all students, a bidirectional audio and video connection between the professor and a single student, and unidirectional audio connections for student questions. The API in the application pool supports the creation and dynamic modification of the session connection structure. The applica-

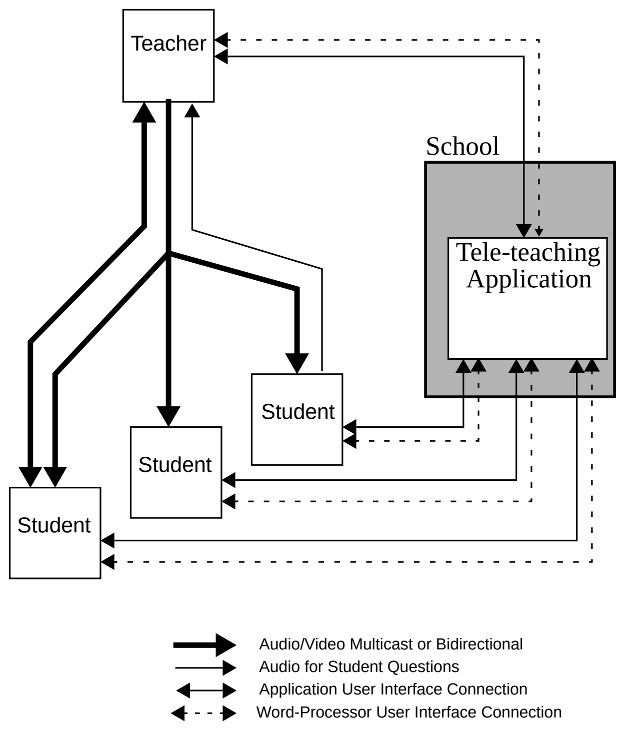
Teacher
School
Tele-teaching Application
Student
Student
Student
Audio/Video Multicast or Bidirectional
Audio for Student Questions
Application User Interface Connection
Word-Processor User Interface Connection

Fig. 4. Tele-Teaching Service Scenario.

tion will once request a multicast connection for audio and video between the teacher and the students. The session control will then automatically add every student to this connection that enters the session. When the teacher goes from student to student, the application will just call a switch function. The session control will then release the connection between the teacher and the student she is currently talking to, and establish the connection to the next student. The API may allow the specification of connection policies. In this case the tele-teaching application will specify that the multicast connection and a bidirectional connection to a single student cannot be simultaneously active. The session control will then automatically deactivate a bidirectional connection when the teacher switches to multicast, and vice versa.

B. Home Shopping

Home shopping is an interactive application that allows a user to see the latest products of a vendor or to walk through a virtual mall. The application consists of short video clips that show the products and a hypermedia structure that leads a customer through the offers or through the mall.

The distributor advertises its service on the directory service of the application pools. As this service is supposed to attract many users, the distributor will install its own application and its media terminals that act as video and image servers. To process the transactions and orders a transaction server is required.

A user who wants to go shopping will select her favorite home shopping service and launch the home shopping application. The application pool of the distributer will instantiate the application and connect the user terminal with the media server. The user will get an attractive welcome screen that shows her how to get to the desired information. These hints are virtual links to hyperdocuments. If a link is selected, a control message is sent to the application, which triggers the media server to transmit the document, e.g, a video clip. Control messages that indicate the decision to order a product are forwarded by the application to the transaction server.

VI. Discussion

The architecture has been designed to facilitate the development of distributed applications such as tele-teaching in Betel. The API should relieve the application designer and developer from caring about the details of the distributed terminals and their interconnection. Instead she can concentrate on the functions of the application and the user interface. Toolkits for CSCW could further facilitate application development.

A main benefit of our architecture is that it supports a fast introduction of new services into a network. A service is accessible through a terminal as soon as it is implemented on an application pool. Service diversity is made possible by the object-oriented API that supports a wide range of terminal features and communication services. New service features corresponding to terminal extensions are introduced as extensions of the control protocols.

The terminal interfaces should be normalized such that the architecture is open for the development of applications, application pools and terminal equipment. The requirements on terminals are minimal to provide the portability of the architecture to a wide range of equipment. The terminal and application pool are software systems that can be implemented on any hardware platform that provides sufficient resources. The integration of local applications in the terminal allows the interoperability with traditional applications.

We plan to build a multimedia service infrastructure based on this architecture in our institute. Currently, we are implementing the stream handlers with their connection control and the session control on the application pool. For the stream handlers users space protocol implementations are under way. All our developments efforts are based on our institute’s infrastructure; on SUN stations that are used as multimedia terminals and as application pools, and on a local ATM network.

References

[1] J. J. Garrahan, P. A. Russo, K. Kitami, and R. Kung, "Intelligent Network Overview", *IEEE Communications Magazine*, Mar. 1993.

[2] S. J. Mullender, I. M. Leslie, and D. McAuley, "Pegasus Project Description", University of Cambridge Computer Laboratory, Technical Report No. 281, Sep. 1992.

[3] N. Williams, G. S. Blair, "Distributed Multimedia Applications: A Review", *Computer Communications*, Vol. 17 No. 2, Feb. 1994.

[4] V. Mak and others, "Touring Machine: A Software Platform for Distributed Multimedia Applications", *Proc. 1992 IFIP International Conference on Upper Layer Protocols, Architectures and Applications*, Vancouver, Canada, May 1992.

[5] C. Blum, "Continuous Stream Synchronization", Eurecom Internal Report, 5/24/1994.

[6] A. Edwards, G. Watson, J. Lumley, D. Banks, C. Calamvoskis, and C. Dalton, "User-space protocols deliver high-performance to applications on a low-cost Gb/s LAN", to appear in Proceedings SIGCOMM '94.

[7] G. Coulson, G. S. Blair, P. Robin, and D. Sheperd, "Extending the Chorus Micro-Kernel to Support Continuous Media Applications", *Proceedings of the 4th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Lancaster, England, Nov. 1993.

[8] Y. Chang, D. Coggins, D. Pitt, D. Skellern, M. Thapar, and C. Venkatraman, "An Open-Systems Approach to Video on Demand", *IEEE Communications Magazine*, May 1994.

[9] L. Gautier, E. Rütsche, "A Communication API for the Eurecom Multimedia Networks", Eurecom Internal Report (Draft), 6/14/94.

[10] Y.-H Pusztaszeri, J.-P. Hubaux, M. Goud, E. Biersack, P. Dubois, and P. Gros, "Multimedia Teletutoring over a Trans-European ATM Network", to appear in the *Proceedings of the 2nd International Workshop on Advanced Teleservices and High-Speed Communication Architectures* (IWACA), Heidelberg, Germany, Sep. 1994.