Proceedings of the 1st International Distributed
Conference IDC'95, Madeira, November 1995

A Semi-Distributed Platform for the Support of CSCW Applications

Christian Blum, Philippe Dubois,

Refik Molva, Olivier Schaller

Institut Eurécom
2229, route des Crêtes,
06904 Sophia-Antipolis, France
Voice: (+33) 93.00.26.38
Fax: (+33) 93.00.26.27
{blum,dubois,molva,schaller}@eurecom.fr

Abstract - This paper describes the communication platform that was implemented for the European BETEUS (Broadband Exchange for Trans-European Usage) project. The BETEUS communication platform supports the fast implementation of networked multimedia applications with conference character and CSCW (Computer Supported Collaborative Work) features. The platform exhibits the notion of a site as one of its main abstractions. A site is a collection of workstations, media input and output devices and switching devices that are, in terms of control, tightly coupled. The available equipment within a site is logically mapped onto nodes, where the definition of a node is application dependent. Connection control in a site is centralized within a connection management entity. A site management entity residing on top of the connection management provides high level connection and session abstractions to applications. Applications can run within a site as well as span multiple sites, in which case application and connection control is distributed among the involved sites. So far, two applications have been implemented on top of the platform: a tele-meeting application and a tele-tutoring application. The ease with which these applications could be implemented justifies the extra-effort that went into the development of a platform.

1 Introduction

The collaborative teleconferencing platform described in this article was developed in the course of the European BETEUS (Broadband Exchange for Trans-European Usage) project [1][2][3] and was tested in field trials over the European ATM pilot network which interconnects the BETEUS partners in France (Eurécom, Sophia Antipolis), Switzerland (CERN in Geneva, EPFL in Lausanne, ETHZ in Zürich), Germany (TUB Berlin) and Sweden (KTH Stockholm). BETEUS is a 16-month follow-up to the BETEL tele-teaching project [4] in which two of the BETEUS partners, Eurécom and EPFL, were involved. The BETEUS platform profited from the experience gained with the BETEL tele-teaching application which was one of the first collaborative applications to be run over cross-national ATM links in Europe.

Today's collaborative teleconferencing systems are usually implemented as stand-alone applications with fixed interaction and communication scenarios. They implement all the components of a teleconferencing system, from low-level media transmission and processing up to the user interfaces, from scratch. One drawback of this approach is that the interaction and communication scenarios may be hard to modify once implementation is reasonably advanced, or their modification may even turn out to be impossible. Another drawback is that software may be hard to reuse for other applications if its internal interfaces were not defined with this requirement in mind.

Ease of modification and software reuse were two prominent requirements imposed on the design of the teleconferencing system for BETEUS. This is because the main focus of BETEUS is not so much on the actual application than on a concept called the *virtual community.* A virtual community can be described as a group of people that interact with each other in a natural way by means of a networked application. One prerequisite for natural interaction is high quality multipoint audiovisual communication which in turn is only possible on a broadband network. Two types of interaction within the virtual community were vaguely envisioned for BETEUS at the beginning of the design phase, one being a simple collaborative meeting, the other being tele-teaching. The final interaction scenarios were thought to evolve out of a series of prototypes that could be tested on the BETEUS network. This situation led to the decision to construct a communication platform on top of which the various scenarios could be implemented with significantly reduced effort as compared to stand-alone prototype systems for every scenario. In March 1995, eight months after the initial design steps, work was advanced to a point that the platform and its components could be deployed for the field trials that started at that time. In July 1995, two application scenarios, tele-meeting and tele-tutoring, were successfully demonstrated to a commission of the European Union. The ease with which these application scenarios could be implemented justifies the extra-effort that went into the development of the platform.

In the following, we will first talk about the objectives and constraints that underlie the design decisions for the BETEUS platform. We will then describe in detail the platform and its components as well as the scenarios that are until now implemented on top of the platform. An example scenario illustrates how BETEUS applications are developed. The paper closes with an outlook on future work.

2 Objectives and Design Issues

A key aspect of BETEUS is that the project partners are interconnected via a broadband network, the European ATM pilot network. The BETEUS CSCW (Computer Supported Collaborative Work) applications are supposed to demonstrate the high quality of human communication and interaction that can be achieved when bandwidth is not a limiting factor. The people that are brought together by a BETEUS application should communicate and interact as freely as if they were sitting together around a table in a conference room, or, in other words, they should experience themselves as a virtual community within a media space [5]. This is only possible if the quality of the audiovisual communication and of the collaboration tools is such that every implicated person has real presence at a remote location. This is clearly not the case in a conferencing environment like for instance MBone [6] over the Internet, where audio is constantly interrupted, where video frame rate is highly irregular and the person in the video window is not perceived as real. The BETEUS applications shall allow people to talk to persons on the screen as freely as they would talk to them face-to-face.

Application Scenarios

Two application scenarios were envisioned for the project. The first, the tele-meeting scenario, was supposed to be a rather informal meeting environment where people could come together to talk and possibly collaborate on some document. The other, which was called the distributed classroom scenario, should combine classrooms at different sites to a single virtual classroom. A professor could give a lecture in one classroom in which remote classrooms would participate. Every classroom would be equipped with multiple screens that show all other classrooms and possibly the slides of the professor, and people within different classrooms could communicate with each other and the professor in a way similar to a panel discussion. The two applications differ fundamentally from each other in that the first one assumes a single user terminal as standard endpoint equipment, whereas the second uses a collection of workstations and media in- and output devices to assemble a classroom. None of the applications was clearly specified at the beginning of the project. It was assumed that the application scenarios would evolve in the course of the project as tests are performed and experience is gained. It was not even clear if both of the applications would be retained since it could turn out that another one was more adequate for the virtual community concept. These considerations led to the decision to implement a real conferencing platform rather than stand-alone systems for everyone of the envisaged application scenarios. The effort to implement an application scenario should be minimal, requiring the platform to constitute the highest possible common denominator between the envisaged application scenarios.

At a later stage of the project the decision was taken to postpone the implementation of the distributed classroom scenario and to implement instead of that a third scenario, the tele-tutoring scenario, which is a replication of the BETEL application on the BETEUS platform. The practical consideration behind this decision was that the tele-tutoring scenario could be demonstrated on the testbed without any changes to the hardware configuration. The tele-tutoring scenario brings together one professor and a couple of students, all of them at a personal workstation, with the purpose to train the students on some software. The decision to introduce a new application scenario at that stage of the project demonstrates the flexibilty that was gained by having a platform.

Platform Architecture

Another design issue was the architecture of the platform. In terms of control, the platform could be completely centralized or completely distributed. The centralized solution was declined, first because there would be a single point of failure, then because of performance and scalability considerations. But it was felt that control should be centralized within the network of a project partner. This is because it was assumed that an application endpoint will not be a single multimedia workstation, but rather a logical unit that is assembled from a collection of equipment including workstations, multiple screens, cameras, speakers and microphones, as well as digital and analog switches. For the total amount of tightly coupled equipment within the network of a project partner the abstraction of a *site* is introduced. The abstraction of a *node* is introduced as the application dependent mapping of equipment onto a logical application endpoint. Connection and session control within a site is performed by a central entity that knows about the application specific node mapping. Control within applications that span multiple sites

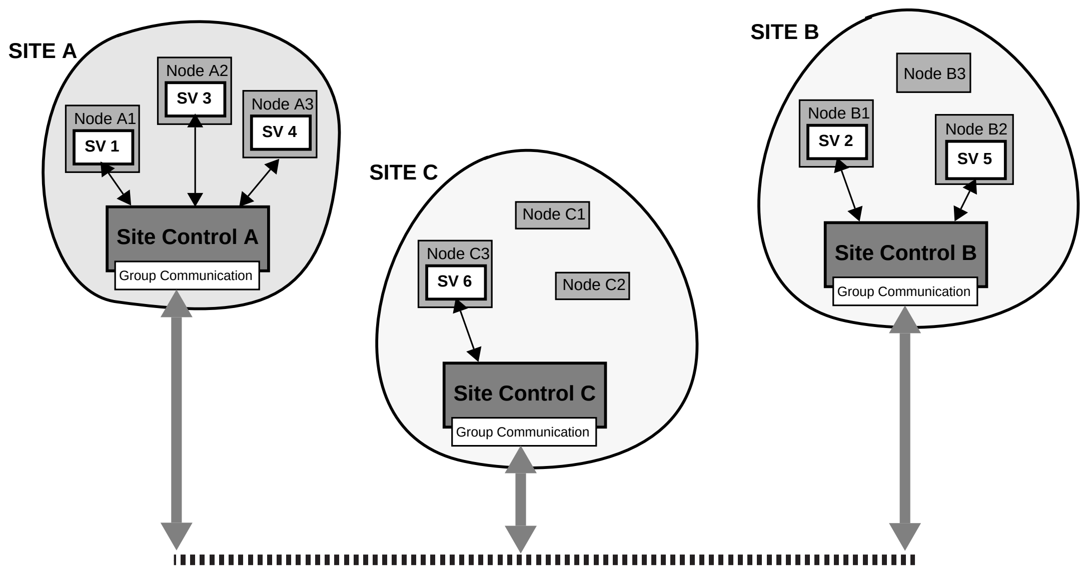


FIGURE 1. The BETEUS application model (SV=Session Vertex)

is distributed among the central entities of the respective sites. The resulting platform is thus semi-distributed: it is centralized within a site, but distributed among sites.

3 The BETEUS Platform

This section describes the BETEUS platform. It starts with a description of the application model; it then summarizes the services offered by the platform and describes the architecture of the software.

4.1. Application Model

At the heart of the BETEUS platform is an application model. The BETEUS application model introduces the abstractions of a *session*, a *session vertex* and a *session application*. A session is the abstraction for one instance of a distributed application that runs on top of the BETEUS platform. A session comprises, from a logical point of view, a set of nodes as session members. From a computational point of view, a session consists of a set of session endpoints, called session vertices, which are processes that run on the session nodes. The ensemble of session vertices within a session constitutes the session application. In the following we will use the term session application interchangeably with application or application scenario. If we want to refer to a process running at a node within the framework of a session application we will explicitly refer to it as session vertex. Note that there is no abstraction for human session participants: the node abstraction covers participants as well as participant groups or seminar rooms. The BETEUS abstraction *participant* is therefore just a dynamic name tag for a node that participates in a session.

Figure 1 shows three sites with each of them having three nodes defined in its site configuration file. A BETEUS application is indicated that spans all three sites, with three nodes being implicated at site A, two at site B, and one at site C. In fact, there is no limitation on the location of the nodes that form a session; they can be all within a single site, or all within different sites. It is therefore also completely hidden to the session vertex on a node if the session in which it participates spans remote sites or if it is local. Session vertices always interact with their local site control, but the processing of a session vertex request may trigger inter-site communication, which is the case whenever connections need to be established in-between sites. The group communication module indicated in Figure 1 provides the session broadcast, multicast and unicast messaging services required for inter-site communication.

Roles

The session vertices of an application are identical in terms of code, but behave according to dynamically taken or assigned *roles*. There is one prominent role within a session, which is the *session master*. The session vertex that is the session master has certain rights with respect to the session that other session vertices do not have. This includes for instance the right to delete nodes or to kill the session. An application may decide for itself to which extent it offers this functionality on a user interface. It may also offer this functionality on other interfaces than on the one of the session vertex that holds the master role. The master role is the only role which exists per default - all other roles are defined by the application itself. Applications may bind certain connection endpoints to roles and let the site infrastructure do the mapping of the given role to a session vertex. All roles, including the master role, can be reassigned to other session vertices. This allows applications to specify the audio and video connection structure once on session start-up; later on it will only transfer roles in-between session vertices when it wants to change the connection structure. An evident example for this would be a speaker role that is at the root of an audio and a video multicast connection. The infrastructure will automatically rebuild this multicast connection whenever the speaker role is passed from one session vertex to another. An application may define as many roles as it wishes to, and session vertices may also hold multiple roles at the same time.

Bridges and Bridge Sets

The introduction of the role abstraction provides already considerable comfort for application development. In addition to this the platform provides abstractions for connection structures. A *bridge* is a single-medium connection structure among session nodes. A bridge can either represent a point-to-point connection, a multicast connection, a broadcast connection, or an all-to-all connection. The concept of a medium bridge hides the underlying network to the application; a connection management entity realizes bridges with whatever transport the network offers. A set of bridges, typically an audio and a related video bridge, can be assembled to form a *bridge set*. An application configures the platform on start-up with a description of the bridge sets that it uses. During a session only one bridge set can be active at a time. If the application wishes to change the connection structure it will switch to another one of its bridge sets. The infrastructure will then tear down any connection that is not included in the new bridge set, and establish the ones that are missing.

BeCool

The concept of bridges and bridge sets, along with the concept of roles being at the endpoints of bridges, allows to build applications that are almost stateless. This more or less reduces the development effort for simpler applications to the design of the user interfaces. Nevertheless, it was realized that the development of more advanced applications, i.e., applications using many different dynamically assigned roles, may pose some problems for unexperienced programmers. It was therefore decided to build a development tool that compiles an application description language to a C++ session vertex skeleton that then has to be filled out by the programmer. The resulting BeCool (BETEUS Cooperative Language) compiler [7][8] turned out to be a useful tool even for the development of simple applications.

4.2. Platform Services

The services provided by the platform are

- connection control
- session management
- application sharing
- messaging service
- directory service

These services are summarized in the following.

Connection Control

As outlined above, the application specifies its connection structure in terms of roles, bridges and bridge sets. The platform establishes the endpoints of bridges, i.e., audio or video sender and receivers, and the necessary transport links that interconnect these endpoints. Endpoint parameters like audio volume or video brightness are controlled via the connection management.

Session Management

Nodes may create sessions, join them, leave them, and delete them. The session management interfaces internally to the connection control - connection structures are updated whenever nodes enter or leave the session.

Application Sharing

Collaboration within BETEUS applications is provided by the possibility to share workspaces within the framework of the X11 windowing system. The interface of an X11 application running at one node can be replicated at other nodes without that the application would need to be prepared for this. This allows to visualize X11 applications at different nodes and further to share their control among the session members. Visualization is already an important aspect since it allows to communicate information, like it is contained in electronic documents or in the interface of a simulator, in a very convenient way. If the control of an X11 application is to be shared, there is a need for floor control. The platform offers the floor control features of the shared workspace system that it uses.

Messaging Service

Session vertices use the messaging service of the platform to communicate among each other.

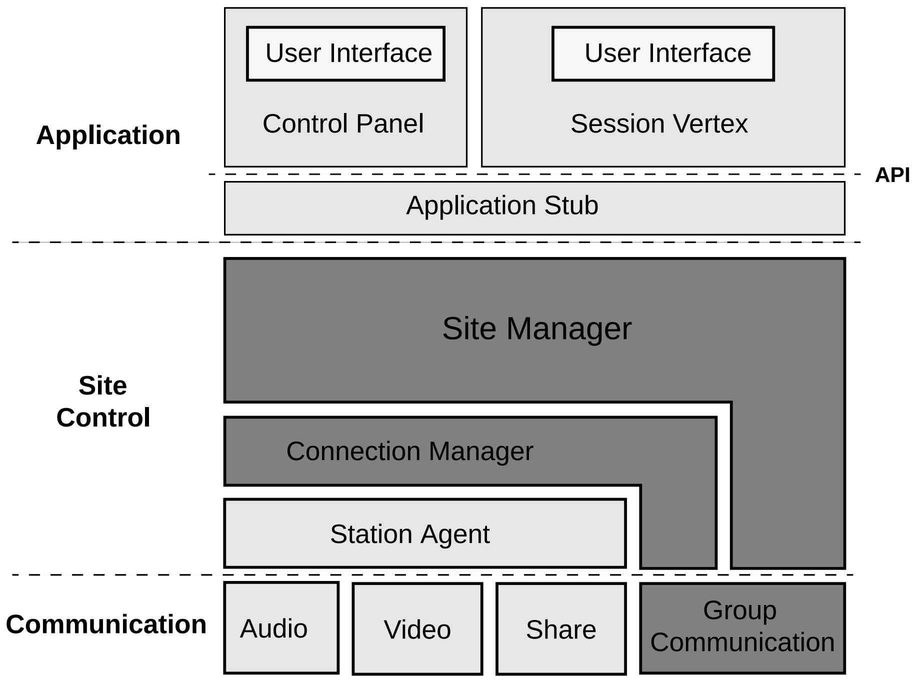


FIGURE 2. The BETEUS site architecture

Directory Service

Nodes register with the platform when they are activated. One field in the information sent to the platform is the participant name under which the node wants to register. Once registered the presence of a node is visible to all other nodes within the network via the directory service. In addition to registration information, the directory service informs about announced and ongoing sessions. This information is used by nodes to create or to join sessions.

The directory service enhances the platform from a development to a complete communication environment that could be used by a group of people scattered over different sites for their daily work.

4.3. Site Architecture

The three principle layers of the BETEUS site architecture are depicted in Figure 2. The top layer is an application layer containing a generic control panel and the BETEUS application processes - the session vertices. In the middle there is the site control layer which comprises the site manager, the connection manager and the station agents. The site manager implements the functionality offered at the high level interface towards the applications, whereas the connection manager performs physical connection establishment in collaboration with the station agents. The communication layer finally contains the audio, video and application-sharing software as well as a group communication entity that supports the exchange of control messages between site managers and and between connection managers. The shaded architecture components in Figure 2, i.e., the site manager, the connection manager and the group communication entity, have only one instantiation within a site and run on a well-known machine. Station agents are daemons that are found on every machine on the site network that may be source or sink of audio or video connections or that may run application-sharing software.

Most of the architecture components shown in Figure 2 correspond to UNIX processes. Application processes communicate with the site manager by means of Tcl-DP [9], a remote procedure call package for Tcl [10]. Since simple applications will mainly deal with user interface issues it is possible to implement them completely in Tcl. More complex applications will have some C++ code in addition, for instance the one generated by the BeCool application compiler, and will be built on top of an application stub that hides the remote procedure call interface towards the site manager. Tcl-DP is also used for the communication between site manager and connection manager. The communication between connection manager and station agent as well as between the station agent and its subordinate processes is based on a proprietary scheme.

4 Platform Components

This section looks at the components of the BETEUS platform, starting with audio and video communication and then moving on to the higher layers of the architecture.

4.1. Audio and Video Communication Components

Both audio and video are built on top of the User Datagram Protocol (UDP) and the Internet Protocol (IP). One of the biggest issues in BETEUS was how to implement multipoint communication with audio and video. It turned out that the structure of the BETEUS network [11], which is fully meshed and has FORE ATM switches at every site that are interconnected via the cross-connect based ATM pilot, is a hostile environment for IP multicast, which would be the natural choice for multipoint communication. This is why the BETEUS audio and video components implement in addition to IP multicast simple sender based stream duplication. The connection managers that control the establishment of connections over the network may employ whichever scheme is possible.

Audio

The multipoint nature of the communication in BETEUS makes it necessary to use some sort of combination of multiple incoming audio streams at the receiving side. This combination could be done either by a stream selector or a mixer. Using a stream selector requires silence detection at the sending side and some support for talkspurt transmission at sender and receiver. A stream selector chooses an active audio stream from the set of incoming streams for output whenever the talkspurt of the currently chosen stream is finished. A mixer could manage continuous audio on all incoming streams as well as talks spurts, but using talkspurt audio avoids the situation that there is a sum of the background noise of multiple remote sites in the mixer's output signal. The BETEUS audio component implements silence detection at the sending side with an adjustable threshold value and is built on top of the Realtime Transport Protocol (RTP) [12], which in turn uses UDP for transmission. The receiving side supports stream selection for the moment, but will support real audio mixing in the near future. Both sender and receiver generate activity events that can be graphically displayed on the user interface. The sender indicates begin and end of talkspurt to the local user, whereas the receiver indicates activity for each of the incoming streams on which it listens. The two audio encodings that are supported are 8kHz sampling rate with 8bit resolution and 16kHz sampling rate with 16bit resolution.

Video

Video transmission is built around the XVideo board from Parallax. The compression of the Parallax board follows the JPEG standard for the compression of still images [13]. On connection setup the video sender allows to specify a maximum data rate that is consequently enforced by means of a control loop in which maximum and measured data rate are constantly compared and the JPEG compression factor is modified according to the result of this comparison. Such a mechanism is clearly necessary in cases where there are data rate restrictions per video stream and traffic policing within the network. The video receiver adapts automatically to the actual compression factor as it also adapts to frame rate and window size. Care was taken to have a constant frame rate within the receiver window because the human eye is extremely sensitive to frame rate irregularities. The quality of the BETEUS video component is excellent even at frame rates as low as five frames/s, which generally makes people overestimate the frame rate when asked for a guess.

4.2. Application Sharing Software

The application sharing component allows a BETEUS session member to share any X11 application running at his node with all other session nodes. The BETEUS platform is not tailored to a specific application sharing system; it can integrate whatever system as long as this system can be controlled via a programming interface. So far SharedX from Hewlett-Packard and Xwedge [14] from the project partner ETH Zürich have been integrated into the platform.

Xwedge is a distributed shared window system that has agents running at all implicated client and server sites. A distributed approach was taken in order to improve performance. Another design goal of Xwedge was to keep it policy-free, i.e., not to prescribe a default admission and floor control. The system that integrates Xwedge has thus the possibility to employ its own sharing policies.

4.3. Group Communication

Group communication is a service offered to the site manager and the connection manager in order to insure the control communication between the different sites. A Group Communication Process (GCP) runs at each site. The messages coming from the site manager and the connection manager are multiplexed and sent to the correspondent GCPs running at the remote sites.

Currently, group communication is based on TCP/IP, but there is a version of the GCP in development that is based on the RMP (Reliable Multicast Protocol) library from the University of Berkeley [15]. RMP is designed to run on top of UDP and IP Multicast.

4.4. Station Agent

The station agent is a daemon that runs on every machine that may possibly be the endpoint of an audio or video transmission or that may be implicated as client or server in application sharing. The station agent launches audio, video and application sharing processes and relays operation requests from the connection manager to these processes and operation results and

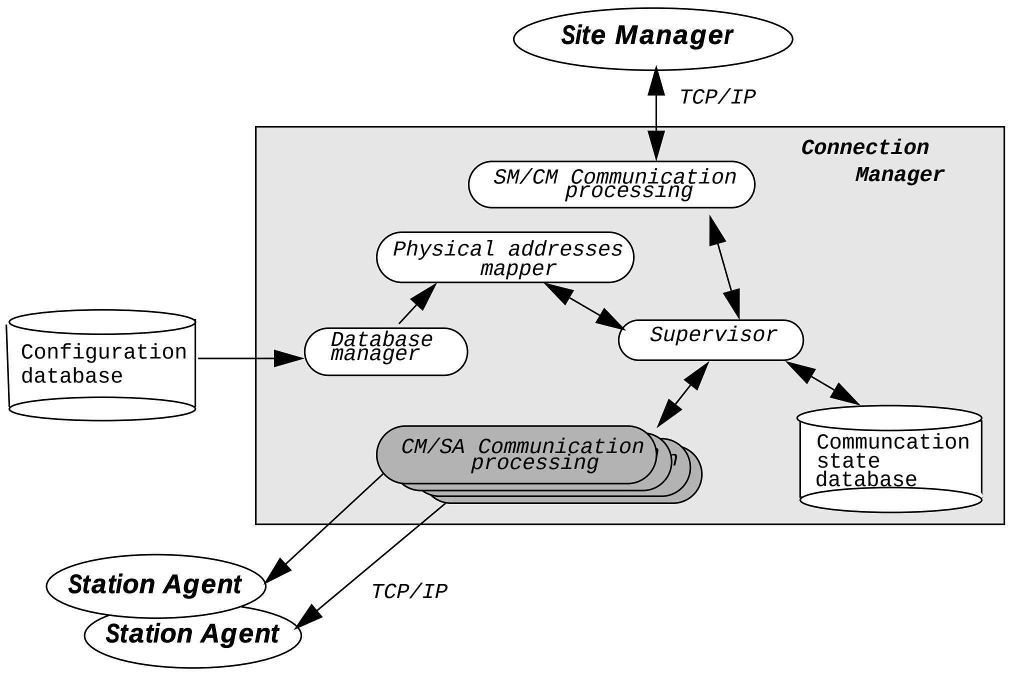


FIGURE 3. The architecture of the connection manager.

asynchronous events back to the connection manager. If one of the launched processes crashes the station agent informs the connection manager about this. It should be noted that the actual processes are transparent to the connection manager. The communication between station agent and connection manager is built on top of a stream abstraction. The mapping of streams onto processes is hidden to the connection manager and at any rate of no concern at this level. If a process becomes idle because all of its streams got disconnected it is the station agents decision to let this process keep on running, or to delete it. Some processes, like for instance the audio receiver, are kept for performance reasons because it can be foreseen that they will be reused.

4.5. Connection Manager

The architecture of the connection manager is shown in Figure 3. The connection manager takes care of the physical realization of a connection and manages all endpoint device parameters like audio volume and video frame size. The connection manager of the master site of a session knows the topology of the complete session, i.e., the member nodes and the audio, video and application sharing endpoints of these nodes. Information about the topology is stored in a hardware configuration database. This database contains a translation of the endpoint abstractions used by the site manager to physical addresses.

The site manager sends orders like:

```
CONNECT Endpoint i FROM Node j OF  site a TO Endpoint i FROM Node k OF site b
```

The connection manager maps the logical endpoint and node names to the corresponding physical addresses. The module which makes this mapping is the *physical address mapper*. The connection manager then decides if it can establish the connection locally, and does so if it is possible. If one or both endpoints of the connection are outside the site of the session master, the connection manager prompts remote connection managers to establish these endpoints.

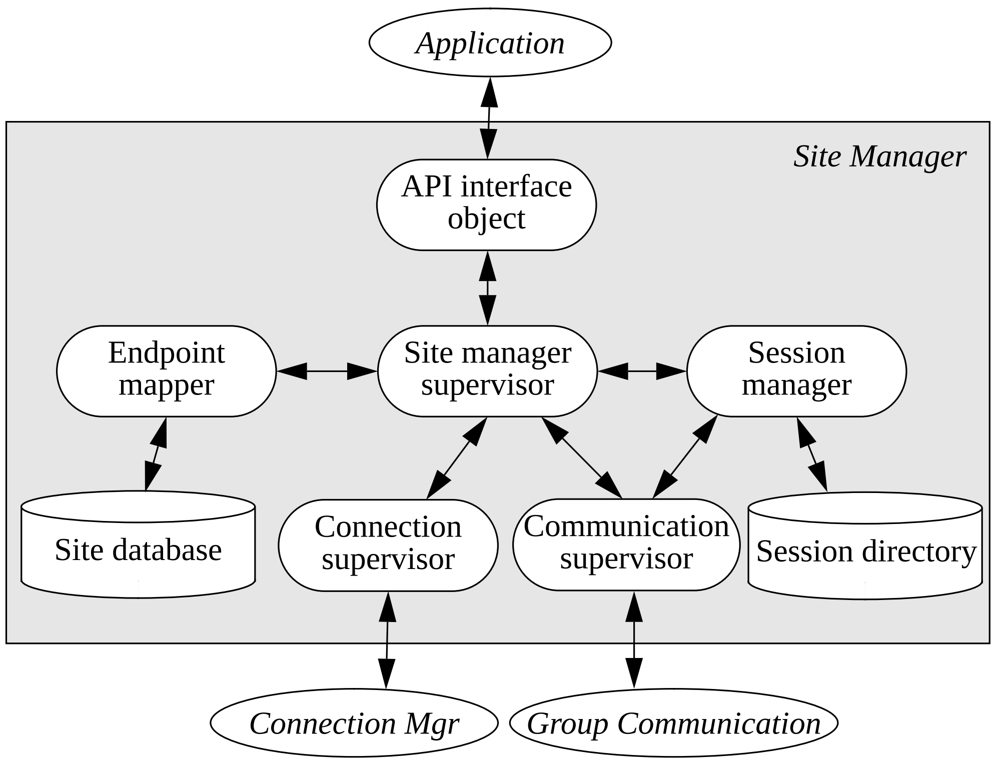


FIGURE 4. The architecture of the site manager

Currently, the connection manager is able to manage four components:

- application sharing component: shared windows and tele-pointers.
- audio: multistream senders and receivers
- video: multistream sender and single-stream receiver

An eventually involved analog video and audio distribution switch could also be controlled by the connection manager.

4.6. Site Manager

The control aspects of a distributed application within BETEUS are performed by the site manager. The site manager provides an API which enables to manage the site resources and to create sessions.

Communication between the site manager API and the application processes is done by Tcl-DP remote procedure calls. Thus calls to the site manager are synchronous and the site manager can return an error message if necessary. Tcl-DP also allows the site manager to send asynchronous notifications back to the application processes.

Figure 4 shows the architecture of the site manager. Remote procedure calls are made by the application to the API object which maps this RPCs to C++ methods of the *site manager supervisor*. The *session manager* maintains list of all announced and ongoing sessions. The endpoint mapper maps endpoints at the user node to a connector defined by a specific application scenario. The *connection supervisor* communicates with the connection manager to establish and remove individual connections. Site manager supervisor and session manager communicate with remote site managers using the services of the group communication entity.

Below is the description of the functionality provided by the site manager supervisor and the session manager.

- login: the user logs on the BETEUS platform and registers with it
- session announcement: the user announces a session with a schedule
- session startup: the user who has announced the session starts it and wait for other participants
- session running: other users join the session and the session is ongoing
- session ending: the session master terminates the session

All these operations are done through the site manager API. But the most innovative part is the configuration of the session using an application scenario. The following concepts are used to define an application scenario:

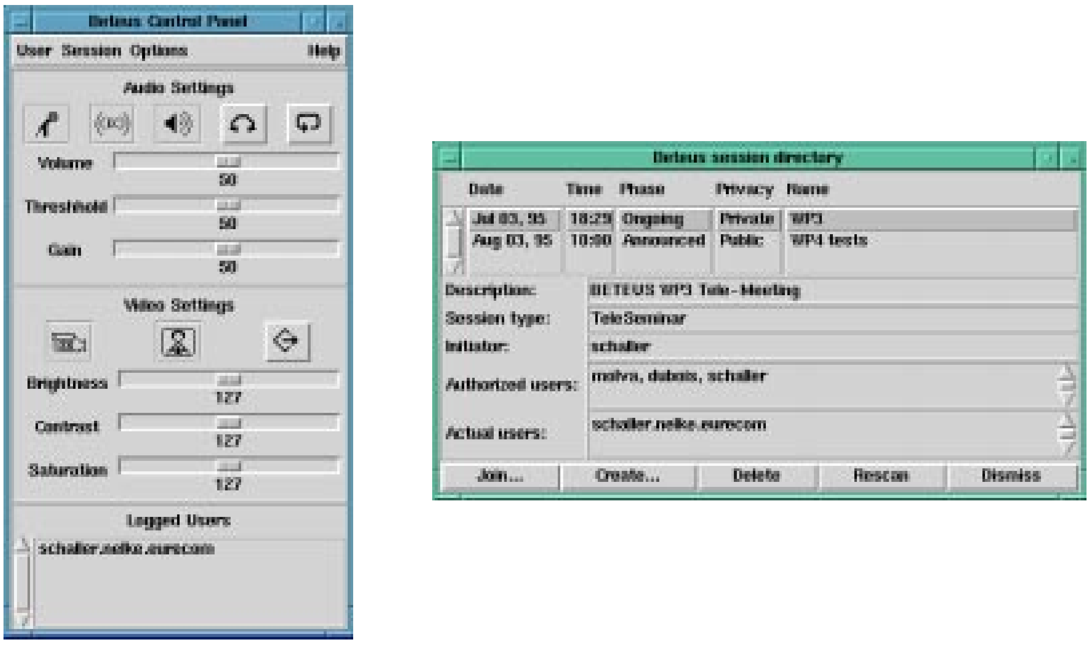
- role: every participant takes one or more roles. Roles are tightly linked with connections between endpoints.
- connector: a connector is the abstraction for a resource.
- bridge: a bridge is the abstraction for a complex multi-source-to-multi-sink connection.
- bridge set: a bridge is a collection of bridges.
- master: the master is a special session role.

At login time the user takes control of a set of multimedia resources (e.g. a video camera, a audio device, a workstation display, etc.). These resources a known by the site manager as endpoints. Each endpoint has a flow type (audio, video, x11) and a direction type (in, out), according to the configuration file of the site. The set of these endpoints is called a node. The user who logs on a node becomes the owner of all the related endpoints until he logs out.

Then the user makes a session announcement. He gives his name, a brief description and a planned schedule and the application name that is specified in the announcement. The application name corresponds to a registered BETEUS application. By default a session is announced as public, but it can be private, in which case the announcing user gives a list of authorized participants. Once the session is announced, it is visible in the session directory.

The user who has announced the session will start it at the time for which the session was scheduled. At this point an instance of a session vertex corresponding to the application identifier is launched on the user workstation. The session vertex starts with configuring the site manager with a specific scenario for the current session. Once this is done, the session is in the state ongoing and the user is now the master participant of this session. He then has to wait for other users to join the session.

As soon as another user joins the session, connections will be established between user endpoints. User endpoints will be connected and disconnected dynamically depending on the application scenario and the users interactions. The session is terminated by the master participant.



Volume 50
Threshold 50
Gain 50
Brightness 127
Contrast 127
Saturation 127

schaller.nelke.eurecom

Date	Time	Phase	Privacy	Name
Jul 03, 95	18:29	Ongoing	Private	WP3
Aug 03, 95	10:00	Announced	Public	WP4 tests

Description: BETEUS WP3 Tele-Meeting
Session type: TeleSeminar
Initiator: schaller
Authorized users: molva, dubois, schaller
Actual users: schaller.nelke.eurecom

FIGURE 5. The BETEUS Control Panel and Session Directory

The site manager of the master removes it from its session directory and disconnects all participants via the connection manager.

The definition of a role specifies a name, a maximum number of members and optionally a list of participants, in which case only these participants can take the role. On a specific node, a connector is directly mapped onto an endpoint. Roles and connectors are used by bridges to interconnect participants. A bridge connects a source connector of a role with a sink connector of another role. As participants take roles, a bridge results in a list of sources and sinks to be connected to each other. A bridge set is a collection of bridges. Only one bridge set is active at a time. The active bridge set corresponds therefore to the state of the distributed application.

In all BETEUS distributed applications, there is a special session role called the master role. Only a participant at a time has the master role of the session. This master can change the current bridge set, add and remove roles to participants, dismiss a participant and transfer its own master role to another participant. Since the connection can be lost with the master, the site manager is able to elect a new master among the remaining participants. For example, in a telemeeting session, the master role is usually assigned to the chairman of the session, but in case of a problem a new chairman will be elected. The master participant receives also a notification each time a participant joins or leaves the session.

4.7. Control Panel

The BETEUS Control Panel (BCP) is a particular application that allows the user to access the resources of a node and to log onto the BETEUS platform. Once the user has successfully started a BCP on a particular node, he can manage his local endpoints (microphone, camera, display, etc...) and consult the session directory of the site manager. He can then create a session, or join ongoing sessions.

The main window of BCP allows the user to control his endpoints as shown in Figure 5. The upper part of the interface is the audio control where volume, gain and silence detection threshold are set. The middle part of the interface controls the video settings of the sending video sender: every session participant is supposed to optimize his video signal. At the bottom of this window, BCP shows the logged users, i.e. the BETEUS users that are currently logged on the BETEUS platform. From the “User” menu, BCP provides also a multitalk functionality called BETEUS Chat. Any message typed in the Chat window is sent to all logged users.

The session directory window provides access to the session directory of the site manager. Everytime a user announces or creates a new session, all site managers are informed and BCP can retrieve this information.

A session has a type. This type is the particular application that will be executed when the session starts. In this figure there is currently an ongoing session which runs a tele-meeting scenario. Another session is announced but not yet running. There is one participant in the ongoing session. The participant is uniquely identified by a user name, a node name and a site name in dotted notation. Several sessions may run in parallel, but a user on a particular node can only participate in one session at a time.

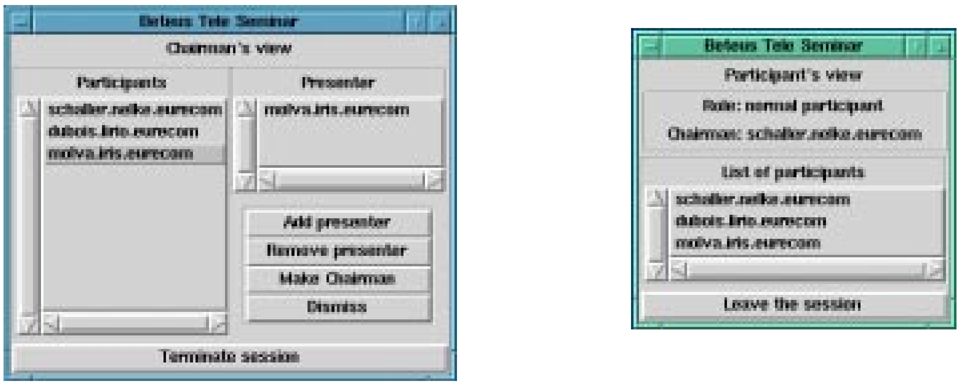


FIGURE 6. The tele-meeting scenario

5 The BETEUS Scenarios

Two BETEUS scenarios have been developed until now. Using the BETEUS Cooperative Language, other scenarios can be easily and quickly implemented. These scenarios are the tele-meeting and tele-tutoring scenarios.

Tele-Meeting

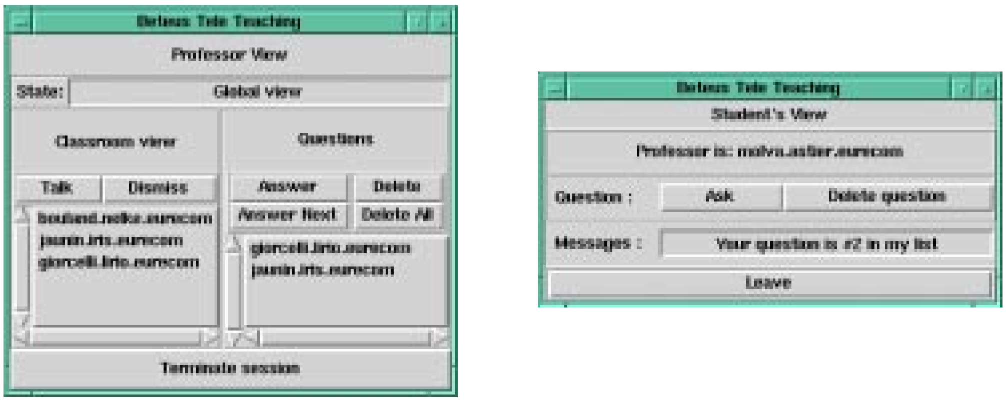
The tele-meeting scenario exhibits the notion of a chairman who chairs the session and gives and removes access to the application sharing component. The chairman has the control of the application. The chairman role is by default given to the first participant who joins the session. All incoming participants will be shown in the participants list. The scenario of this application is to have a bidirectional video and audio connection between all participants, and participants who have the presenter role can share their X-windows with everybody else in the session.

The chairman's role is unique, but transferable. The "make chairman" button allows the actual chairman to select a new chairman. The previous chairman then becomes a standard participant. The chairman can also dismiss a participant. The participant then receives the message "You have been dismissed" and all the connections concerning this participant are closed.

The interface of the tele-meeting scenario is shown in Figure 6.

Tele-Tutoring

The tele-tutoring scenario is a reimplementation of the BETEL application. This proves how easily an application can be implemented on top of the BETEUS platform. In this scenario, a professor give a course to students located at different locations. The current scenario defines two roles, the professor role and the student role. The professor is the initiator of the session, and students can join or leave the session.


Beteus Tele Teaching

Professor View

State: Global view

Classroom view

Questions

boutand.nelke.eurecom
jaunin.iris.eurecom
giorcelli.lirio.eurecom

giorcelli.lirio.eurecom
jaunin.iris.eurecom

Beteus Tele Teaching

Student's View

Professor is: molva.aster.eurecom

Messages : Your question is #2 in my list

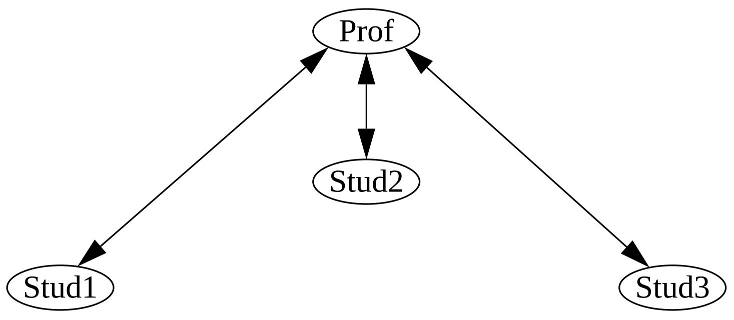
FIGURE 7. The tele-tutoring scenario

The professor's and the students' views are depicted in Figure 7. The left end list is the list of the students in the session. The right end list is the list of students having a question and waiting for the professor to answer. The professor can answer (or directly delete) questions from students and also talk with a student to ask him a question. Students can also be dismissed by the professor. The state of the application can be either *global* or *talk*. In the global state, the professor has a view on all participating students. The students see and hear the professor, but not each other. In the talk state, the professor can talk with a particular student. In this case the audio and video of both the student and the professor are distributed to all session participants so that everybody can follow the discussion between the professor and the student. In addition this student can share his X-windows applications to show his current work.

6 Role driven applications: Example

This section explains on a simple example how easy a distributed application can be built on top of the BETEUS platform.

This example defines a tele-tutoring application. This is depicted in Figure 8. Two video connectors are defined: camera (source) and monitor (sink). Three roles are defined: professor, talker and student. A participant starts the session and gets the professor role plus the master role. Other participants (stud1, stud2 and stud3) join the session. They have the student role.



```
defineBridge (bridge1, camera, professor, monitor, student)
defineBridge (bridge2, camera, student, professor, monitor)
defineBridge (bridge3, camera, talker, monitor, student)
```

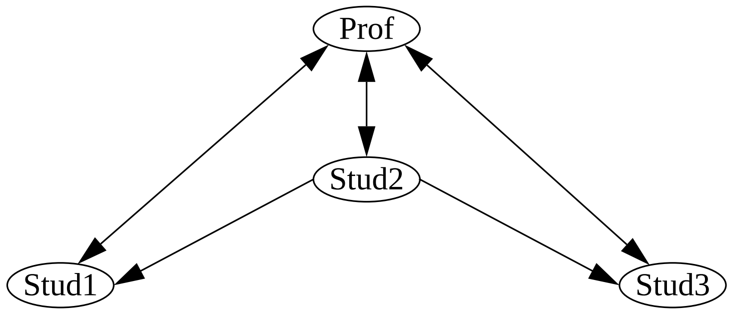
FIGURE 8. The professor is giving a talk

According to the pseudo-definition of bridge1 and bridge2 in Figure 8, each student receives the video from the professor and the professor sees every student.

Now stud2 has a question. He asks the professor by pressing a button "ask" on his student interface. Application messages are exchanged between prof and stud2 through the site manager to notify the professor of the student’s question. Once the professor decides to answer the question, he just has to add the role’talker’ to stud2. Then stud1 and stud3 directly receive the video of stud2 as shown in Figure 9. So the power of this approach relies on the fact that the application itself does not really care about the list of participants (only if necessary) to create connections between them, but only on roles they take or leave. Without this role abstraction, the application would have done something like:

```
connect (camera, stud2, monitor, sud1)
connect (camera, stud2, monitor, sud3)
```

The application would need to keep a list of participants and for each of them a list of connections to the other participants, then make separate calls for each connection and disconnection.



```
addRole (stud2, talker)
```

FIGURE 9. A student is asking a question

7 Conclusion and Future Work

We believe that the platform approach will be taken by more and more projects. A platform allows rapid prototyping of application scenarios and ease of modification, which in turn allows an incremental improvement of existing scenarios. It has to be seen that the platform itself does not need to be static. It can be continuously improved along with the scenarios that run on top of it. New scenarios will be conceived with features that could be implemented into the platform rather than into the application itself. Every feature that is incorporated into the platform rather than the application has to be considered as a preserved investment that is profitable as soon as it is reused.

It is already clear that the BETEUS platform will be reused within at least one other project. This gives us the opportunity to improve some of its components. As was already pointed out, RMP will be used for inter-site communication in the future. Another major change will probably be the use of object request broker technology for intra-site communication.

8 Bibliography

[1] BETEUS Report: "Functional specification", Deliverable D2, July1994.

[2] BETEUS Report: "Detailed specification", Deliverable D6, November 1994.

[3] BETEUS Report: "Working Prototype of the Application Platform Specification", Deliverable D8, June 1995.

[4] Y.-H. Pusztaszeri, E. Biersack, Ph. Dubois, J.-P. Gaspoz M. Goud, P. Gros, JP Hubaux :"Multimedia Teletutoring over a Trans-European ATM Network", *2nd IWACA Conference*, Heidelberg, September 1994.

[5] S. A. Bly, S. R. Harrison and S. Irwin,"Media Spaces: Bringing People Together in a Video, Audio and Computing Environment", *Communications of the ACM*, January 1993.

[6] M. R. Macedonia and D. P. Brutzman:"MBone Provides Audio and Video across the Internet", *IEEE Computer*, April 1994.

[7] J. Lindblad and O. Schaller:"BeCool and the BETEUS Application Programming Interface", Eurécom Technical Report, June 1995.

[8] J. Lindblad:"The BeCool Thesis Project Report", Master's thesis at the KTH Sweden, 1995.

[9] L. A. Rowe, B. Smith, and S. Yenftp:" Tcl Distributed Programming (Tcl-DP)", University of Berkely Computer Science Division, *ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/Tcl-DP/tcl-dp-v1.0ak*, March 1993.

[10] J. K. Ousterhout, "TCL and TK Toolkit", Addison-Wesley Publishing,1994.

[11] T. Walter, M. Brunner and D. Loisel:"The BETEUS Communication Platform", to appear in the *Proceedings of the first International Distributed Conference IDC '95*, Madeira, November 1995.

[12] IETF Internet Draft:"RTP: A Transport Protocol for Real-TIme Applications", Audio-Video Transport WG, *ftp://ds.internic.net/internet-drafts/draft-ietf-avt-rtp-07.txt*, March 1995.

[13] G. K. Wallace:"The JPEG Still Picture Compression Standard", *Communications of the ACM*, April 1991.

[14] Th. Gutekunst, D. Bauer, G. Caronni, Hasan and B. Plattner:"A Distributed and Policy-Free General-Purpose Shared Window System", *IEEE/ACM Transactions on Networking*, Februrary 1995.

[15] T. Montgomery:"Design, Implementation and Verification of the Reliable Multicast Protocol", Master's thesis at West Virginia University, *http://research.ivv.nasa.gov/projects/RMP/Docs/RMPdocs.html*, December 1994.

9 References

HTTP server available on: http://www.tik.ee.ethz.ch/~beteus/