A Cooperative Multimedia Environment with QoS Control: Architectural and Implementation Issues

Marco Alfano and Nikolaos Radouniklis
{alfano, nikolaus}@icsi.berkeley.edu

TR-96-040
September 1996

International Computer Science Institute
Berkeley, California

Abstract

A cooperative multimedia environment allows users to work remotely on common projects by sharing applications (e.g., CAD tools, text editors, white boards) and simultaneously communicate audiovisually. Several dedicated applications (e.g., MBone tools) exist for transmitting video, audio and data between users. Due to the fact that they have been developed for the Internet which does not provide any Quality of Service (QoS) guarantee, these applications do not or only partially support specification of QoS requirements by the user. In addition, they all come with different user interfaces.
We have developed a Cooperative Multimedia Environment (CME) made up of Cooperative Multimedia Applications (COMMA), one for each user. A COMMA presents a user with a single interface that allows him to invite other users to a cooperative session, select the media services to be used in the session, and specify his Quality of Service (QoS) requirements for the media services throughout the session.

In this work, we describe the architectural details of the CME and its componentents with particular emphasis to the QoS mapping and control mechanisms. We also present the design and implementation details of an experimental prototype that provides video, audio and white board services.

Contents

1 Introduction

The increasing availability of broadband networks allows the deployment of new services for an ever growing number of possible users. Among these new services, a great deal of interest has been addressed towards real-time and interactive applications, e.g., videoconferences and shared document editors, particularly because of the worldwide and decentralized structure of today's research and development organizations.

A cooperative multimedia environment allows users to work remotely on common projects by sharing specific applications. Several dedicated application suites (e.g. MBone tools [29]) have been developed to address the need for cooperative work. They offer a variety of media services including desktop videoconferencing and application sharing. Since these applications are highly demanding in terms of resources, resource availability is crucial for an efficient cooperative work. Especially in a scenario where host and network resources are often limited or do not provide any Quality of Service (QoS) guarantees, it is important to make an efficient use of existing resources in order to accommodate the user's requirements. Unfortunately, most of the existing multimedia applications do not consider resource restrictions or concurrency among employed services. Furthermore, there is no means for the user to prioritize directly specific tools in case he uses more than one tool. Inevitably, the need for an architecture that manages multimedia applications within a cooperative session emerges.

In this work, we introduce an architecture for a Cooperative Multimedia Environment (CME) with Quality of Service control. The CME thereby represents an integrated approach to define and control Quality of Service at the user, application and resource layers. The flexible and modular character of the CME architecture integrates different Quality of Service scenarios, depending on the availability of QoS guarantees for various resources (e.g. host, network). If the host and network resources offer QoS guarantees, then the architecture's components make use of them. If the resources do not offer QoS guarantees, then a set of control mechanisms is employed in order to meet the user's requirements in a best effort approach and to use resources simultaneously in an efficient way.

The work is organized as follows:

Chapter 2 discusses the motivation for a CME and presents the architectural details of the CME and its functional components.

Chapter 3 deals with Quality of Service mapping issues as an integral part of the CME architecture.

Chapter 4 mainly concentrates on Quality of Service control issues. The CME Quality of Service Controller is presented and the design issues of the control mechanisms operating at different layers are discussed.

In *Chapter 5*, the CME experimental prototype is presented. The prototype focuses on various implementation aspects of the CME architecture. The different processes running on a session participant's site are described.

Finally, *Chapter 6* summarizes the key issues of this work. Based on the evaluation and comparison of the CME architecture and its prototypical implementation, future work issues are briefly discussed.

2 An Architecture for a Cooperative Multimedia Environment

2.1 Introduction

In general terms multimedia applications provide an appropriate means to exchange video, audio and data information between users. A cooperative multimedia environment allows users to work remotely on common projects by sharing specific session related applications. Several suites of applications (e.g., MBone [29] and Berkom [15] tools) exist to fulfill this task offering various services, such as desktop videoconferencing and application sharing.

Since these applications are highly demanding in terms of resources, resource availability is crucial for efficient cooperative work. Unfortunately, the modular and independent character of existing multimedia applications does not offer a means of directly prioritizing specific applications to the user in case he uses concurrently more than one application. Especially in scenarios where host and network resources are limited or do not provide any QoS guarantees, an overall mechanism for controlling the different independent media services might prove very useful.

2.2 Motivation for a CME

In order to evaluate the practical behavior of multimedia applications, we examined the MBone tools [29]. We observed that each MBone tool offers a separate user interface and that the user must have a deep understanding of each media service and its parameters in order to use the related multimedia applications properly. The MBone tools do not provide any possibility to remotely specify quality parameters. The sending parameters are exclusively controlled by the sending side. It is a strenuous and cumbersome process to adjust the media specific parameters in order to achieve the desired quality on the receiving side. The MBone tools compete for host and network resources and there is no possibility for the user to prioritize one tool over another.

In order to examine the performing behavior of the MBone tools, we executed a set of experiments, which is reported in the Appendix. For our tests, we chose the MBone tool vic [24] to send a unidirectional video stream. Our preliminary measurements showed that applications often suffer quality degradation during a multimedia session caused by network saturation or host congestion. In particular, network saturation may lead to abrupt quality decrease.

In a scenario where users want to collaborate remotely by means of multimedia services, the users should not be confused by media service specific issues or details. For example, users are generally not interested in selecting the suitable encoding scheme in a video application. On the contrary, the session participants should be equipped with an easy mechanism to specify the quality level of the used media services. Furthermore, the specification of the desired user quality should be uniform and consistent.

As a consequence of what is discussed above, we decided to develop a Cooperative Multimedia Environment (CME) to address the experienced problems. Our CME can be understood as an approach to support QoS requirements at the user level and translate them into media service and resource specific terms. It offers a uniform platform to integrate the various media services supplying the user with a single view of the multimedia environment. Resource issues are handled independently from the underlying network or transport layer peculiarities.

2.3 General Structure of the CME

Since the experienced problems arose primarily due to the lack of integration, the CME architecture comprises both horizontal and vertical integration. Horizontal integration operates exclusively within a certain layer. Vertical integration spans to the different layers. Horizontal integration embraces all sites of a cooperative environment, whereas vertical integration only operates within one site. Figure 2.1 illustrates the numerous integration fields addressed by the CME architecture.

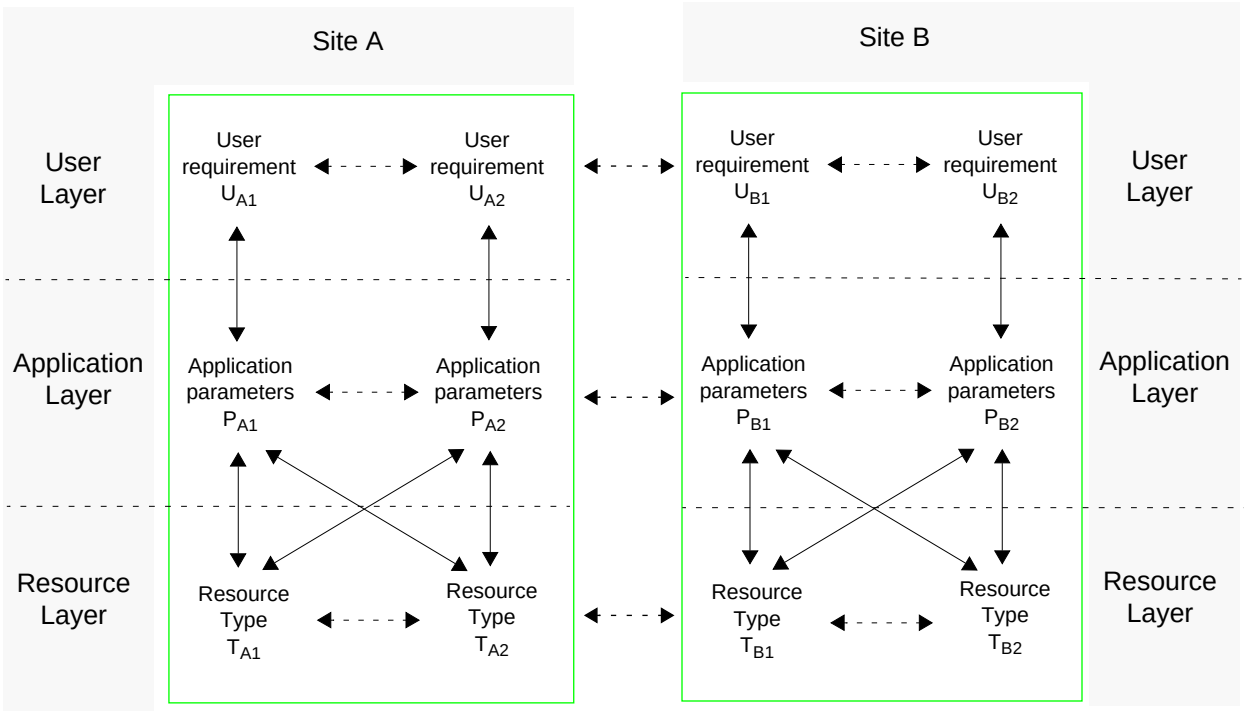


Figure 2.1 Horizontal and Vertical Integration

2.3.1 Vertical Integration

Vertical integration interrelates user, application and resource layers. The connection between the different layers is achieved by introducing mapping mechanisms. User QoS requirements are translated in media service parameters and in QoS requirements for the underlying resources.

The QoS Mapper/Controller, a basic component of our CME architecture, mainly includes mapping functionality. The CME prototype, as we will see in the following chapters, provides a mapping table for each media service. The tables include only a fraction of the variety of QoS attributes that have been defined for the different layers [22]. However, due to the modular and

generic character of the CME architecture, more accurate and sophisticated mapping mechanisms can be added in order to extend the existing mapping schemes.

2.3.2 Horizontal Integration

Horizontal integration is a result of the distributed structure of the CME architecture. It operates exclusively within a certain layer.

- User layer

  User layer integration is achieved by offering the user the possibility to specify quality requirements and priorities for the employed media services. As we will show in the following chapters, our prototypical implementation, for example, includes a control panel in order to specify the user quality wishes for the various media services in a uniform way.

- Application layer

  Integration at the media service layer is achieved by embedding the media services into the CME architecture. This means specifically that applications are not handled independently anymore. Control of media services represents an integral part of the CME architecture. The knowledge about the status of the media services we use in our cooperative environment allows us to control the media services more efficiently.

- Resource layer

  Integration at the resource layer is achieved by providing mechanisms for the orchestration between the operating system and network resources and their management structure. The CME architecture accomplishes horizontal integration at the resource layer by taking the different resource quality parameters into account. Monitor and control mechanisms keep track of resource status and availability and prevent resource saturation. The knowledge about the resource availability on the one hand and the user requirements on the other allows the cooperative environment to assign resources to the media services more accurately and efficiently.

2.3.3 Related Work on Integrated Multimedia Environments

The need for integration within a specific layer (user, media service, resource layer) and between layers has been addressed by several research groups [14], [31], [33], [35], [38], [41]. Integration efforts of other research groups differ from the integration approach of the CME architecture in that they mainly cover only a specific integration field. This section provides a few selected examples for such integration efforts.

The MBONE tool developers have introduced several media service synchronization mechanisms to address the problem of horizontal integration [33]. Cross-media synchronization is carried out

over a Conference Bus. The Conference Bus abstraction provides a mechanism which coordinates the separate media service processes. Each real-time application induces a buffering delay, called the playback point, to adapt to packet delay variations. This playback point can be adjusted to synchronize across media. The Conference Bus is also used for voice-switched windows. A window in voice-switched mode uses cues from the *vat* audio tool [23] to focus the current speaker.

Similar concepts are pursued by several research groups [38], [41]. In [41], a local control architecture and communication protocols are described that tie together media agents, controllers and auxiliary applications such as media recorders and management proxies into a single conference application. The conference controllers and media agents (in our terminology referred to as media services) communicate by sharing a message replicator. This approach is similar to the MBone Conference Bus [31] and is mainly employed to establish horizontal integration at the application layer.

User interface integration can be found in various multimedia conferencing products. With the AT&T Multimedia Communication Exchange Server (MMCX) [33], team members can get together in a virtual meeting room. Along with providing a visual representation of the virtual meeting, MMCX combines multimedia calling features with collaboration tools to allow users to add or drop services and media as they choose.

The QoS Broker [35] addresses the relationship between the various resource types (mainly operating system and network resources) and provides an architecture for horizontal resource integration in the resource layer. Processing capacity is managed in concert with networking to deliver guaranteed behavior to applications. Furthermore, the QoS Broker integrates mapping aspects by offering an appropriate scheme to convert application QoS parameters into network QoS requirements and vice versa.

A QoS architecture interrelating levels for media specific and transport level QoS handling is introduced in [4]. A negotiation and resource reservation protocol (NPR) for configurable multimedia applications [14] allows QoS negotiation and resource reservation. As an application level protocol, it offers transparency from the underlying transport layer structure.

2.4 CME Architectural Components

Figure 2.2 depicts the main components of the CME architecture. As illustrated, one of the essential properties of the CME is its distributed structure. Each site of the CME consists of a Cooperative Multimedia Application (COMMA). The Session Manager controls the different Media Services. It is also responsible for conference control, floor control, configuration control and membership control. Let us focus now on the various functional components that are comprised by a COMMA.

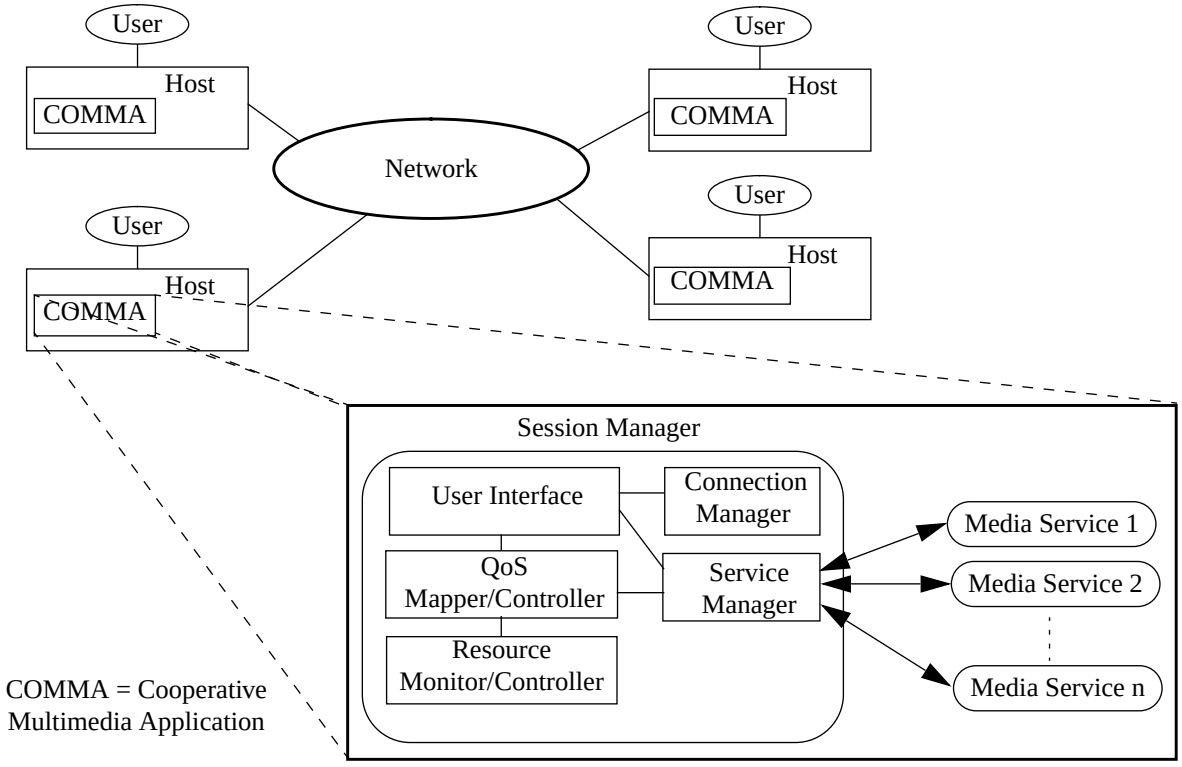


Figure 2.2 Cooperative Multimedia Environment

2.4.1 Connection Manager

The Connection Manager provides the necessary communication primitives for establishment and disconnection of cooperative sessions. During session establishment, other users are invited to join the session. Since any connection manager can initiate a collaborative session, the session does not rely on any centralized session moderator, but is based on a distributed peer-to-peer model.

- Invitation procedure:

  First the Connection Manager of the inviting site propagates an invitation message to the Connection Managers of the users to be invited. The invitation message contains the proposed media services (to be used in the cooperative session) that have been specified by the session initiator through the User Interface. The invitation message is transmitted to the different sites where an invited user can either accept or refuse the invitation. Additionally, he can specify that he will join the conference with a subset of the proposed Media Services.

The above described invitation protocol is a brief summary of the Connection Manager we have implemented in our prototype. The modular and flexible character of the CME architecture allows us to replace our specific Connection Manager with more sophisticated approaches [18], [42].

2.4.2 QoS Mapper/Controller

The QoS Mapper/Controller translates the user requirements into application specific parameters for the media services and into QoS requirements for the underlying resources (i.e. host and network resources). Media service customized mapping tables build the core of the QoS Mapper. For each new integrated media service, a new mapping table is added to the QoS Mapper.

The controlling part of the QoS Mapper/Controller executes various control mechanisms to compare and control the user specified quality with the currently provided quality. The controlling mechanisms operate on different layers (user, application, resource layer). Media service prioritization or automatic actions to prevent resource saturation are examples of controlling mechanisms.

A control mechanism is specified by a rule. A rule in turn is split into a condition and an action part. A rule base contains a set of rules that can be applied through a session. In case a ruling condition becomes true, the rule “fires” and invokes the rule’s action part.

The control mechanism uses the mapping tables of the QoS Mapper/Controller to determine new values for the media services. Eventually, the Service Manager is used to transmit the new application parameters to the media services. Alternatively, it is sometimes necessary to readjust process priorities. In this case the new resource allocation values are sent to the Resource Monitor/Controller.

2.4.3 Resource Monitor/Controller

The Resource Monitor/Controller monitors and controls the available host and network resources. Various QoS scenario alternatives can be outlined:

- Host and network resources provide QoS
- Host and network resources don't provide QoS
- Only host provides QoS
- Only network provides QoS

Figure 2.3 illustrates some scenario alternatives with practical examples of host and network resources.

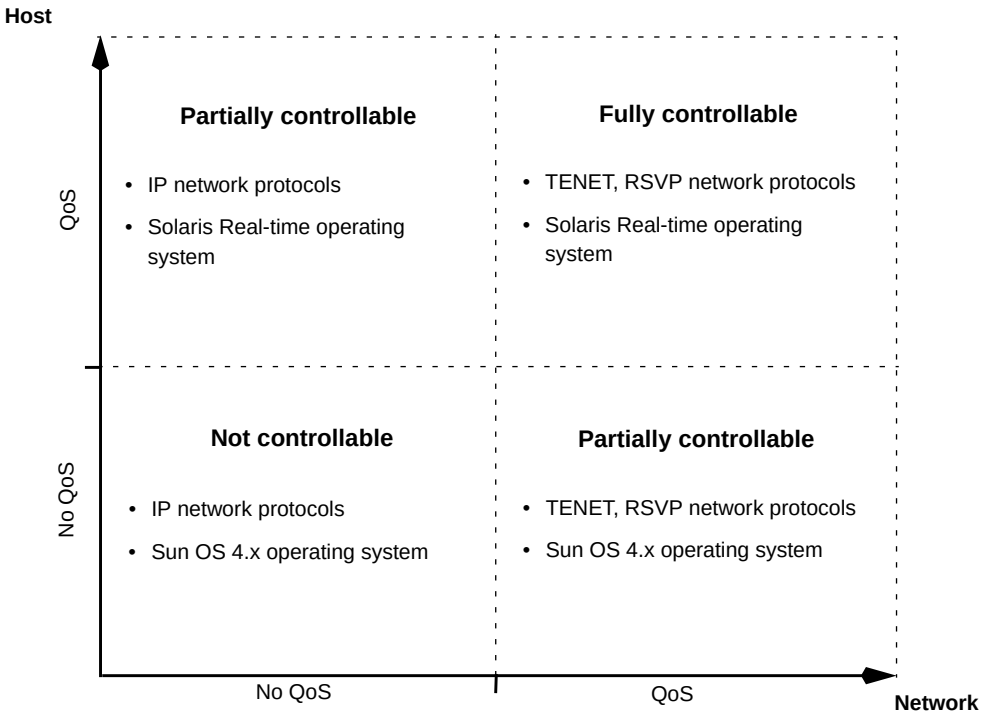


Figure 2.3 QoS Scenario alternatives.

If the host or network resources provide QoS guarantees, the Resource Monitor/Controller is used to perform three main actions:

- To reserve and allocate resources (end-to-end) during multimedia call establishment so that traffic can flow according to the QoS specification. This means distribution and negotiation of the QoS specification for system components involved in the data transfer from the source(s) to the sink(s).

- To provide resources according to the QoS specification. This means adhering to resource allocation during multimedia delivery using proper service disciplines.
- To adapt to resource changes during ongoing multimedia data processing.

If no QoS is provided by the network and/or by the host, the Resource Monitor/Controller monitors the resource status (e.g. cpu load) and provides the QoS Mapper/Controller with the monitored resource information.

2.4.4 Service Manager and Media Services

The Service Manager provides mechanisms to start and stop user-requested media services. It also retrieves and controls media specific parameters for dynamic adaptation and reconfiguration purposes.

From the architectural point of view, the Service Manager provides an interface to the various media services. The media services are thereby considered as bare tools that perform only network and media processing. The modular character of the CME architecture allows an easy integration of new media services. Since each media service offers a variety of different media specific parameters to be adjusted, the Service Manager provides a customized interface for each media service.

In case a new media service has to be integrated into the CME architecture, the Service Manager is extended by a new customized interface. A new media specific mapping table then has to be added to the QoS Mapper/Controller.

Our experience with several media services of the same media type showed that although the different tools represented the same service type, there were differences in the interfaces the services offered. Thus, we decided to build a customized service interface for each integrated media service rather than for each media service type.

As we will discuss in the following chapters, the customized Service Manager interfaces of the CME prototype integrate the MBONE tools [29] into our architecture without performing any code modifications of the bare media services.

The Service Manager interface is used either by the QoS Mapper/Controller or directly by the user. The User Interface employs the Service Manager’s functionality in order to start, stop and adjust media services. The QoS Mapper/Controller retrieves and readjusts media specific parameters by employing the Service Manager’s functionality.

2.4.5 User Interface

The User Interface offers to the user a single interface for the various media services. It is mainly divided into two parts. The first part provides an interface for the Connection Management. The user inputs for starting a session are transmitted and directly processed by the Connection Manager who generates a session related invitation message. The session initiator uses the graphical user interface to specify the session participants, the media services he wants to use and the initial media service quality in user terms. As discussed above, the invited users receive the invitation message and can accept or refuse the session invitation.

The second part of the User Interface offers a means to define the media service qualities during a session. Session participants can specify and change the media services for the session and their QoS requirements. A quality window for each invoked media service displays the current media service quality in user terms.

3 Quality of Service Mapping

3.1 Introduction and Related Work

Quality of Service Mapping deals with the translation of quality parameters among user, media service and resource layers. User requirements are mapped into application specific parameters for the media services and into QoS requirements for the underlying resources (i.e., host and network resources).

QoS Mapping is a non trivial and still open research issue, largely because the user perception of QoS is not completely understood. In addition, since there are numerous ways to describe a QoS representation for each layer, a clear understanding has yet to be developed of which QoS parameters have to be employed. Only a few research groups have concentrated on QoS mapping issues for executing multimedia applications in a distributed environment [4], [17], [20], [35].

In [35], the work focuses more on how to reserve local and remote resources by deploying a broker architecture rather than offering adequate mechanisms on how to map application QoS requirements onto resource requirements.

In [4], a mapping mechanism is introduced to translate media specific parameters into a uniform communication load representation. The advantage of this representation is to abstract from the QoS interface characteristics [16], [13] of a specific transport system. An example is further provided that illustrates how to map the parameters of the uniform communication load representation into parameters defined for the Tenet Protocol Suite [3].

As in [4], [17] exploits methods and mapping functions between QoS parameters of the application and transport layers. Continuous media stream parameters (i.e., period, quality, reliability, delay, start offset) are mapped to transport QoS parameters (i.e., throughput, reliability, delay, jitter, maximum transfer unit). Mapping of application parameters to operating system resources is not covered. The authors only address the need to map application attributes onto operating system attributes as size and number of buffers, scheduling classes (e.g. real time, timesharing etc.), priority, and number of CPU cycles.

3.2 The CME Quality of Service Mapper

The QoS Mapper/Controller plays a central role in our architecture and provides the appropriate mapping mechanisms. A small set of meaningful QoS parameters at the user level is mapped onto media service specific parameters and onto QoS resource requirements for the host and network resources.

3.2.1 QoS Representation at the Different Levels

- User Level (Direct Requirements)

Since our experience with the various MBONE tools [29] led us to the conclusion that we as end users had to deal with too many hard-to-understand application parameters, we tried to explore a simple mechanism that allows the user to specify his quality requirements directly in user terms.

A proper way to express user requirements entails a detailed analysis on how a user expects a media service to behave more or less properly and how satisfaction of the user for the media service quality can be expressed in quantitative terms. A simple approach considers the use of a five-level scale to define the quality of a media service and we give the user the possibility to specify one of these levels as a way to express his requirements. This scheme is related to many studies dealing with quality estimation of digitally coded video sequences [5], [7] and audio sequences [12], [48].

Compared to other approaches that attempt to define QoS in user terms [17], [24], where the user has to specify various media specific parameters, the quality specification in our architecture is one-dimensional. In order to provide the described simplicity, we have to define sophisticated mapping functions and partially sacrifice the end user's freedom adjusting all the media service specific parameters.

The solution we propose offers more than a simple specification mechanism of QoS. Since we transform user QoS requirements into a one-dimensional scale, qualities of different media services become comparable and provide eventually a useful priorization mechanism. The user is equipped with a powerful means to express absolute and relative requirements of a media service.

Table 1 illustrates the quality rating, the impairment and the corresponding quality we employ to specify QoS in user terms:

RATING	IMPAIRMENT	QUALITY
5	Imperceptible	Excellent
4	Perceptible, not annoying	Good
3	Slightly annoying	Fair
2	Annoying	Poor
1	Very annoying	Bad

Table 1: Quality Rating on a 1 to 5 Scale

- User Level (Indirect Requirements)

During a cooperative session a participant may use the different media services more or less extensively. In a typical cooperative scenario, the participants may for example first discuss a specific problem using videoconferencing tools and explore afterwards the problem domain in more detail using specific shared applications.

Through a session, users perform actions such as iconifying a window or putting a window in the background. These actions indirectly make suggestions about the user's interest on a particular media service and express subsequently indirect user requirements.

- Application Level

QoS in the application level is a term that has been interpreted in various ways. Some sources try to introduce a set of generic QoS application parameters. In [45] QoS parameters (i.e., period, quality, reliability, delay, start offset) for continuous media streams are defined.
The generic application QoS parameters are mapped to transport level QoS parameters and as a second step to the QoS parameters provided by a specific transport system like the TENET protocol suite [3].

Our view of application QoS is slightly different. Different media services imply different application parameters. Due to the diversity of existing media services, we prefer to define QoS parameters for each media service type. A video media service for instance can be characterized by temporal, spatial, frequency, amplitude and color space attributes types. Corresponding attribute instances are the transmission rate, the frame jitter, the resolution and window size, the employed encoding scheme, the color depth or the number of entries in the color space.

The various media services usually provide an interface to manipulate only a subset of the numerous media specific parameters. In order to keep our architecture modular, flexible and extensible, the QoS Mapper comprises a customized mapping table for each integrated media

service. The customized mapping table describes accurately the characteristics of the related media service. The description includes resource demands for selected media service settings. It further includes the QoS representation at the user level.

- Resource Level

Various approaches have been undertaken to specify and provide QoS at the resource level [22], [20]. At the transport and network layer the definition of the TENET protocol suite [3] provides the primitives for QoS issues as connection set-up, resource reservation, admission control and policing. The RCAP, RMTP/RTIP real-time channel administration, real-time internetwork and real-time message transport protocols are designed to provide the desired QoS primitives.

The realization of a certain QoS at the operating system level requires mainly CPU scheduling, memory management for buffering and efficient storage on mass media. Some work has been done exploring real-time CPU scheduling [36],[44]. A few operating systems like Sun-Solaris [47] provide real-time operating classes.

The CME architecture comprises all the possible QoS scenarios in the resource level. The resource scenario alternatives given in the previous chapter were illustrated by various practical examples. The CME concept allows coexistent resource type descriptions. For simplicity and with regard to the prototypical implementation, we consider only a small subset of possible resource parameters (i.e. network bandwidth, host CPU).

3.2.2 Mapping between User and Application Level

In order to map the user level QoS specification onto media service specific parameters, the CME architecture employs a set of mapping functions. These functions are similar to the “benefit functions” found in [39] and require the execution of subjective tests.

- Experimental Test Suite

Some tests suites were executed in order to evaluate the user’s perception of a given media service using a quality rating on a 1 to 5 scale. In our experiments we presented video and audio sequences to a group of ten people asking them to rate the perceived quality using the given user level quality scale. As test media services we used the MBone tools *vic* [24] for video transmission and *vat* [23] for audio transmission respectively.

In all video experiments we used a fixed encoding scheme (JPEG) and a fixed window size (320 x 240). In two different test suites we asked the users to rate independently the temporal and the spatial quality of the shown video sequences.

In the first video test we concentrated on the temporal quality. We presented a reference video sequence with 30 fps. We told the viewers to consider the motion quality of that video sequence to be five and asked them to rate the following video sequences in comparison to the reference sample.

In the second video test we concentrated on the spatial quality. We presented a reference video sequence with the maximum resolution of 100%. Again, we told the viewers to consider the resolution quality of that video sequence to be five and asked them to rate the following video sequences according to the reference sample.

In our audio test we varied the encoding scheme in order to evaluate the audio quality. The PCM encoding scheme was considered to provide quality five.

- Mapping Tables

Table 2 and Table 3 show the test results. The tables present the relationship between the user and media service layer. For each media service one or more service specific mapping tables are included in the QoS Mapper. Several media service specific parameters strongly interrelate with each other. In the *vic* [24] video application, for instance, some video encoding schemes include resolution adjustment and some do not. In the CME prototype, we offer an additional mapping table for each video encoding scheme (JPEG, H.261, NV).

QUALITY	FRAME RATE (FPS)	RESOLUTION (%)
5	25 - 30	65 - 100
4	15 - 24	50 - 64
3	6 - 14	35 - 49
2	3 - 5	20 - 34
1	1 - 2	1 - 19

Table 2: Video Quality Rating for JPEG Video.

QUALITY	ENCODING SCHEME
5	PCM
5	PCM2
5	PCM4
4	DVI
4	DVI2
4	DVI4
3	GSM
2	LPC4

Table 3: Audio Quality Rating.

3.2.3 Mapping between Application and Resource Level

Various application specific parameters can be mapped onto a set of resource specific values. For simplicity we consider QoS requirements for the following resource parameters:

- Network resources {bandwidth (Kb/s)}
- Host resources {CPU type, CPU load (%)}

Since media services offer numerous ways to be manipulated, it is very difficult to correlate media service performance and requirements on resources. Especially for video streams, network bandwidth and CPU load highly depend on the employed encoding scheme, on the window size, the degree of movement (depending on the encoding scheme) or the resolution. In addition, resource utilization is also heavily influenced by the available devices (e.g. hardware video encoder) or by the employed tools. In [33], the run time-performance of the two video tools *vic* and *ivs* is compared using the same environment (SGI Indy 133 MHz, H.261, low-motion, 20 fps). While the measured cpu utilization for ivs was 100%, vic operated below 40%.

Considering the mapping between the quality levels and the media service parameters discussed above, we estimated the resources that are needed to obtain the different quality levels. As a test environment we used Sun sparc5™ workstations and for video we estimated the necessary resources for receiving JPEG video (320 x 240).

Table 4 and Table 5 illustrate the experimental test results for video and audio quality. The tables describe directly the relationship between the media service and resource layers.

QUALITY	DEGREE OF MOVEMENT	FRAME RATE (FPS)	RESOLUTION (%)	BANDWIDTH (KB/S)	USED CPU (%)
5	High motion	25	65	1700	> 100
5	Slow motion	25	65	1650	> 100
5	Still	25	65	1600	49
4	High motion	15	50	840	> 100
4	Slow motion	15	50	820	69
4	Still	15	50	800	37
3	High motion	6	35	270	38
3	Slow motion	6	35	260	34
3	Still	6	35	260	21
2	High motion	3	20	102	16
2	Slow motion	3	20	102	14
2	Still	3	20	100	7
1	High motion	1	1	17	6
1	Slow motion	1	1	16	6
1	Still	1	1	16	5

Table 4: Mapping of Video Quality to Resources for JPEG Video.

QUALITY	ENCODING SCHEME	BANDWIDTH (KB/S)	USED CPU (%)
5	PCM	68	< 1
5	PCM2	66	< 1
5	PCM4	64	< 1
4	DVI	38	~1
4	DVI2	35	~1
4	DVI4	34	~1
3	GSM	15	~26
2	LPC4	7	~11

Table 5: Mapping of Audio Quality to Resources.

4 Quality of Service Control

4.1 Introduction and Related Work

Quality of Service Control comprises in general a variety of mechanisms that are applied in order to satisfy the user requirements with respect to the quality of media services. Control mechanisms can be employed at the user, application and resource layers.

Some research groups have concentrated on QoS control issues [1], [8], [26], [40], [49] for executing multimedia applications in a distributed environment. In [8] and [40], a mechanism is described for dynamic adjustment of the bandwidth requirements of multimedia applications based on the Real Time Protocol (RTP) [43]. The sending application uses RTP receiver reports to compute packet loss. Based on these metrics, the congestion state seen by the receivers is determined and the bandwidth is adjusted by a linear regulator with a dead zone.

Similarly to [8], in [26] a feedback control mechanism is presented. Differently from end-to-end mechanisms, network switches send their buffer occupancies and service rates back to the source. The source receives the reports and recomputes the media service parameters. Unfortunately the control mechanism is not designed to scale for multicast distributions.

In [1], a Priority Encoding Transmission (PET) scheme is presented as an approach to the transmission of prioritized information over lossy packet-switched networks. The source assigns different priorities to different segments of data, encodes the data using multi-level redundancy and disperses the encoding into the packets to be transmitted. The destination is able to recover the data in a priority order based on the number of received packets per message.

The problem of scalability in multicast distributions is addressed in [49]. Video gateways and layered encoding schemes are presented to deal with this problem. Video gateways take as input an encoded flow using a scheme with certain bandwidth requirements and forward this flow down the multicast tree using another scheme with different, usually lower, bandwidth requirements. A layered encoding scheme splits the video flow generated by the source into multiple flows, each one with different bandwidth requirements than the original flow. A layered or hierarchical encoding scheme is employed to encode the flows. While all flows are transmitted to non-congested branches of the multicast tree, only the basic flows are transmitted to the congested branches.

The QoS control part of the CME architecture is designed to integrate various controlling mechanisms that use status information at different layers. The CME control mechanisms regard the users requirements, the application and the resource state, recompute the media service

parameters by employing the CME mapping mechanisms and finally control the media services through the Service Manager.

4.2 The CME Quality of Service Controller

The CME Quality of Service Controller implements the different control mechanisms of the CME architecture and operates at the user and resource levels.

A control mechanism in general is specified by a rule. A rule is divided into condition and action parts. A rule base contains the various rules that can be applied through a session. In case a ruling condition becomes true, the rule “fires” and invokes the rule’s action part (Figure 4.1).

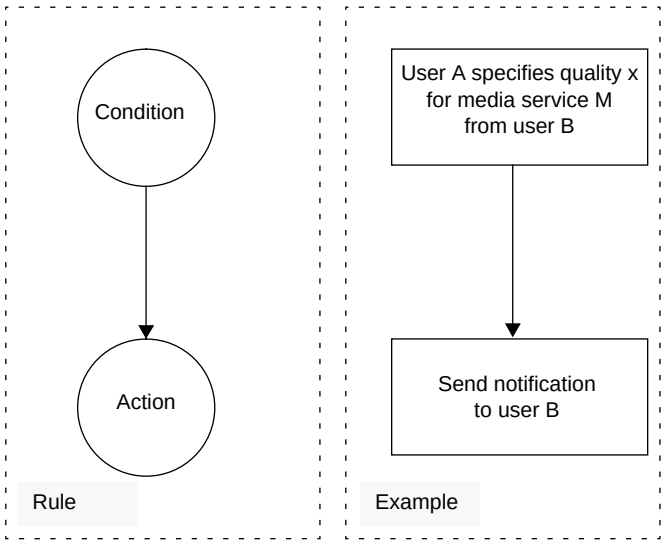


Figure 4.1 Rule Description and Example.

For a condition that holds true we use the term *event*. The QoS Controller handles events on the user and resource level. Depending on the event type and the QoS scenario, the QoS Controller invokes different actions to handle the various events.

4.2.1 The CME Event Classification

Figure 4.2 depicts the CME event classification. Nodes represent event types. The hierarchy refines stepwise event types by splitting each parent node into child nodes. The leaves at the end of the hierarchy represent concrete events.

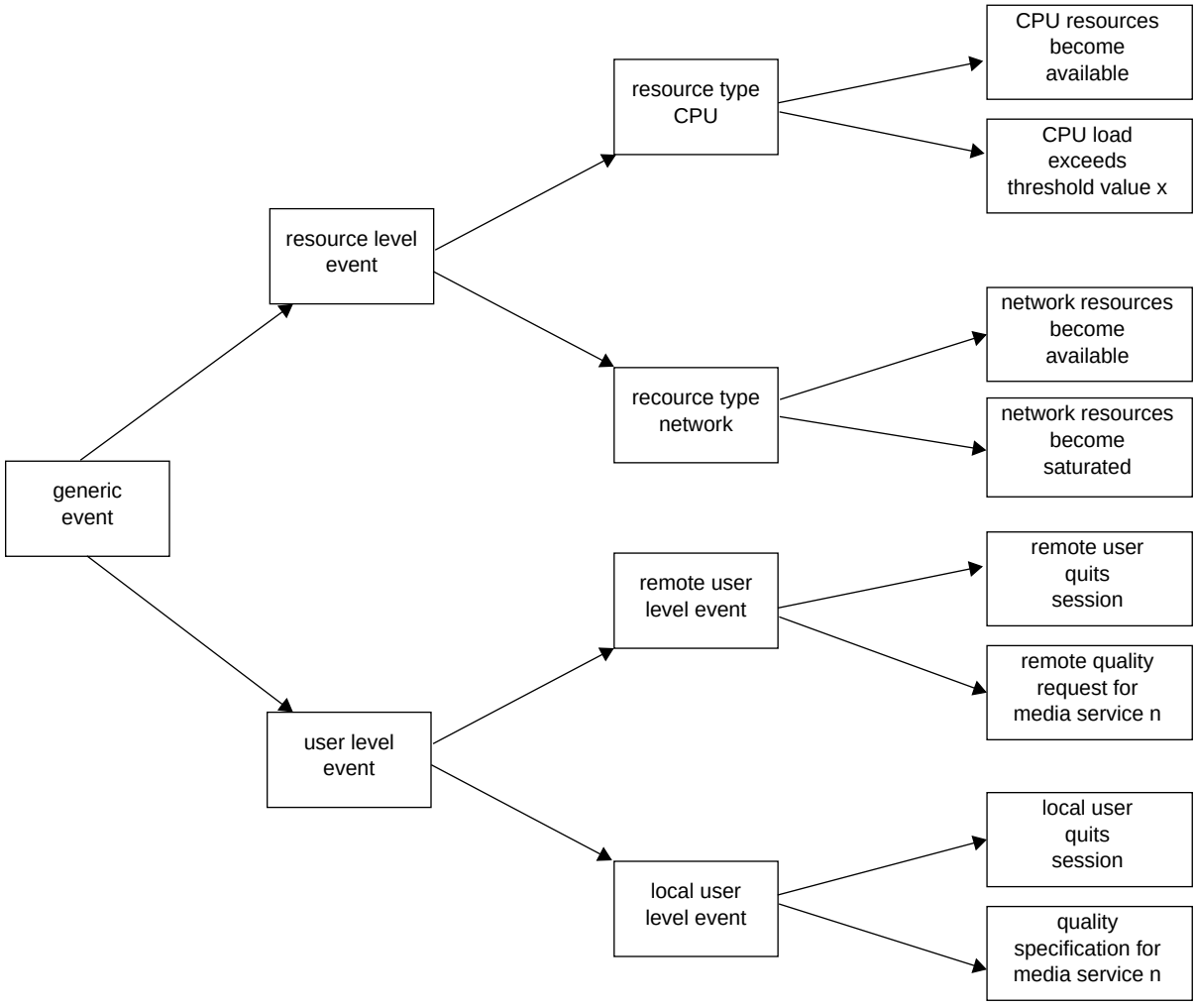


Figure 4.2 The CME Event Hierarchy.

One of the most essential properties of the CME event hierarchy is its extensibility. This means particularly, that new events can be integrated easily into the CME QoS Controller. In the current stage of the CME architecture, the QoS Controller distinguishes between user initiated events and resource driven events. The following sections will examine in further detail how the various control mechanisms, initiated by these events, operate.

4.2.2 User Initiated Control Mechanisms

In a cooperative session, participants normally do not use all the involved media services at the same time. For example, at the beginning of a cooperative session, a user prefers to see and talk to other users in order to exchange the basic ideas on the common work and decide how to proceed with it. Once started working on a common issue, the interests are mainly directed to the shared application while the participants communicate using audio media services.As a consequence, user interests on the employed media services are likely to change during a cooperative session. In the CME architecture each participant is provided with a graphical interface to express his requirements for each media service he receives from any other participant.

For each quality request for a specific media service that a participant wants to receive from another participant, an event is generated and the quality request is sent to the involved site. On the other hand, whenever a participant receives a remote quality request for a media service, a corresponding control mechanism is activated. The control mechanism employs the QoS Mapper functionality in order to recompute new media service parameters according to the remote quality request. In case the resource layer provides QoS guarantees, the corresponding resources are allocated or freed.

In a cooperative session where two participants $P = \{P_1, P_2\}$ work together, a quality request of P_1 to P_2, $q_{requested}(P_1, P_2)$, determines how P_2 has to recompute his new media service parameters. The sending quality of P_2, $q_{send}(P_2)$, is given by:

$$q_{send}(P_2) = q_{requested}(P_1, P_2) \tag{4.1}$$

In case more than two participants $P = \{P_1, P_2, \ldots, P_m\}$ work together, a participant P_i receives quality requests from all the other participants. Since usually one multicast channel is employed to transmit the media services, participant P_i cannot satisfy all the user requirements. The control mechanism of participant P_i recomputes the sending quality by applying a “democratic rule”. The democratic rule averages over all quality requests and is given by:

$$q_{send}(P_i) = \frac{\sum_{j=1}^{i-1} q_{requested}(P_j, P_i) + \sum_{j=i+1}^{m} q_{requested}(P_j, P_i)}{m-1} \tag{4.2}$$

The table that we provide at the end of the chapter, lists all the user-level CME control mechanisms. They are similar in that they are initiated by a participant action. Participant actions are for example an explicit quality request for a media service or quitting a cooperative session.

Participant actions usually cause events on at least two participant sites. For example, a new quality request of participant P_i for participant P_j invokes the QoS Controller at site *i*. The QoS Controller sends the quality request to the QoS Controller at site *j*. The QoS Controller at site *j* applies the "democratic rule" to obtain the new user level quality. It then employs the QoS Mapper to determine the corresponding media service parameters and resource requirements, and transmits the new media service parameters to the Service Manager. The Service Manager finally modifies the settings of the media service with reference to its sending site *j* (Figure 4.3).

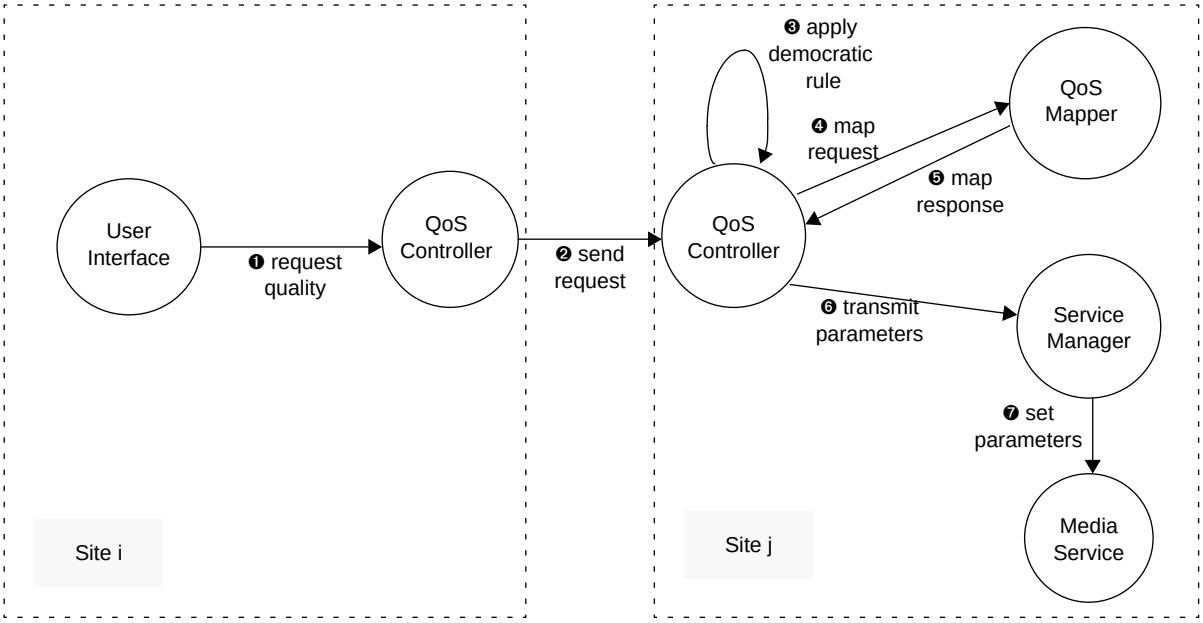


Figure 4.3 Event Trace for a Quality Request.

4.2.3 Resource Level Control Mechanisms

Resource level control mechanisms react in general to resource-state changes. As already mentioned, the CME architecture takes into consideration the cpu load and the network bandwidth as resource types. The Resource Monitor/Controller is the CME component that monitors the state of the cpu and the network bandwidth.

In case resources do not provide any QoS guarantees, they can get saturated or become available during a cooperative session. The Resource/Monitor Controller generates events if certain resource states are entered. If the Resource Monitor/Controller notices for example that the cpu load exceeds an upper threshold, the QoS Controller is informed that the local cpu resources became saturated. The QoS Controller will employ a control mechanism that reacts to the cpu saturation event by invoking the QoS Mapper. Since the mapping functions take the resource status into account, they will try to find media service parameters that are less cpu demanding in order to retain the sending quality. If this is not possible, the sending quality is gracefully decreased and/or lower quality requests for the receiving quality are sent to the other participants (Figure 4.4).

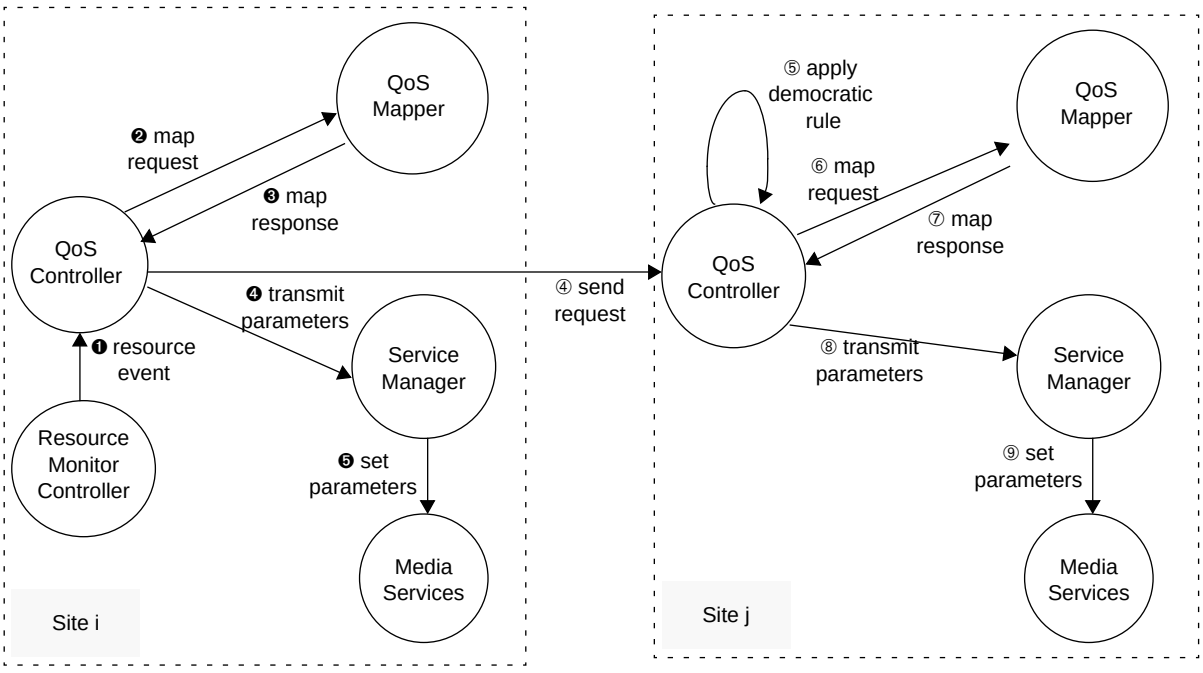


Figure 4.4 Resource Event Trace with Site i and j

4.2.4 Tabular Summary

In order to summarize the behavior of the various CME QoS control mechanisms we provide a tabular summary. Table 6 distinguishes thereby between resource driven and user initiated control mechanisms.

The event column specifies the event that caused a control mechanism to be executed. The *event generated by* and the *indication* columns provide information about the CME component that generated the event and the indication that led the CME component to generate the event. The *QoS* column specifies if the resources provide any QoS guarantees. The *additional condition* column specifies if an additional condition has to be true in order to apply the action part of the control mechanism. The *action* column comprises local actions that have to be performed for each control mechanism. The *result space* finally represents a set of results that can occur as a result of the taken actions.

EVENT	EVENT GENERATED BY	INDICATION	QOS SCENARIO	ADDITIONAL CONDITION	ACTIONS	RESULT SPACE
Resource level events						
Network resources get saturated	Resource Monitor Controller (RMC)	Packet loss exceeds loss threshold and CPU is not saturated	No QoS		Invoke QoS Mapper	Modified media service parameters with less bandwidth demand but same user quality Graceful degradation of sending and/or receiving quality
CPU resources get saturated	Resource Monitor Controller (RMC)	Monitored CPU load exceeds upper threshold value	No QoS		Invoke QoS Mapper	Modified media service parameters with less bandwidth demand but same user quality Graceful degradation of sending and/or receiving quality
Network resources become available	Resource Monitor Controller (RMC)	Packet loss falls below loss threshold	No QoS	Sending quality < requested quality	Invoke QoS Mapper	Higher sending quality through new media service parameters
CPU resources become available	Resource Monitor Controller (RMC)	Monitored CPU load exceeds upper threshold value	No QoS	Sending quality < requested quality	Invoke QoS Mapper	Higher sending quality through new media service parameters
Network or CPU resources become available	Resource Monitor Controller (RMC)	System reports resource becomes available	QoS	Sending quality < requested quality	Invoke QoS Mapper	Higher sending quality through new media service parameters Resource allocation
User level events						
Remote quality request for media service n	QoS Mapper/ Controller	New user level quality request value	No QoS		Invoke QoS Mapper	Change of media service parameters according to remote quality request
			QoS		Invoke QoS Mapper	Allocation/deallocation of corresponding resources and change of media service parameters according to remote quality request
Remote user quits session	Connection Manager	Disconnection request	No QoS		Recompute new requested sending quality Invoke QoS Mapper	Change of media service parameters
			QoS		Recompute new requested sending quality Invoke QoS Mapper	Allocation/deallocation of corresponding resources and change of media service parameters
Local quality request for remote media service n	User Interface	User input	No QoS		Send quality request	
			QoS		Send quality request	
Local user quits the session	User Interface	User input	No QoS		Send notification to session participants	
			QoS		Send notification to session participants	Resource deallocated for employed media services

Table 6: The CME QoS Control Mechanisms

5 The CME Experimental Prototype

5.1 Introduction

In order to evaluate the architectural framework of the Cooperative Multimedia Environment presented in the previous chapters, we have implemented an experimental prototype. The CME prototype accomplishes the main architectural goals. It offers in detail an interface for:

- the user to specify absolute and relative QoS requirements for media services;
- to adjust the media services performance dynamically depending on the resource status;
- to monitor and/or control the resources.

The prototype has been implemented by using the Sun Solaris™ 2.4 operating system. The programming environment comprises the ANSI-C [27] and Tcl/Tk [37] programming languages. For storage of persistent data and for interprocess communication within one system, the relational database *MiniSQL*[19] has been employed. MiniSQL implements a subset of the ISO-SQL92 standard [21]. Finally, for interprocess communication between processes on different systems, the Berkeley socket paradigm [11] has been used.

In Chapter 2, we outlined various QoS scenario alternatives. The experimental prototype considers a resource scenario where neither the network nor the host offer any QoS guarantees. However, the flexibility of the CME architecture allows us to extend the prototype in order to include different QoS scenarios.

The host and network resource properties can be described as follows. Processes residing on a participant’s host offer a time sharing capability. This specific property allows for changing process priorities but does not offer any absolute QoS guarantees. The network resources in turn do not offer any QoS guarantees since the employed media services are based on the IP network protocol.

As already discussed in Chapter 2, each session participant runs a *COoperative MultiMedia Application* (COMMA). A COMMA consists of a set of processes as depicted in Figure 5.1, namely a Session Manager, a Resource Monitor/Controller, the CME Database, a set of Media Services and a set of adjacent Media Service Monitors. We denote that from a process oriented point of view the Resource/Monitor Controller has been split off the Session Manager because it runs as an independent process.

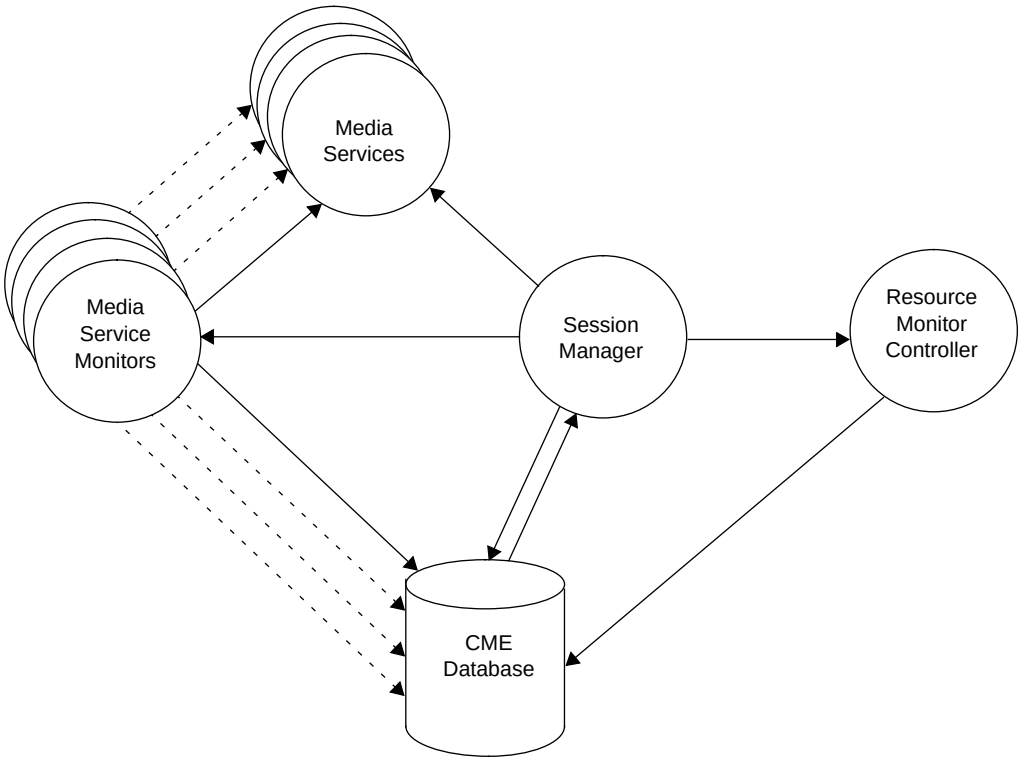
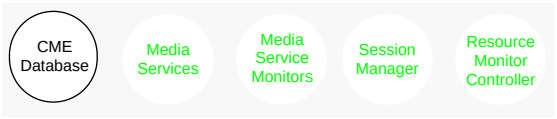


Figure 5.1 Process Oriented View of a Cooperative Multimedia Application (COMMA).

In the remainder of this chapter we will examine in detail various implementation aspects of the COMMA prototype and its processes. The chapter concludes with a detailed illustration of the functional relationship between the COMMA components.



5.2 The CME Database

All CME components exchange their data through the CME database which consists of a set of tables that are mainly employed to store monitored information and to register the invoked media services and corresponding monitors.

5.2.1 Environment

The CME prototype uses *Mini SQL* [19] as a database engine. Mini SQL, or mSQL, is a lightweight relational database engine designed to provide fast access to stored data with low memory requirements. As its name implies, mSQL offers a subset of SQL as its query interface in accordance with the ISO-SQL [21] specification.

The most important property of mSQL with regard to the CME prototype is its C language API. The API allows any C program to communicate with the database engine through the *msqld* database daemon. The API and the database engine have been designed to work in a client/server environment over a TCP/IP network.

5.2.2 Extendend Entity Relationship Model (EERM)

The Extended Entity Relationship Model (EERM) that is given below (Figure 5.2) illustrates in detail the information that is stored in the CME database. The EERM thereby represents a conceptual model that describes the view of one session site.

The entity *Active Media Service* comprises the attributes that characterize a running media service. An Active Media Service is mainly specified by a process identifier and a media service name. For each Active Media Service, a *Media Service Monitor* is launched. The monitor itself is a weak entity since it depends on the existence of a media service.

The Active Media Service entity is a generalization of a specific media service. As depicted by the ER-diagram, the two entities *vic* and *vat* are examples of specific media services including their own attributes.

The entity *Session Participant* specifies all participants that are involved in the current session. The relationship *Monitor Sample* is built together with a time identifier obtained from the *Timer* entity and together with each specific media service entity.

The *Quality Wish* relationship expresses the quality requirements in user terms between two session participants for an active media service.

The *Resource Utilization* relationship specifies the resource requirements of an active media service at a certain time. Finally, the *Monitored Host Resources* relationship specifies the state of the entity *Host Resources* at a certain time indicated by the entity *Timer.*

The employed ER-notation does not follow exactly the graphical notation that was originally proposed by Chen [9]. The used extensions comprise a hexagon in order to express generalization, a double lined rectangle that stands for a weak entity object and a list of entity attributes that is included in each entity rectangle and relationship diamond.

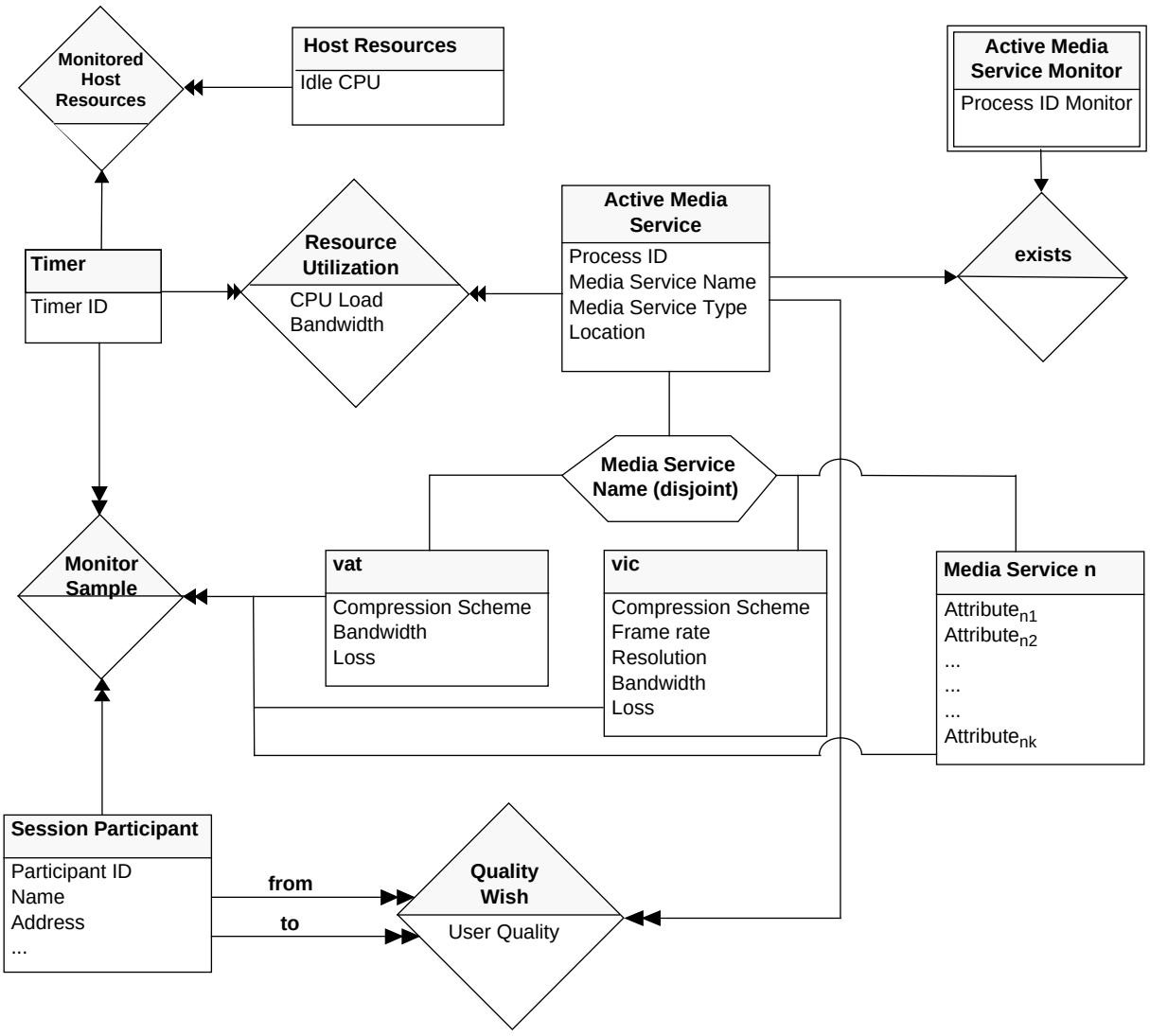


Figure 5.2 ER Notation of the CME Information Structure.

5.2.3 Relational Database Design

Based on the graphical ER-notation, the diagram is mapped onto a relational database scheme that logically appears as a simple collection of tables. The translation process from the conceptual model (depicted by the ER-diagram) into the relational table model leads to the following set of relations:

$$T\_ActiveMS(\underline{A\_ProcessID},A\_MSName)$$

$$T\_ActiveMSM(\underline{A\_ProcessID},A\_MSName)$$

$$T\_VicMonitor\left(\begin{array}{l}\underline{A\_TimerID},\underline{A\_ParticipantID},A\_Compression,A\_FrameRate\\A\_Resolution,A\_Bandwidth,A\_Loss\end{array}\right)$$

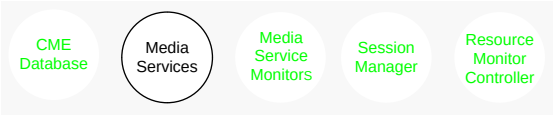
$$T\_VatMonitor(\underline{A\_TimerID},\underline{A\_ParticipantID},A\_Compression,A\_Bandwidth,A\_Loss)$$

$$T\_Wish(\underline{A\_ParticipantID},\underline{A\_MSName},A\_UserQuality)$$

$$T\_ResourceMonitor(\underline{A\_TimerID},\underline{A\_IdleCPU})$$

Since there are numerous ways to map associations or generalizations to tables, the relational table model represents only one of many possible solutions. Furthermore, in order to reduce the number of tables, the relational table model has been denormalized.

Each session site is provided with a collection of tables as illustrated above. The applied terminology uses the prefix *T_* to specify a table and the prefix *A_* for a table attribute. The underlined attributes build together the relation's primary key.



5.3 Media Services - MBone tools

The COMMA prototype uses as media services the MBone tools developed at the UC Berkeley (UCB) and the Lawrence Berkeley National Laboratory (LBNL), i.e. the video conferencing tool *vic* [24] for video, the audio tool *vat* [23] for audio and the whiteboard tool *wb* [25] as a whiteboard. The tools are briefly characterized through the following descriptions.

- Video service *vic*

The UCB/LBNL video tool, vic[24], is a real-time, multimedia application for video conferencing over the Internet. Vic is designed with a flexible and extensible architecture to support heterogeneous environments and configurations. For example, in high bandwidth settings, multi-megabit full-motion JPEG streams can be sourced using hardware assisted compression, while in low bandwidth environments like the Internet, aggressive low bit-rate coding can be carried out in software.

Vic is based on the Draft Internet Standard Real-time Transport Protocol (RTP) developed by the IETF Audio/Video Transport working group [43]. RTP is an application-level protocol implemented entirely within vic.

- Audio service *vat*

The LBNL audio tool, vat [23], is a real-time, multi-party, multimedia application for audio conferencing over the Internet. Vat, like vic, is based on the Draft Internet Standard Real-time Transport Protocol (RTP) developed by the IETF Audio/Video Transport working group. RTP is an application-level protocol implemented entirely within vat.

- Whiteboard service *wb*

Wb [25] is a remote conferencing tool that provides a distributed whiteboard. The whiteboard separates the drawing into pages, where a new page can correspond either to a new viewgraph in a talk or to the clearing of the screen by a member of a meeting. Any member can create a page and any member can draw on any page.

5.3.1 The Real-time Transport Protocol (RTP)

As already mentioned in the previous section, the MBone tools vic and vat are based on the Draft Internet Standard Real-time Transport Protocol (RTP) [43]. RTP is a connectionless application level protocol. It is usually implemented as part of the application.

RTP does not provide any Quality of Service guarantees. It neither offers any mechanisms to provide packet loss nor in-order delivery guarantees. However, it offers timestamps and sequence numbers for RTP packets in order to detect packet delays and packet losses.

The RTP protocol is divided into two sub protocols: the *data delivery protocol* and the *real-time transport control protocol* (RTCP). The data delivery protocol provides functionality for the determination of media encoding, framing, error detection, encryption and source identification. RTCP manages control information like sender identification, receiver feedback and inter-media synchronization. Each session participant sends RTCP packets periodically to all other session participants. The time interval between the sending of two consecutive RTCP packets is randomized and adjusted to the number of participants in the session in order to keep the RTCP bandwidth under a certain limit.

One of the main features of the RTCP protocol is to support a distributed monitoring of QoS parameters. Participants in a multimedia session provide quality feedback to all other session members by issuing RTCP Sender Reports (RTCP-SR).

Since the *vic* and *vat* MBone tools do not use the control information provided by the RTCP protocol, it is possible to collect the RTCP control information independently by listening to the media service related RTCP ports.

5.3.2 The Tcl/Tk Send Command

As depicted in Figure 5.1, the Session Manager and the Media Service Monitors set and retrieve data from the media services. One of the key design issues of the CME prototype is to integrate the media services into the CME framework without modifying their source code. Since all MBone tools are implemented in C++ and Tcl/Tk, the Tcl/Tk *send* property is employed as an interface to the media services.

The *send* command [37] provides a powerful form of communication between applications. With *send*, any Tk application can invoke an arbitrary Tcl script in any other Tk application on the display; these commands can both retrieve information and also take actions that modify the status of the target application.

5.4 Media Service Monitors

The media service monitors retrieve information directly from the media services. For each media service, a corresponding media service monitor is launched. Each media service monitor is an independent process that periodically polls information from its media service and writes it into the CME database.

The most essential design issue for the media service monitors, is to retrieve the desired information without modifying the source code of the media services. As already mentioned in the media service description, the MBone tools include two properties that allow us to monitor them without modifying source code; they employ the RTP application-level protocol and offer a Tcl/Tk interface. Since all MBone tools provide a Tcl/Tk interface, the *send* command is used to communicate with the media services.

The Media Service Monitors directly access the media service data structures where statistics information is stored. A set of Tcl/Tk procedures, employing the *send* command, is used to retrieve these data structures. The data structures mainly comprise media service related information (e.g. sending rate, receiving rate, loss rate, bandwidth usage) that has been computed by the media services based on sent and received RTP packets. The retrieved data is finally written into the corresponding tables of the CME database (Figure 5.3).

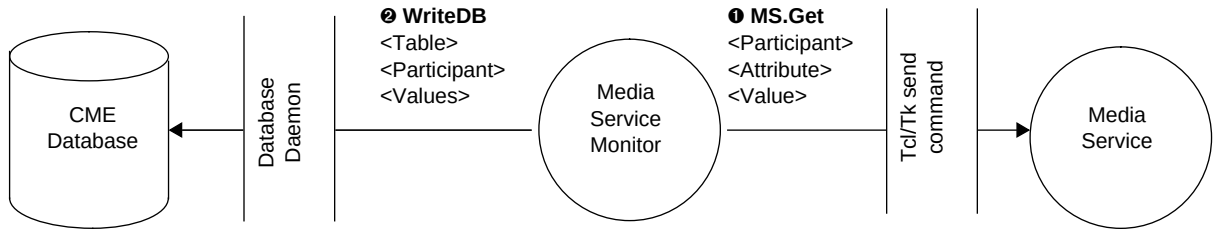


Figure 5.3 The COMMA Media Service Monitor.

- Interface to Media Services

The media service attributes are polled periodically. Depending on the desired accuracy of the QoS control mechanisms that process the monitored data, a suitable value for the poll interval is specified.

As depicted in Figure 5.3, a media service monitor uses a parametrized *get* method to retrieve media service specific data. The *get* method consists of a set of Tcl/Tk procedures using the *send* command. It encapsulates the retrieval of the media service specific data structures. The input parameters of the *get* method are a unique identifier for each participant and the name of the attribute whose value the monitor has to retrieve.

Table 7 offers a detailed overview of the *get* methods for the *vic* and *vat* media services. The table comprises the monitor method, the attribute names and the value range of the attributes.

MONITOR METHODS	ATTRIBUTE NAME	VALUE RANGE
Vic.Get	Compression	jpeg, h.261, nv
	Frame rate	0 .. 30 fps
	Bandwidth	0 .. 3072 Kb/s
	Loss	0 .. 100 %
Vat.Get	Compression	dct, dvi, gsm, lpc
	Bandwidth	0 .. 70 Kb/s
	Loss	0 .. 100 %

Table 7: Vic.Get and Vat.Get Monitor Methods.

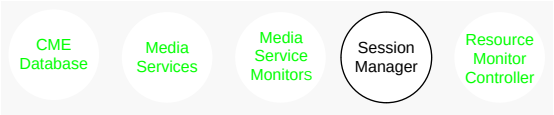
- Interface to the CME database

As depicted in Figure 5.3, the media service monitor stores the retrieved data in the CME database where a separate table for each media service is employed. The launched media service monitors carry a row number parameter, that specifies an upper threshold for the stored media service samples. If the number of retrieved samples exceeds the row number parameter, the tables behave like a FIFO buffer.

Table 8 offers a description of the *T_VicMonitor* and *T_VatMonitor* tables. In addition to the monitored data, the tables contain the counter attribute *A_Counter* that serves as a timestamp and as a primary key in combination with the participant attribute.

TABLE NAME	TABLE ATTRIBUTE	SAMPLE
T_VicMonitor	A_Counter	251
	A_Participant	nikolaus@128.32.201.23
	A_Compression	jpeg
	A_FrameRate	15 f/s
	A_Bandwidth	700 kB/s
	A_Loss	9%
T_VatMonitor	A_Counter	251
	A_Participant	nikolaus@128.32.201.23
	A_Compression	dct
	A_Bandwidth	68 kB/s
	A_Loss	11%

Table 8: The T_VicMonitor and T_VatMonitor Tables.



5.5 Session Manager

5.5.1 User Interface

The COMMA User Interface is mainly split into two parts. The first part provides a graphical user interface for the connection management and it is employed by the session initiator who creates a session specific invitation message. The second part provides a graphical user interface for the session management. It is employed by all session participants in order to specify their media service requirements.

- **User Interface - Connection Management**

The connection management part (Figure 5.4) of the COMMA User Interface is employed by the session initiator. The following session specific settings can be specified:

- A participant list that includes all participants the initiator wants to invite to the cooperative session. A session participant is specified through his username and the host address following the syntax <username>@<hostaddress>.
- The media services he wants to use during the session. The current CME prototype offers video, audio and whiteboard services. For each media service the user can specify an initial quality that is mapped into media service start-up parameters. The session initiator also specifies a modifiable predefined UDP port number for each media service.
- The address of the cooperative session. In case the cooperative session involves only two participants (two party session), the address widget specifies the unicast address of the user's host the session initiator wants to invite. If the cooperative session involves more than two participants (multiparty session), the address widget specifies a multicast address that is employed by all participants.
- The scope radio buttons determine the scope of the cooperative session. Possible settings for the scope are site, region and world. If the destination address is not an IP multicast address, the scope is ignored. The number that is displayed in the entry widget beneath the scope radio buttons specifies the time to live (ttl) value. A ttl value of 2 restricts the traffic to the local net and is connected to a local scope; a ttl value of 16 is connected to a regional scope and a ttl value of 127 to a worldwide scope respectively.

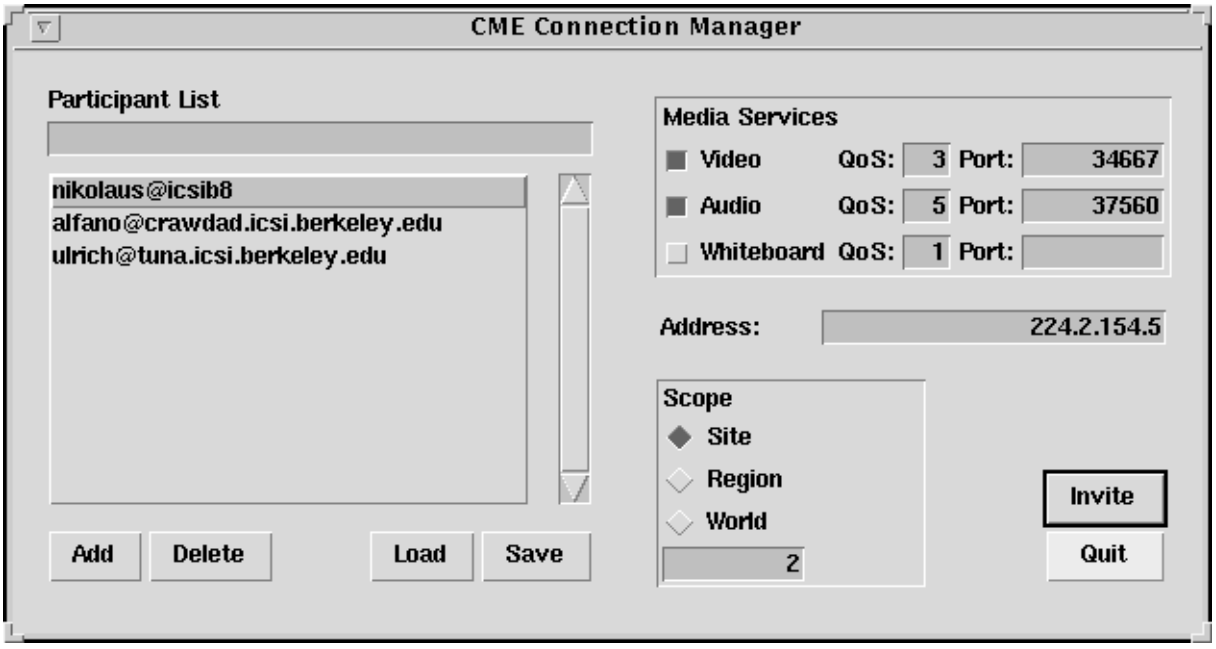


Figure 5.4 User Interface for Initiating a Session.

- **User Interface - Session Management**

The session management part (Figure 5.5) of the COMMA User Interface is employed by all session participants. It includes the following functional parts:

- A list of all session participants. The media service sliders and the quality meters relate to a selected participant in the list.
- A media service slider for each media service (e.g. video, audio). A slider value for a selected participant indicates the media service quality the user wants to receive from that participant. If the user himself is selected in the participant list, the slider values indicate the averaged quality requirements of the other session participants.
- A quality meter for each media service. The quality meter displays the currently received quality for a selected participant. If the user himself is selected in the participant list, the quality meter displays the current sending quality. For each media service the quality display ranges from zero to five. Level zero indicates that the service is not being received. The other levels relate to the quality rating presented in Chapter 3.

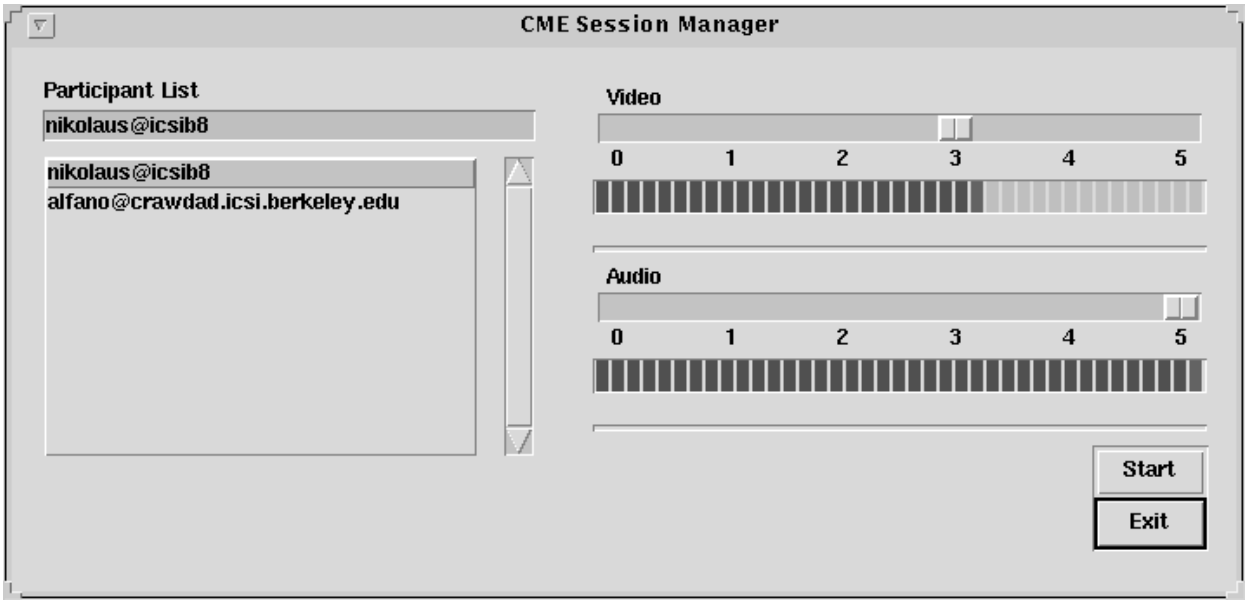


Figure 5.5 Control Interface for the Media Services.

5.5.2 Connection Manager

The CME prototype employs UNIX sockets [46] in order to provide connection management functionalities. Sockets are a client/server paradigm that perform exactly like UNIX files or devices, so they can be used with traditional primitives like *read* and *write*. The CME prototype makes use of this specific socket property by employing Tcl/Tk file handlers that provide event-driven mechanisms for reading and writing files that may have long I/O delays.

With the invocation of a COMMA, the Connection Manager is initialized and enters an idle state where it can send invitations or wait for invitations. Thus, the relationship between Connection Managers can be characterized by a peer-to-peer model. Since the underlying communication primitives are UNIX sockets that follow the client/server paradigm, this means specifically that a Connection Manager may act at the same time as a client and as a server.

The Connection Manager of the user who initiates a session is depicted on the left side of Figure 5.6. After the session initiator has specified his session requirements, the Connection Manager assembles the input to an invitation message.

INVITATION MESSAGE	EXAMPLE		
Participant List	nikolaus@icsib8.icsi.berkeley.edu (session initiator)		
	alfano@crawdad.icsi.berkeley.edu		
	ulrich@tuna.icsi.berkeley.edu		
Media Services with initial qualities and port numbers	Vic (Video)	Q_{init}: 3	Port: 4568
	Vat (Audio)	Q_{init}: 5	Port: 5190
	Wb (Whiteboard)	Q_{init}: 5	Port: 6374
Multicast Address	224.2.154.5		
Session Scope	Site: (ttl=16)		

Table 9: Invitation Message sent by the Connection Manager.

Before the invitation message is sent to the users specified in the participant list, the Connection Manager checks the validity of the invited users. The checking procedure ensures the correctness of the destination addresses. It further examines whether COMMA processes are running on the involved sites, i.e. if the other Connection Managers are waiting for an invitation.

In case the check procedure succeeds, the invitation message is sent to the Connection Managers of the other sites and the *In Session* state is entered. In case the check procedure fails, a list of invalid or not reachable participants is displayed and the Connection Manager returns to the *Idle State*.

The callee's Connection Manager waits for an invitation call by listening to a well defined port. Whenever a session initiator sends an invitation message, the callee's Connection Manager receives the invitation request and processes it. The invited user reacts to the request by accepting or rejecting the invitation. In case the invitation is rejected the Connection Manager returns to the *Idle State*. In case the invitation is accepted the *In Session* state is entered (Figure 5.6).

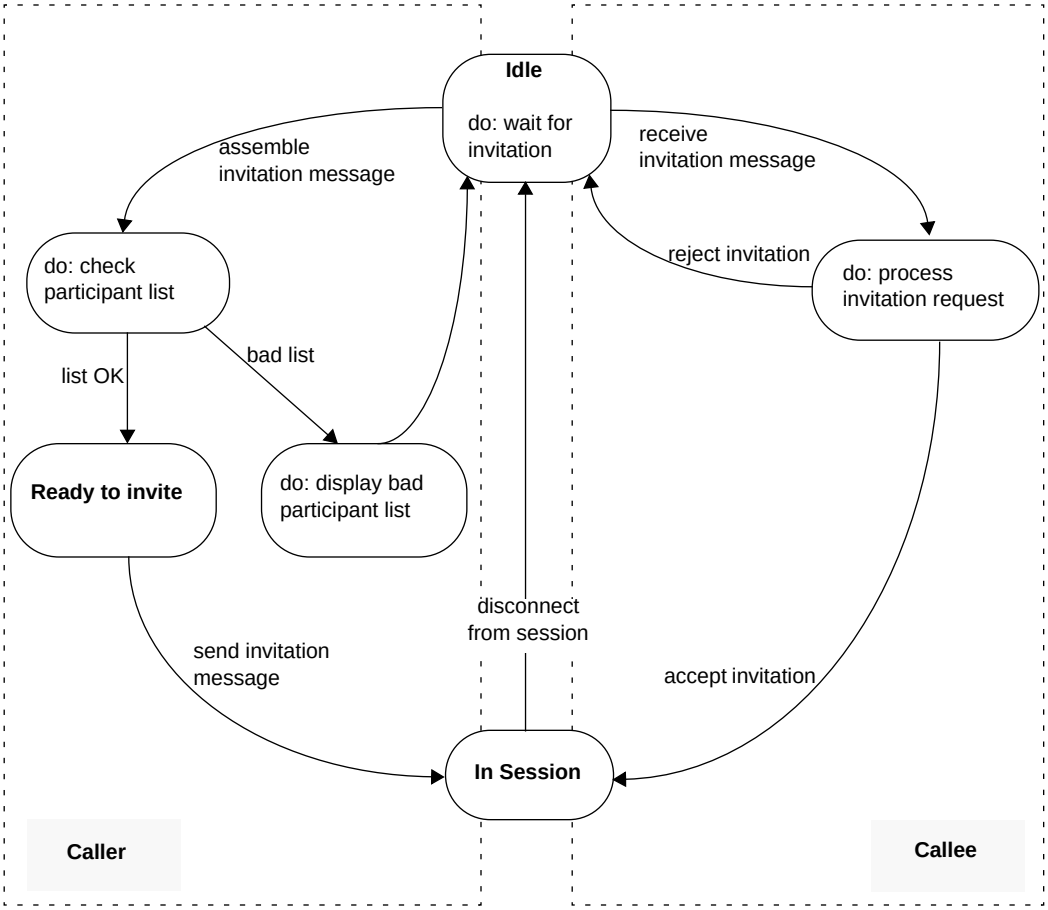


Figure 5.6 State Diagram for Connection Management.

5.5.3 Service Manager

The CME Service Manager provides functionality for the other CME components, mainly for the QoS Mapper/Controller, in order to start and stop media services and to set and get media service parameters. The methods the Service Manager provides are listed in Table 10 .

METHOD	PARAMETER
SeM::StartMS	Media service, initial settings
SeM::StopMS	Media service
SeM::GetMSParameterNum	Media service, parameter, number of samples
SeM::GetMSParameterString	Media service, parameter
SeM::SetMSParameter	Media service, parameter, parameter value

Table 10: Service Manager Methods.

- Start a Media Service

The *StartMS* method comprises several steps that are illustrated in Figure 5.7. First, the specific media service is launched with the initial quality parameters that have been specified by the invitation message. In order to set the initial service parameters, the Service Manager employs a specific media service interface. The communication to the media service processes is realized through the Tcl/Tk send primitive.

After the media service has been successfully launched, the service is registered in the CME database as an *Active Media Service*. The registration comprises the media service name and a process identifier. Finally, the corresponding *Media Service Monitor* is started. In the monitor start-up procedure, a separate monitor process is forked for each media service. The process connects to the CME database, monitors data from the corresponding media service by employing the Tcl/Tk send primitive and writes the monitored information periodically on the database.

After the Media Service Monitor has been successfully launched, the monitor is registered in the CME database as an Active Media Service Monitor. The registration data includes the media service name and the monitor's process identifier.

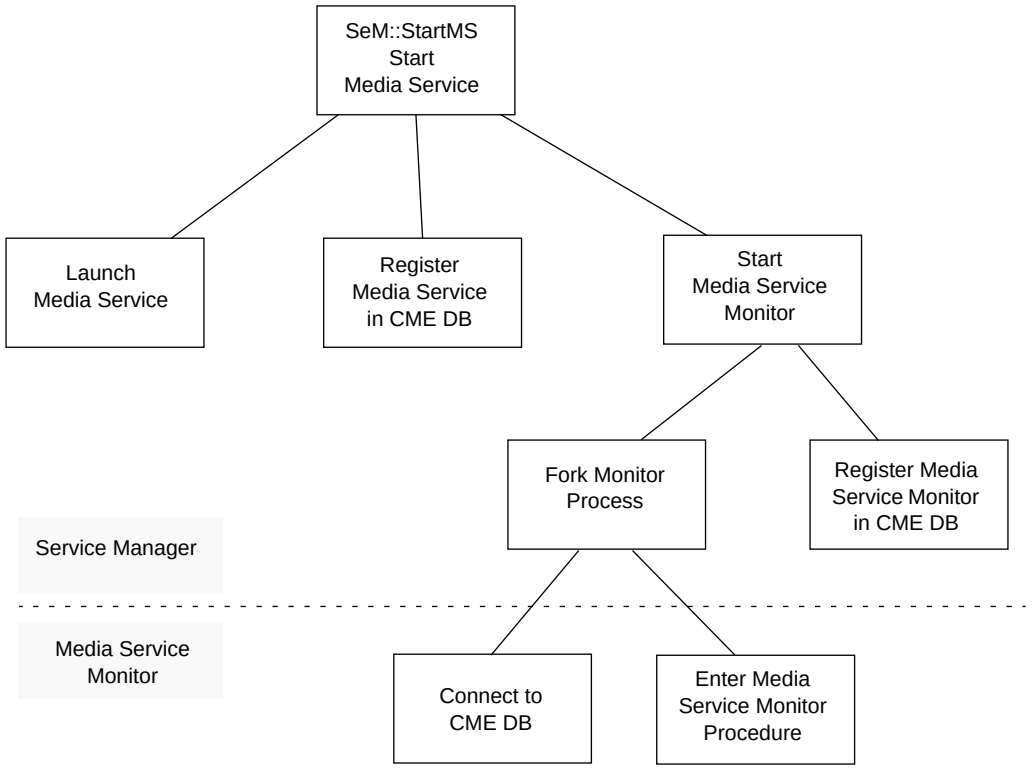


Figure 5.7 Starting a Media Service.

- Stop a Media Service

The *StopMS* method comprises several steps to stop a media service properly as in the *StartMS* method (Figure 5.8). First, the Media Service Monitor is stopped. The Service Manager performs this task by sending a termination signal to the monitor process. The monitor process in turn receives the signal and invokes a signal handling routine to process the signal accurately. With the termination of the monitor process, the database table that contains the monitored data is cleared and the monitor process is unregistered from the list of Active Media Service Monitors. Finally, the Media Service is terminated by employing the Service Manager's interface to the media services. With the termination of the Media Service, the service process is unregistered from the list of Active Media Services.

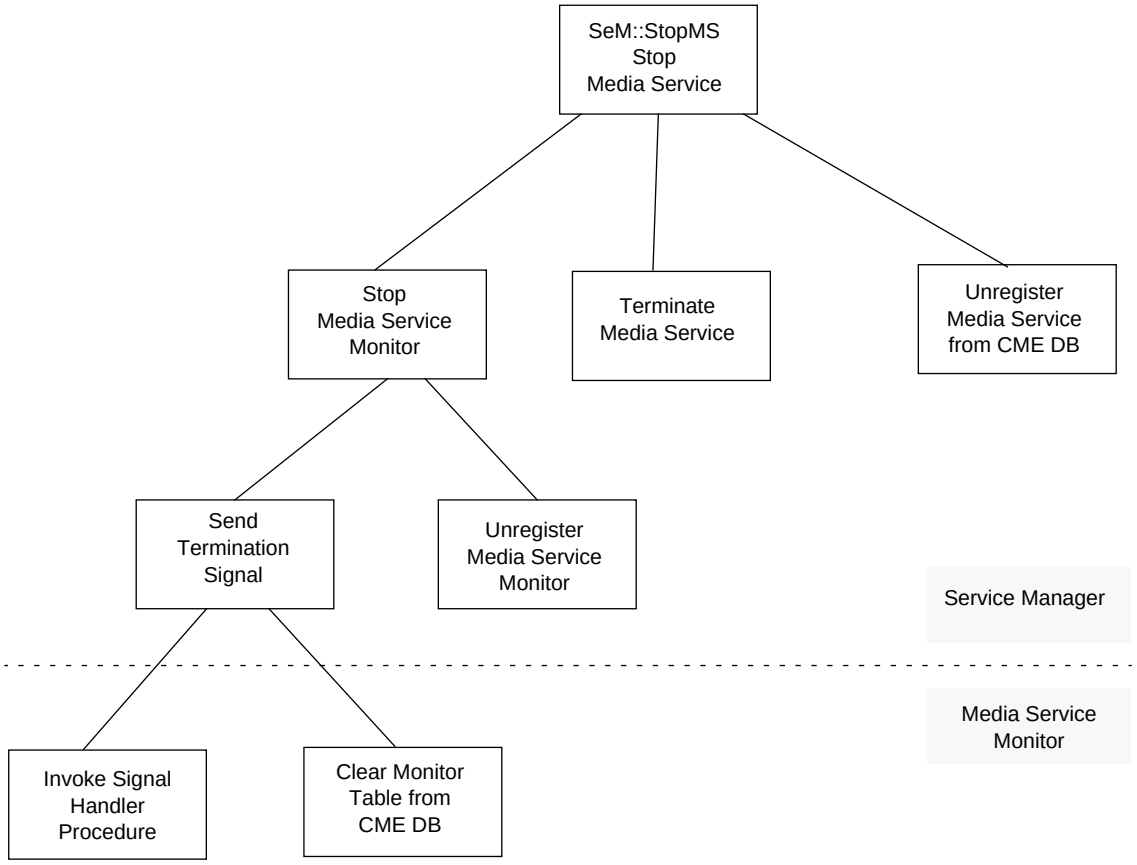


Figure 5.8 Stopping a Media Service

- Set and Get Methods

The Service Manager's *set* method is employed to control the various Media Services. The *set* method is invoked by specifying a media service, a service parameter, and a parameter value. For each media service, the function call is mapped into the customized media service interface of the Service Manager. Finally, the new parameter value is transmitted to the media service by employing the Tcl/Tk send primitive.

With the Service Manager's *get* method, current media service parameters are retrieved from the CME database. In case the requested parameter is a numerical value, an additional *number of samples* parameter specifies the monitoring period that has to be considered for the computation. In case the requested parameter value is a string, the most recent table entry is fetched and returned.

The *set* and *get* methods are mainly employed by the QoS Mapper/Controller and by the User Interface. The QoS Mapper/Controller's mapping tables and control mechanisms retrieve media service parameters, map them into resource requirements, determine new parameter values and adjust media services dynamically. The User Interface uses the set and get primitives to set the initial media service parameters and to refresh the display of the quality meters periodically.

5.5.4 QoS Mapper/Controller

The QoS Mapper/Controller is mainly split into a QoS Mapper and a QoS Controller. The core functionality and design of both components has been already discussed in the previous two chapters since mapping and control aspects represent a crucial part of the CME architecture. We now describe the interfaces that are offered by both components.

- QoS Mapper

For each new media service that is included in COMMA, the QoS Mapper is extended by a media service specific mapping table. The table thereby describes the mapping between user, application and resource layers. The current COMMA prototype includes mapping tables for the *vic* and *vat* media services. Based on the mapping tables, the QoS Mapper provides the mapping functions illustrated in Table 11 .

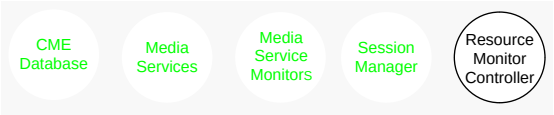
MEDIA SERVICE	METHOD	DESCRIPTION
Vic	QoSMC::VicUser2App	Map user requirements into service parameters
	QoSMC::VicApp2User	Map service parameters into user requirements
	QoSMC::VicApp2Resource	Map service parameters into resource values
	QoSMC::VicResource2App	Map resource values into service parameters
Vat	QoSMC::VatUser2App	Map user requirements into service parameters
	QoSMC::VatApp2User	Map service parameters into user requirements
	QoSMC::VatApp2Resource	Map service parameters into resource values
	QoSMC::VatResource2App	Map resource values into service parameters

Table 11: Mapping Functions for the vic and vat Media Services.

- QoS Controller

In Chapter 4, we discussed the event-driven character of the QoS Controller component. According to this specification, the CME prototype makes extensive use of event handling procedures by employing the corresponding Tcl/Tk library functions.

Two event handling types are mainly integrated in the implemented control mechanisms: file event handlers and time event handlers. File handlers provide an event-driven mechanism for reading and writing files that may have long I/O delays. Time events trigger callbacks after particular time intervals.



5.6 Resource Monitor/Controller

In Chapter 2 we outlined the tasks of the Resource Monitor/Controller. They mainly comprised monitoring and controlling host and network resources.

In our experimental prototype, however, the Resource Monitor/Controller only monitors the availability of host resources and the allocation performed by the Media Services. The consumption of network resources has not to be monitored since this task is already performed by the Media Service Monitors.

- **Monitoring Host Resources**

The available host resources (i.e. idle CPU) are monitored continuously by employing the *iostat* BSD Unix tool. The CPU load of each media service is additionally monitored. The process identifiers of the Active Media Services are thereby directly retrieved from the CME database table *T_ActiveMS* (Figure 5.9).

- **Controlling Host Resources**

In our prototype, Media Service processes run under the time-sharing class. By employing the *priocntl/priocntl_set* library functions, the Resource/Monitor Controller assigns process priorities dynamically to active Media Services. Although this mechanism allows to privilege certain processes, it does not offer QoS guarantees in absolute terms.

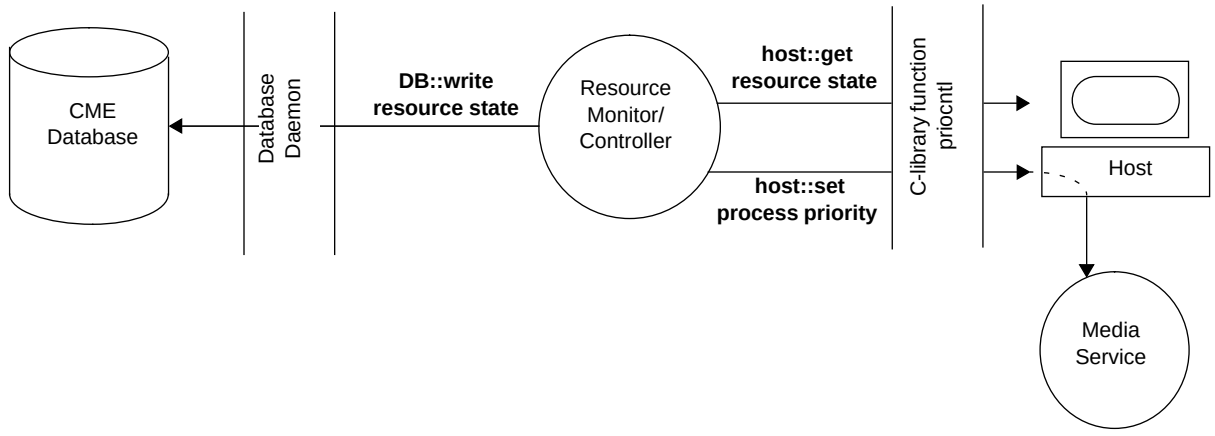


Figure 5.9 Monitoring and Controlling Host Resources.

5.7 Functional Relationship between the COMMA Components

We conclude this chapter by providing a detailed summary of the CME prototype. Figure 5.10 depicts all the COMMA components and their functional relationships. The key issues as illustrated in Figure 5.10 are the following:

- A COMMA consists of a set of processes, namely the Session Manager, the Media Services, the Media Service Monitors, the Resource Monitor/Controller and the CME Database.
- The COMMA processes exchange their information through the CME Database. Interprocess communication between the different COMMA processes that reside on different systems is performed through the socket paradigm.
- Due to the employment of the RTP application protocol, information over the status of other COMMA applications is obtained indirectly by the Media Services.
- Media Service Monitors collect information from adjacent Media Services by employing the Tcl/Tk send primitive. The information is written into the CME Database tables *T_VicMonitor* and *T_VatMonitor* respectively.
- The Resource Monitor/Controller monitors the resource state and writes the retrieved values into the CME Database *T_ResMonitor.*
- The components of the Session Manager (Connection Manager, User Interface, Service Manager, QoS Mapper/Controller) build the core part of COMMA. QoS Mappers of different COMMA sites exchange their quality requirements through the CME Database table *T_Wish*. The Service Manager and the QoS Mapper/Controller include a customized part for setting Media Service parameters and performing appropriate mapping.

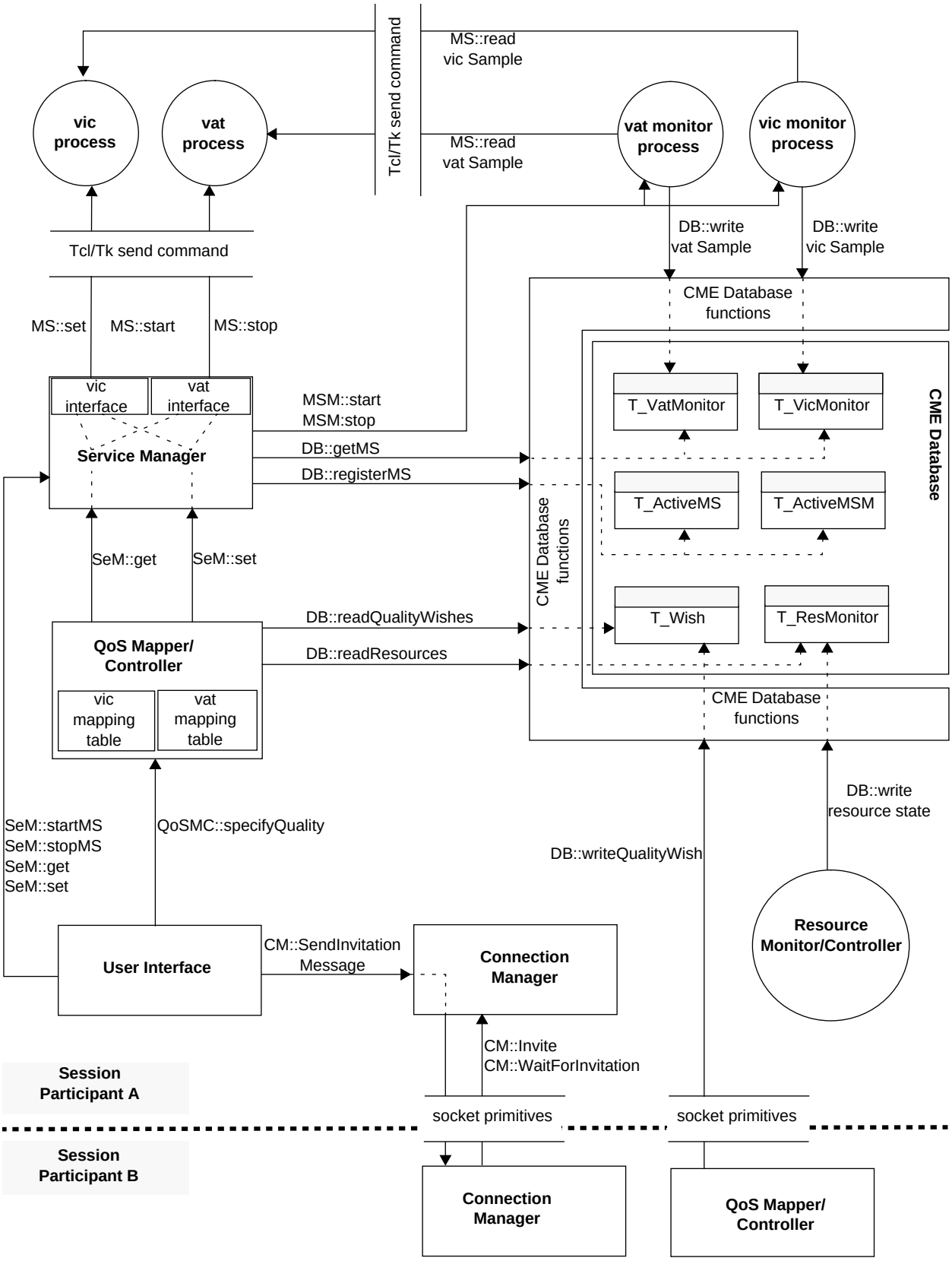


Figure 5.10 The COMMA Components and their interfaces.

6 Conclusions and Future Work

In this work we have presented a Cooperative Multimedia Environment (CME) architecture and its experimental prototype. The architecture allows the integration of multimedia applications in an overall framework and the user-driven control of the QoS of the employed media services. The CME architecture comprises various functional components. Users can specify their QoS requirements in a uniform way and are able to prioritize particular multimedia applications. QoS mapping functions perform the translation from user level QoS representations into application parameters and resource requirements. A set of QoS control mechanisms enables the dynamic adaptation of application parameters depending on user requirements and resource status. Resource monitoring and control lead to an efficient use of resources.

There are still open issues that require further investigation. Among them, a better understanding of user requirements is necessary. More work is also needed in mapping user requirements into media service parameters and system resources. Additional service and resource parameters should be taken into account. Finally, more work needs to be done on defining accurate QoS control mechanisms.

The presented architecture, moving towards the vision of a real multimedia environment, can make the best use of network protocols and operating systems that offer QoS guarantees. Unfortunately, today's dominant network, the Internet, can ensure only a best-effort approach considering data delivery. On the other hand, the deployment of network protocols that offer QoS guarantees [3] has been rather disappointing due to the required transition to new network technologies like ATM.

From the given scenario emerges the need for a set of interrelated protocols that offer QoS guarantees and can be easily integrated in the Internet protocol suite. The IETF has developed its Resource Reservation Protocol [6] (RSVP) that permits the reservation of network bandwidth and assignment of priorities to various traffic types. The IETF chose RSVP for its simplicity and robustness associated with IP and other connectionless protocols.

The Real Time Protocol [43] (RTP) works alongside TCP, providing end-to-end delivery of such data as video broadcasting and multiparticipant audio and video. Feedback on reception quality and optional identification of the receivers of the multicast stream are provided by the real-time transport control protocol (RTCP), which is an integral part of RTP.

With the deployment of these protocols, multimedia applications that run on the Internet will offer QoS guarantees. With such guarantees, integrated control offered by an architecture like the CME will be essential for the quality control of multimedia sessions.

Appendix

Preliminary Measurements

In a set of tests we analyzed how host and network resources influence the execution of multimedia applications. For our tests, we chose an unidirectional video stream. This setup enabled us to examine specifically the CPU consumption for decoding video frames at the receiving host. The complete test setup is illustrated in Figure A.1.

As sending host (host A), we used a Sun^{TM} sparc20 workstation equipped with a Parallax^{TM} video board. The Parallax video board supports JPEG compression in hardware. This enabled us to vary the video frame rate in a wide range without saturating CPU resources. On the receiving side (host B), we used a Sun sparc5 workstation with a Sun Video^{TM} board. The Sun video adapter does not provide the same hardware support as the Parallax board, therefore the decompression has to be done in software which requires high levels of CPU resources.

We executed two sets of experiments. For the first set of experiments, we considered an environment for world-wide collaboration with limited, but guaranteed, network resources. We employed the MAY [34] (Multimedia Applications on Intercontinental Highway) network, an ATM network connecting North America to Europe. By setting up a loopback in Germany, we established a world-wide ATM link with a fixed transmission rate of 1.5 Mb/s. In the second set of experiments we used our local ATM network that provides a transmission rate of 155 Mb/s allowing us to avoid network congestion.

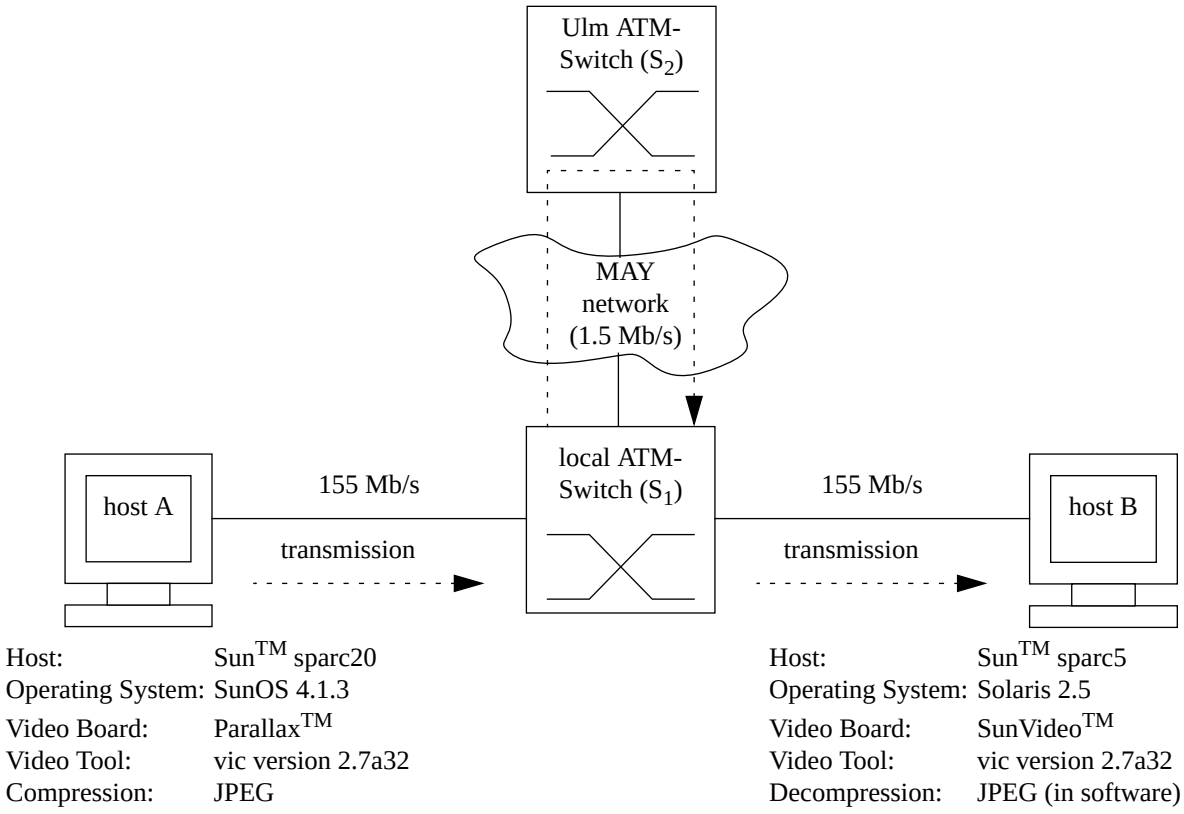


Figure A.1 Measurements Setup.

In the first set of experiments, we used the Mbone tool vic [33] to send video from host A to host B through the world-wide ATM link. For a progression of sent video frame rates we monitored the displayed rate and the CPU load at the receiving side, and the transmission rate of the ATM network. To track the receiving rate, we utilized the vic application itself. To monitor the CPU load, we employed the Unix *top* utility. The ATM network transmission rate was measured by the Synoptics™ ATM switch management tools.

Figure A.2 depicts the functional dependencies of the sending frame rate with the displayed frame rate and the CPU utilization (at the receiving side), and the ATM network transmission rate. The three graphs illustrate the influence of both CPU and network resources on the displayed frame rate. At a sending rate of 16 fps, the CPU became saturated. At 19 fps, the network became additionally congested, leading to the dramatic reduction of the displayed frame rate at the receiving side.

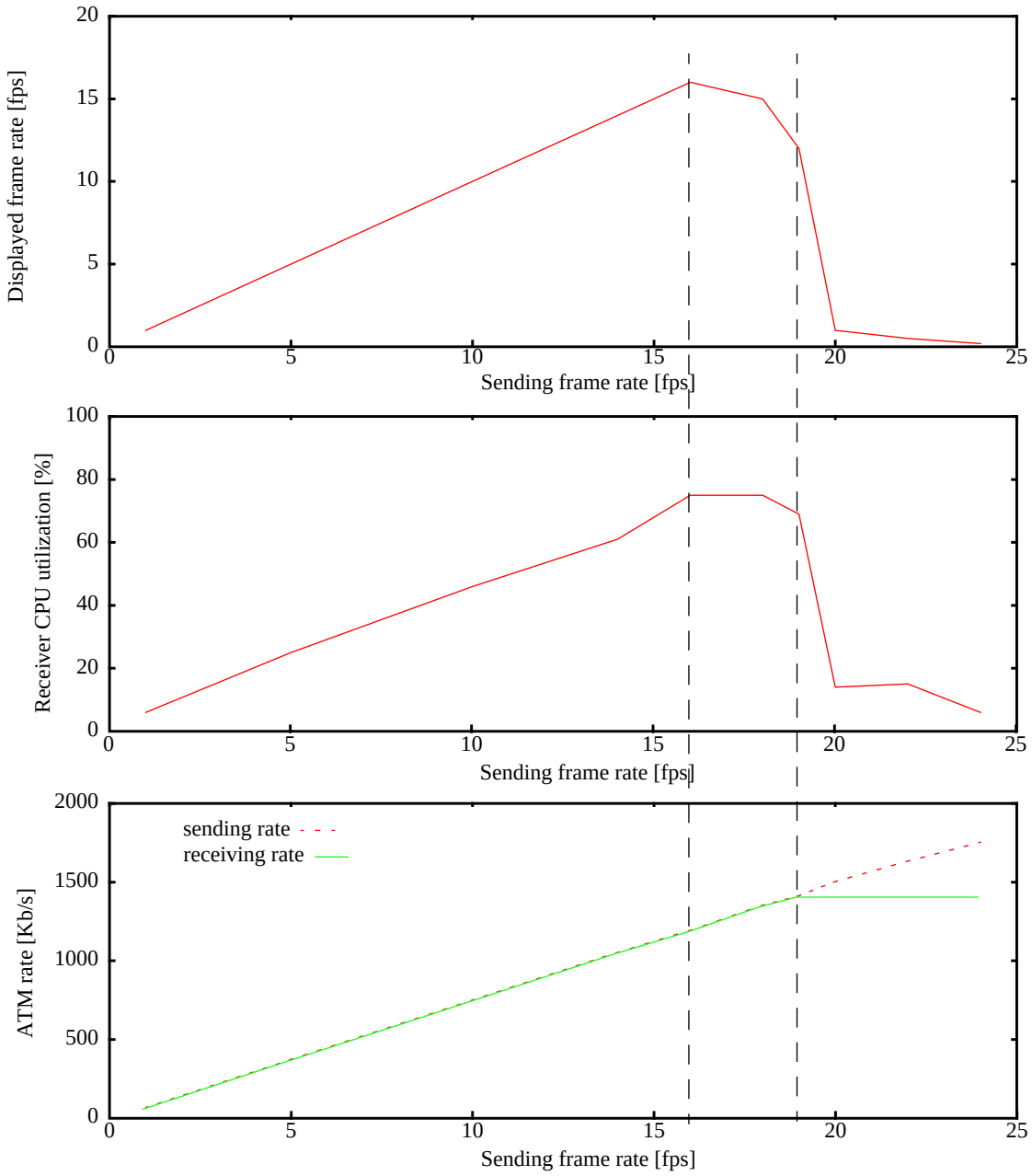


Figure A.2 Experimental Results with Host and Network Congestion.

In the second set of experiments, we used vic to send video from host A to host B through our local ATM network. As depicted by Figure A.3, even though we did not encounter any problem with the network, the CPU of the receiving host played a basic role in limiting the displayed frame rate. Beyond the CPU saturation point (at a frame rate of 18 fps), further increases of the sending frame rate barely altered the displayed frame rate on the receiving host.

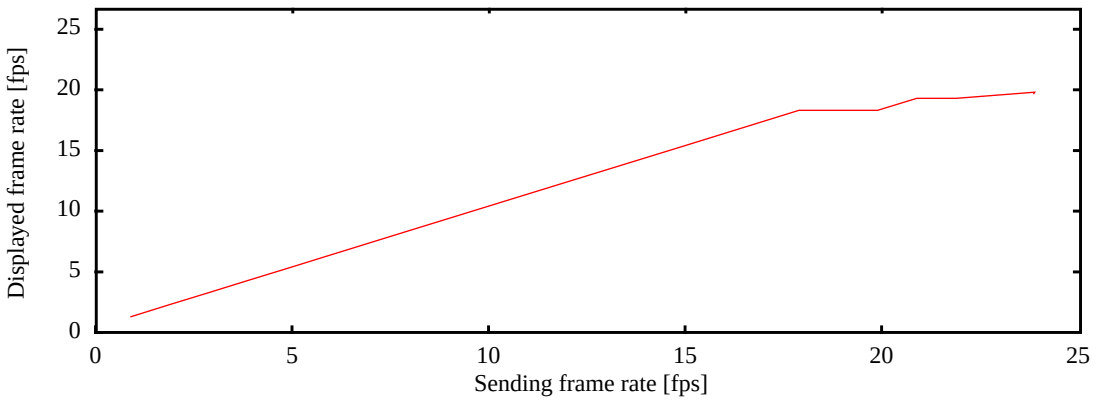


Figure A.3 Experimental Results with Host Congestion.

In our last experiment, we added a second sending host similarly equipped as host A. The results are shown in Figure A.4. In this case, the CPU on the receiving side became saturated at a lower frame rate (8 fps) compared to the previous experiment. Furthermore, beyond the saturation point, the two video sessions affected each other in a very unpredictable way, as clearly shown in Figure A.4. We did not explore any further the reciprocal influence of the two video sessions.

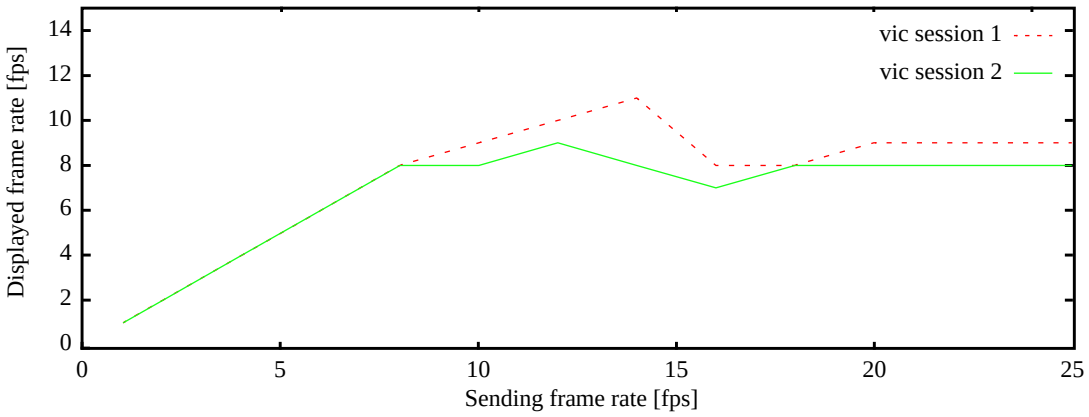


Figure A.4 Experimental Results with two Video Sources (Host Congestion).

References

[1] A. Albanese, J. Bloemer, J. Edmonds, M. Luby. Priority Encoding Transmission. *International Science Computer Institute TR-94-039*, Berkeley, California, August 1994

[2] M. Alfano, R. Sigle and R. Ulrich. Management of Cooperative Multimedia Sessions with QoS Requirements. *Proc. of IEEE Gigabit Networking Workshop GBN '96*, San Francisco, March 1996.

[3] A. Banerjea et al. The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences. *University of California at Berkeley and The International Computer Science Institute, TR-94-059*, November 1994.

[4] I. Barth, G. Dermler, W. Fiederer, K. Rothermel. A Framework for Negotiable Quality of Service in Distributed Multimedia Systems. Online Publication, *http://www.informatik.uni-stuttgart.de/ipvr/vs/vs_publikationen.html#1995-dermler-01*, October 1995

[5] A. Basso et al. Study of MPEG-2 Coding Performance based on a Perceptual Quality Metric. *Proc. PCS 96*, Melbourne, 1996.

[6] R. Braden et al. Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. *Internet-Draft, draft-ietf-rsvp-spec-10.ps*, February 1996.

[7] C.J. van den Branden Lambrecht, O. Verscheure. Perceptual Quality Measure using a Spatio-temporal Model of the Human Visual System. *Proc. SPIE International Symposium on Visual Communications and Image Processing '96*, Orlando, March 1996.

[8] I. Busse, B. Deffner, H. Schulzrinne. Dynamic QoS Control of Multimedia Applications based on RTP. Online Description, *http://www.fokus.gmd.de/step/acontrol/ac.html*, May 1995.

[9] P.P.S. Chen. The Entity-Relationship Model - Toward a unified view of data. *ACM Transactions on Database Systems 1*, March 1976

[10] Z. Chen, S. Tan, R. Campbell, Y. Li. Realtime Video and Audio in the World Wide Web. Online Descr., *http://choices.cs.uiuc.edu/HyperNews/get/video.research.forum/3.html*.

[11] D. Comer, Internetworking with TCP/IP, 2nd Edition, Volume I, Principles, Protocols, and Architecture, *PTR Prentice Hall*, 1991

[12] W.R. Daumer. Subjective Evaluation of Several Efficient Speech Coders. I*EEE Transactions on Communications, pp. 655-662,* April 1982.

[13] L. Delgrossi, R.G. Hertwich, C. Vogt and L.C. Wolf. Reservation Protocols for Internetworks: A Comparison of ST-II and RSVP. *4th International Workshop, NOSSDAV'93*, Lancaster U.K., November 1993.

[14] G. Dermler, W. Fiederer, I. Barth, K. Rothermel. A Negotiation and Resource Reservation Protocol (NPR) for Configurable Multimedia Applications. Online Publ., *http://www.informatik.uni-stuttgart.de/ipvr/vs/vs_publikationen.html#1995-dermler-01*, November 1995

[15] DeTeBerkom, BERKOM - Teleservice Multimedia Collaboration (MMC), Online Description, *http://www.deteberkom.de/projekte/texte/MMCoverview.html*

[16] D. Ferrari. Real-Time Communication in an Internetwork. *Journal of High Speed Networks, pp. 79-103*, 1992.

[17] S. Fischer, R. Keller. Quality of Service Mapping in Distributed Multimedia Systems. *IEEE International Conference on Multimedia Networking (MmNet95), pp. 132-141,* Aizu-Wakamatsu, Japan, September 1995.

[18] M. Handley and V. Jacobson. SDP: Session Description Protocol. *Internet-Draft, draft-ietf-mmusic-sdp-01.ps*, November 1995.

[19] Hughes Technologies, Mini SQL: A Lightweight Database Engine, Online Manual, Release 1.1, *http://Hughes.com.au/product/msql/manual.htm*, January 1996

[20] D. Hutchison, G. Coulson, A. Campbell and G.S. Blair, Quality of Service Management in Distributed Systems, *ACM SIGCOMM*, April 1994.

[21] ISO/IEC 9075: Information Technology --- Database Languages --- SQL, 1992

[22] ISO/IEC JTC 1/SC 21. Quality of Service - Basic Framework - *Working Draft #4, SC 21 Meeting*, Southampton, August 1994.

[23] V. Jacobson and S. McCanne. vat - LBNL Audio Tool. Online description, *http://www-nrg.ee.lbl.gov/vat*

[24] V. Jacobson and S. McCanne. vic - LBNL Video Tool. Online description, *http://www-nrg.ee.lbl.gov/vic*

[25] V. Jacobson and S. McCanne. wb - LBNL Whiteboard Tool. Online description, *http://www-nrg.ee.lbl.gov/wb*

[26] H. Kanakia, P. Mishra, and A. Reibman. An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport, *SIGCOMM Symposium on Communications Architectures and Protocols pp. 20-31*, San Francisco, September 1993

[27] B. Kernighan, D. Ritchie, The C Programming Language, 2nd Edition, *PTR Prentice Hall*, 1988

[28] A. Krishnamurthy, T.D.C. Little. Connection-Oriented Service Renegotiation for Scalable Video Delivery. *International Conference on Multimedia Computing and Systems, pp. 502-507*, May 1994.

[29] V. Kumar, MBone: Interactive Multimedia On The Internet, *Macmillan Publishing, Simon & Schuster*, November 1995

[30] A. Lazar, S. Bhonsle, K. Lim. A Binding Architecture for Multimedia Networks. *Journal of Parallel and Distributed Computing, Vol. 30, No. 2, pp. 204-216*, November 1995

[31] Lucent Technologies. Multimedia Communication Exchange Server (MMCX). Online description, *http://www.lucent.com/BusinessWorks/olc/product/mmcx.html*

[32] A. Maerz, An ATM Traffic Analyzer in JAVA, *Master's Thesis at the International Science Computer Institute*, Berkeley, California, May 1996

[33] S. McCanne and V. Jacobson. vic: A Flexible Framework for Packet Video. *Proc. of ACM Multimedia'95*, pp. 511-522, San Francisco, November 1995.

[34] Multimedia Applications on Intercontinental Highway (MAY). Online Description, *http://www.icsi.berkeley.edu/MAY/index.html*

[35] K. Nahrstedt and J.M. Smith, The QoS Broker. *IEEE Multimedia Vol. 2, No.1, pp. 53-67*, Spring 1995.

[36] R. Needham and A. Nakamura. Approach to Realtime Scheduling but is it Really a Problem for Multimedia? *3rd International Workshop*, La Jolla, California, November 1992.

[37] J. Ousterhout. Tcl and the Tk Toolkit. Addison-Wesley Publishing Company, 1994

[38] E. Schooler, Case Study: Multimedia Conference Control in a Packet-switched Teleconferencing System, *Journal of Internetworking: Research and Experience, Vol. 4, No. 2, pp. 99-120,* June 1993.

[39] L.C. Schreier and M.B. Davis, System-level Resource Management for Network-based Multimedia Applications, *Proc. NOSSDAV'95,* Durham, April 1995.

[40] H. Schulzrinne. Control Protocol for the Dynamic Configuration of Multimedia Conferencing Applications. *Internet-Draft, draft-ietf-mmusic-cpdcm-00.ps,* January 1996.

[41] H. Schulzrinne. Dynamic Configuration of Conferencing Applications using Pattern-Matching Multicast. *Proc. of NOSSDAV'95,* Durham, April 1995.

[42] H. Schulzrinne. Simple Conference Invitation Protocol. *Internet-Draft, draft-ietf-mmusic-scip-00.ps,* February 1996.

[43] H. Schulzrinne et al. RTP: A Transport Protocol for Real-time Applications. *IETF RFC 1889*. January 1996

[44] R. Steinmetz. Analyzing the Multimedia Operating System. *IEEE Multimedia, pp. 68-84,* Spring 1995.

[45] R. Steinmetz, K. Nahrstedt. Multimedia: Computing, Communications & Applications, Innovative Technology Series, Prentice-Hall 1995.

[46] W. Stevens, UNIX Network Programming, PTR Prentice Hall, 1990

[47] Sun Microsystems, Solaris Porting Guide, 2nd Edition, *SunSoft Press Books*, 1995

[48] W.C. Treurniet, L. Thibault. Perceval - A Model for Objective Perceptual Assessment of Audio. Online Publication, *http://www.crc.doc.ca:80/crc/branches/DRB/list.htm*l.

[49] T. Turletti and J.-C, Bolot. Issues with Multicast Video Distribution in Heterogeneous Packet Networks, *Proc. 6th International Workshop on Packet Video*, Portland, Oregon, 1994

[50] L.C. Wolf, L. Delgrossi, R. Steinmetz. Issues of Reserving Resources in Advance, *Proc . of NOSSDAV'95*, Durham, April 1995.