ELSEVIER

Computer Communications 21 (1998) 350–361

computer communications

Design and implementation of a cooperative multimedia environment with QoS control[1]

Marco Alfano*

Centre for Communication Studies 'Anghelos', Via Pirandello, 40-90144 Palermo, Italy

Abstract

The spread of distributed multimedia applications is setting forth a new set of challenges in the management of host and network resources for guaranteeing QUALITY AND SERVICE (QoS). When the multimedia applications compete for resources, as in the case of a cooperative multimedia environment, the management task becomes even more difficult.We have developed a cooperative multimedia environment (CME) that manages multimedia services and the underlying resources in an integrated way. Each user is provided with a single interface to invite other users to a cooperative session, select the media services to be used in the session, and specify his QoS requirements for the media services throughout the session. In this work, we describe the architectural details of the CME and its components with particular emphasis on the QoS mapping and QoS control mechanism. We also present the design and implementation details of an experimental prototype that provides video, audio and white board services. © 1998 Published by Elsevier Science B.V.

Keywords: QUALITY AND SERVICE; Multimedia; Application sharing

1. Introduction

Distributed multimedia applications are constantly growing in popularity thanks also to the presence of a widespread network like the Internet. Particular attention has been addressed towards real-time and interactive applications, e.g. videoconferences and shared applications, mainly because of the worldwide and decentralized features of today's research and development organizations.

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. In order for a cooperative multimedia environment to be widely used, it should utilize the same system resources (hosts and networks) that users have normally available (e.g. PCs, workstations, Internet). However, this entails that the same environment has to be shared by multimedia applications with strict requirements (e.g. real-time) and other applications that do not have comparably strict performance. Presently, there is no globally available mechanism for managing system resources that discriminates among applications privileging, for example, the real-time ones. Moreover, different policies are used to manage different resources and the management of the different resources is often not coordinated, particularly when the resources are distributed. The relative importance of different applications is seldom considered in a uniform way across the system resources.

Several dedicated applications (e.g. the MBone tools [1–3]) exist for transmitting video, audio and data between users. While on one hand these applications have the advantage of working in a widespread environment like the Internet, on the other hand they are usually highly demanding in terms of both network and host resources and may not get enough resources to work properly. Furthermore, if the user is using more applications at the same time there is no direct way for him to privilege an application over another.

In this paper, we present the architectural details and the experimental prototype of a cooperative multimedia environment (CME) that we have developed in order to help the user to set-up and control a cooperative multimedia session in an integrated way. The paper is organized as follows. Section 2 presents the architecture of the CME. Sections 3 and 4 respectively present the QoS mapping and QoS control mechanism. Section 5 presents an experimental prototype of the CME that provides video, audio and

* E-mail: Marco.Alfano@cselt.it

[1]A previous version of this paper has appeared in: M. Alfano, R. Sigle. Controlling QoS in a collaborative multimedia environment. Proc. of the Fifth IEEE International Symposium on High Performance Distributed Computing (HPDC-5) Aug. 7–9, 1996, Syracusae NY.

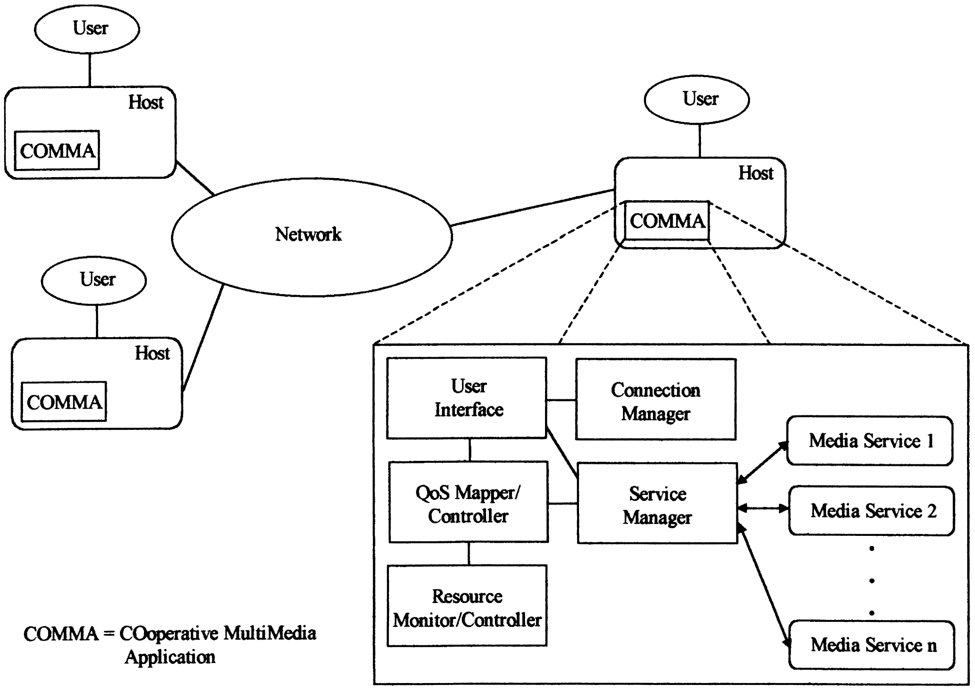


Fig. 1. The CME architecture.

white board services. Section 6 presents the results of a first set of experiments carried out on the experimental prototype. Finally, Section 7 presents some conclusions and a discussion on the future work.

2. The CME architecture

In order to create an integrated environment for cooperative work, we have developed a CME that realizes an efficient use of resources while providing the user with a single interface to easily start a cooperative session and control the QoS parameters of each media during the session. Our CME consists of cooperative multimedia applications, one for each user (Fig. 1).

A COoperative MultiMedia Application (COMMA) is made up of the following components.

- A media service for each media (e.g. video service). Each media service provides basic functions (e.g. sending, receiving and displaying video frames) and interacts with devices (or servers controlling devices) in its media category.
- A connection manager 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 cooperative session, the cooperative session does not rely on any centralized session moderator but is based on a distributed peer-to-peer model.
- A QoS mapper/controller that translates user QoS requirements into parameters for the media services and into QoS requirements for the underlying resources (i.e. host and network resources). It also executes the control mechanism in order to satisfy the user requirements on the media services.
- A resource monitor/controller of those host and network resources used to provide the different media services.
- A service manager for starting and stopping user-requested media services for the session. The service manager also monitors and changes the service parameters (e.g. video frame rate) following the indication of the QoS mapper/controller.
- A user interface that provides a graphical interface for starting or joining a cooperative session. Through this interface, a user can specify the media services he wants to use in the session and change his QoS requirements on the services.
- A user who wants to start a cooperative session specifies, through the graphical interface, the addresses of the users he wants to invite to the session and the media services to be used. The connection manager contacts the invited users who receive a message containing the name of the inviting person and decide whether to accept or refuse to join the session. When this set-up phase has been completed, the service managers at the different hosts start the provision of the chosen media services with some default values and the cooperative session takes place.

During the cooperative session, a user can change his QoS requirements on the media services. QoS requirements at user level are specified by means of simple attributes (e.g. low, medium and high quality video). These ‘high-level’ attributes are translated by the QoS mapper/controller into parameters for the media services and into QoS requirements for the underlying resources. Thus, depending on the user requests for the media services, the QoS mapper/controller decides the performance parameters for the services (e.g. sending video at 10 fps) and evaluates through the resource monitor/controller whether these parameters can be supported by the underlying resources. Finally, the QoS mapper/controller makes the necessary adjustments so that the media services can perform as planned.

In Sections 3 and 4 we describe the QoS mapping and QoS control mechanism in detail.

2.1. Integration aspects of the CME architecture

One of the main goals of the CME architecture is to realize an integrated environment while keeping the advantages of using a distributed heterogeneous environment. Thus, the CME architecture presents a horizontal integration that operates exclusively within a certain layer and a vertical integration that spans the different layers. Horizontal integration embraces all sites of a cooperative environment, whereas vertical integration only operates within one site. Fig. 2 illustrates the integration aspects addressed by the CME architecture.

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 resource requirements.

Horizontal integration is a result of the distributed structure of the CME architecture and, as said above, operates exclusively within a certain layer. User layer integration is achieved by offering a user the possibility to specify quality requests for the employed media services. As will be shown in the following sections, our prototypical implementation, for example, includes a control panel that allows users to specify their quality requirements for the media services in a uniform way. Integration at the media service layer is achieved by embedding the media services into the CME architecture. This specifically means that applications are not handled independently any more. Integration at the resource layer is achieved by providing mechanisms for the orchestration between host and network resources and their management structure. The CME architecture accomplishes horizontal integration at the resource layer by taking the different resource parameters into account. Monitor and control mechanisms keep track of resource status and availability and prevent resource saturation. The knowledge at the same time of resources availability and user requirements allows the CME to assign resources to the media services more accurately and efficiently.

2.2. Related work

The need for integration within a specific layer (user, application, resource) and between layers has been addressed by several research groups [3–8]. 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 examples of such integration efforts.

The MBone tool developers have introduced several media service synchronization mechanisms to address the problem of horizontal integration [3]. Cross-media synchronization is carried out over a conference bus. The conference bus abstraction provides a mechanism which coordinates the separate media service processes. In Ref. [8], a local control architecture ties 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 and is mainly employed to establish horizontal integration at the application layer. User interface integration can be found in the multimedia communication exchange server (MMCX) [5] where team members 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 media services. The QoS Broker approach [6] addresses the relationship between 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 for media specific and transport level QoS handling is introduced in Ref. [4]. A negotiation and resource reservation protocol (NPR) for multimedia applications allows QoS negotiation and resource reservation. As an application level protocol, it offers transparency from the underlying transport layer structure. In Refs. [9,10] two different approaches for guaranteed resource reservation at the network level are presented. In Ref. [11], resource reservation is also discussed, but from the point of view of host resources.

3. QoS mapping

Since this architecture is oriented towards the end user, he must be able to express his QoS requirements for the media services in a simple way (e.g. low, medium, or high quality video). These requirements, as shown in Fig. 2, will then be translated on one hand into parameters for the media services (e.g. frame rate for video) and, on the other hand, in QoS requirements for the

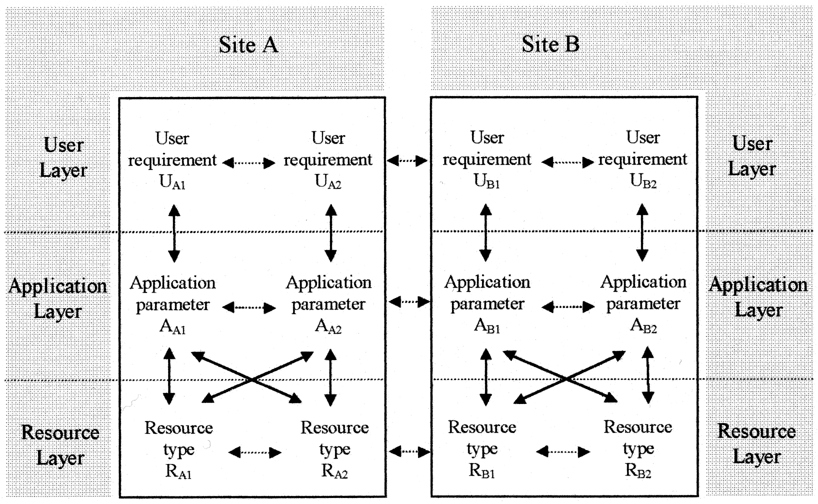

Site A
Site B
User Layer
Application Layer
Resource Layer
User requirement U_{A1}
User requirement U_{A2}
User requirement U_{B1}
User requirement U_{B2}
Application parameter A_{A1}
Application parameter A_{A2}
Application parameter A_{B1}
Application parameter A_{B2}
Resource type R_{A1}
Resource type R_{A2}
Resource type R_{B1}
Resource type R_{B2}

Fig. 2. Horizontal and vertical integration of the CME architecture.

underlying resources. As said above, the architecture component that provides the QoS translation at the different levels is the QoS mapper/controller.

Let us now examine how QoS requirements can be expressed at different levels. At the user level, QoS requirements can be grouped into two categories:

- direct requirements, when the user explicitly specifies a requirement for a media service, e.g. he asks for high quality video;
- indirect requirements, when the user makes some actions, such as iconifying a video window, that indirectly makes suggestions about the user interest on a particular media service (in the case of iconifying a video window, the user shows no interest for that video). While the user actions that lead to indirect requirements can be detected by examining the user environment and the way he interacts with it (e.g. iconifying or deiconifying a window and putting a window in background), more effort is required to define direct requirements. A direct requirement should be, at the same time, easy to understand for the user, general enough to include more particular requirements (so that the user is not required to specify too many requirements), and tractable so that it can be translated in QoS requirements for the underlying resources.

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 the satisfaction of the user for the media service quality can be expressed in quantitative terms. In what follows, we present the first results of an analysis on user requirements that we have been carrying out.

In order for the user not to deal with too many parameters, we define only one global requirement for each media and we indicate it with the generic term of quality. Thus, we will have video quality, audio quality and so forth. The quality requirement is a repository of more specific requirements on a media service. For example, video quality is intended in a broader sense than just considering how good received video pictures are compared to the original ones. This is, of course, part of video quality and is related to spatial vision but there is also temporal vision that must be taken into account, i.e. how the user perceives scene changes in the received video compared to the original one [12].

Many studies in the literature dealing with quality estimation of digitally coded video sequences [12,13] and audio sequences [14,15] use a five-level scale, reported in Table 1, for quality rating. This scale is also used for subjective testing in the engineering community [16].

We use the same 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. In the case of video, this scale is used to assess quality for both spatial and temporal perception [12]. In practice, the user will use a slider for each media service to indicate his quality requirements from a minimum value (quality level 1) to a maximum value (quality level 5).

Once the way to express user requirements has been defined, the next problem is to find a mapping between user requirements (quality levels) and parameters of the media services. The question is, what is the performance a media service must have in order to provide a certain quality level? We need some mapping functions that connect, for example, video quality to video frame rate. These functions are similar to the benefit functions found in Ref. [17] and require the

Table 1
Quality rating on a one to five scale

Rating	Impairment	Quality
5	Imperceptible	Excellent
4	Perceptible, not annoying	Good
3	Slightly annoying	Fair
2	Annoying	Poor
1	Very annoying	Bad

Table 2
Video quality rating for JPEG video

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

execution of subjective tests in order to determine whether for a given performance the user perceives the quality level of the media service as bad, poor, fair, good, or excellent.

We executed some tests with a group of ten people and asked them to rate the quality of video and audio sequences according to the one to five scale discussed above. As a video application, we used vic [3] and as an audio application, we used vat [1]. In the first test, we showed a video sequence at different frame rates in order to evaluate the temporal quality of video. First, we showed the video sequence at 30 fps. We told the viewers to consider the quality of that video sequence to be five and to rate the quality of the next video sequences against that one. In the second test, we showed a video sequence with different resolutions in order to evaluate the spatial quality of video. First, we showed the video sequence with the best resolution the video application could provide. Again, we told the viewers to consider the quality of that video sequence to be five and to rate the quality of the next video sequences against that one. We then varied the resolution as a percentage of the best provided resolution. As a final test, we transmitted an audio stream with different encoding schemes in order to evaluate the audio quality. The PCM encoding scheme was considered to provide quality five.

We averaged the obtained results and built the tables that map the quality levels to the service parameters. The results of this mapping are reported in Table 2 for video and in Table 3 for audio.

The third and final step in QoS mapping is to translate the media service parameters in QoS requirements for the host and network resources. Different resource parameters can be connected with the performance of a media service. For simplicity, we consider QoS requirements for the following resource parameters:

- Network resources {bandwidth (Kb s^{-1})};
- Host resources {CPU type, CPU load (%)}.

Table 3
Audio quality rating

Quality	Encoding scheme
5	PCM
5	PCM2
5	PCM4
4	DVI
4	DVI2
4	DVI4
3	GSM
2	LPC4

Table 4
Mapping of video quality to resources for JPEG video

Quality	Degree of movement	Frame rate (fps)	Resolution (%)	Bandwidth (Kb s^{-1})	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

For video, it is very difficult to correlate media service performance and requirements on resources. Network bandwidth and mainly CPU utilization are very influenced by the frame size (assuming the user has the possibility to change the video size), compression scheme, and degree of movement (slow or rapid scene changes).

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. The bandwidth was provided by the vic and vat applications while the CPU usage was measured with the Unix top utility. The results of these measurements are reported in Table 4 for video and in Table 5 for audio. For video, we estimated the necessary resources for receiving JPEG video (320×240). For each quality level, we considered three possible degrees of movement, i.e. still, slow motion, and high motion. As a receiving host, we used a Sun™ sparc5.

4. The QoS control mechanism

The user who starts the cooperative session chooses the

Table 5
Mapping of audio quality to resources

Quality	Encoding scheme	Bandwidth (Kb s^{-1})	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

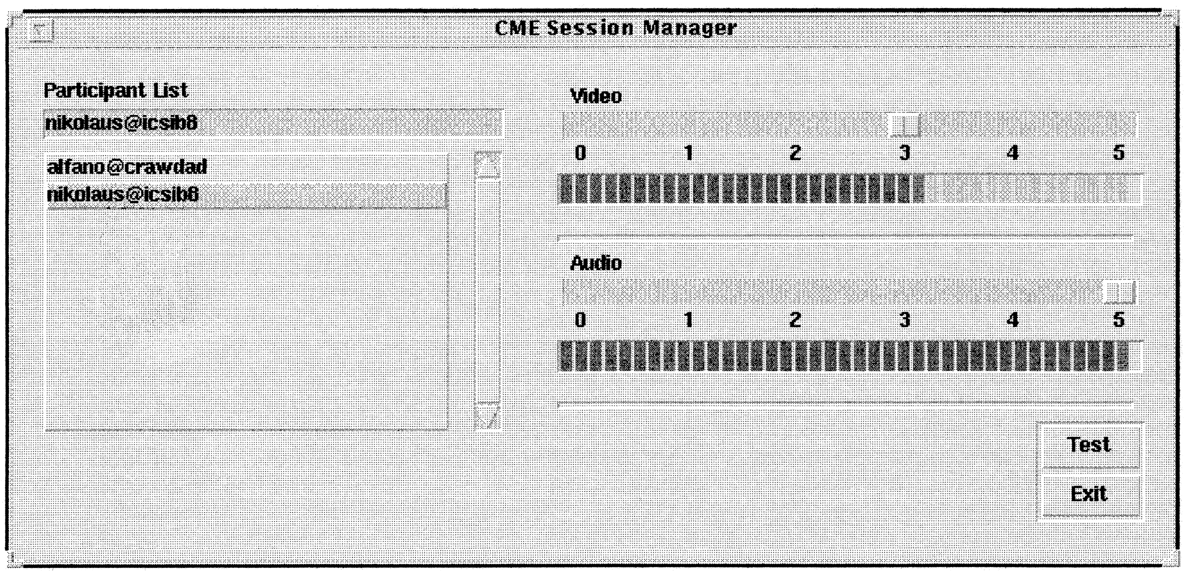


Fig. 3. User interface for managing a cooperative session.

media services to be used for that session. The media services are started with some default values and each participant is presented with a graphical interface that contains information on the media services used in the session (Fig. 3). For each media service there is a meter that goes from zero to five and indicates the quality of that service. Level zero indicates that the service is not being received.

The service manager monitors the media services and reports the values of the service parameters to the QoS mapper/controller, which in turn translates these values to quality levels and passes them to the graphical interface for displaying. If a quality level is connected with more than one service parameter, the QoS mapper/controller uses the following expression to compute a single value for the service quality (Service_Q):

$$\text{Service\_Q} = \alpha_1 Q_1 + \alpha_2 Q_2 + \ldots + \alpha_n Q_n$$

where Q_i is the quality level obtained for parameter P_i; α_i is a weight indicating the relative importance of P_i for Service_Q and is related to the subjective interest of the user towards a parameter rather than another; $\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1$.

For a video service, for example, we will have:

$$\text{Video\_Q} = \alpha_1 \text{ frame\_rate} + \alpha_2 \text{ resolution}$$

Normally, a user will not use all the media services involved in a cooperative session at the same time. For example, at the beginning of a cooperative session, a user usually wants to see and talk to the other participants in order to exchange the basic ideas on the common work and decide how to proceed with it. Once started working on a common text/picture, the interests of the users are mainly directed to the shared application. Users are not too interested in seeing each other any more, but they do want to keep talking to each other. Thus, user interests on the different media services are likely to change over time during a cooperative session.

The graphical interface provides an easy way to express user requirements by means of a slider that allows specification of the quality level for each media service. The slider moves in a discrete way and the user may specify any of the five levels that correspond to the five quality levels discussed above. In addition, specifying zero, the user indicates that he does not want to receive that service. An indirect requirement, as iconifying a video window, will have the same effect as to move the slider to zero.

The indication of the slider is two-fold:

- it indicates the quality the user wishes to perceive for that media service;
- it indicates the interest of the user for that service compared to the other services by assigning a priority to the service. The priority corresponds to the quality level with five being the highest priority and zero being the lowest.

  If a pricing policy is applied to the used resources, the quality requirements (properly converted) will indicate how much a user is willing to pay for the different media services. Thus, in this case, the slider mechanism allows the user to save money for a particular media service when the quality of that service does not have a particular relevance.

The control mechanism, in trying to satisfy the user requirements, will establish a priority list of services based on the assigned priorities and will privilege more those services with higher priorities. The control mechanism will be activated when the quality level chosen by the user differs from the actual value supplied by the system beyond a threshold for a given time interval (to avoid having continuous control activity). This may happen because the user changes his requirements on a media service either directly through the slider or indirectly, e.g. iconifying a video window. Moreover, the status of the resources may not allow a media service to perform in a way that is even close to the quality level requested by the user. The control mechanism will try either to change the media-service parameters or to reassign the resources so to satisfy the user requirements. To this end, it monitors the status of the host and network resources through the resource monitor/controller.

We can consider different scenarios depending on to what extent the resource monitor/controller can control resources. If it is able to reserve both network and host resources, the QoS mapper/controller will recompute the service priorities and, based on these priorities, will reassign resources to services trying to guarantee the quality of the services with higher priorities. A user requirement then becomes an objective for the whole system that should properly act in order to satisfy this requirement [18].

If a CME has to make use of a world-wide network like the Internet, it must be considered that, presently, the Internet does not provide any globally available mechanism which discriminates among applications and manages the network as a whole. A more realistic scenario then assumes that the resource monitor/controller can partially control resources, i.e. it can control host resources but not network resources. Hosts usually belong either to users or to institutions where users work. This means that users (or system administrators who act on users' behalf) can directly access host resources. On the other hand, users and system administrators cannot control the resources of the networks through which the data of the cooperative session flow, unless all the participants are in the same local network. Thus, the particular assumption that users can only control their host resources seems quite reasonable.

In this scenario, when the QoS mapper/controller observes a disequilibrium between the request of the user and the achieved performance of a media service, it will try to understand whether the system resources are not sufficient or the problem is caused by a wrong setting of the service parameters. Let us consider, for example, the case where the service meter indicates that a media service is performing with a lower quality level than the one requested by the user. The control mechanism first checks whether the reason of poor performance of a media service is related to the underlying resources that, for example, get saturated. In this case, the control mechanism determines whether the problem is either in the network or in the host. If the problem is the host CPU, the control mechanism reassigns the CPU resources in order to increase the performance of that service while, at the same time, respecting the priorities of the services. If the problem lies in the network or the CPU is not powerful enough to support the service requirements, the QoS mapper/controller asks its peer on the sending site to lower the parameters of the media service. In fact, as shown in Ref. [19], sometimes decreasing the values of the service parameters on the sending site may entail a better performance of the service on the receiving site and consequently a higher quality for the user. For example, if JPEG video is sent through a network and this network becomes congested, as soon as a packet is dropped in the network a whole frame will be discarded on the receiving side. This may entail a remarkable difference between the sending and received frame rate. In this case, an increase in the sending frame rate will probably cause the network to become even more congested, whereas a decrease in the sending frame rate will more likely lead to an increase in the received frame rate and, in turn, a better quality (unless the network is already congested by other applications, but even in this case the received frame rate practically does not change).

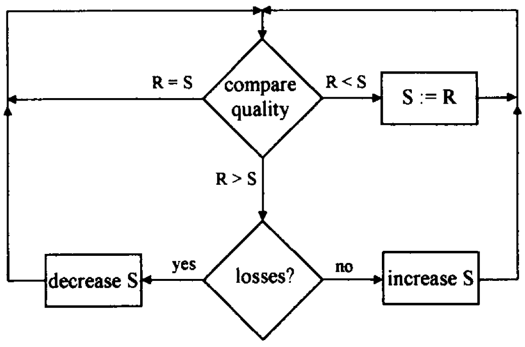


Fig. 4. Global control mechanism.

Whenever the QoS mapper/controller of a sending site receives a request from its peer on a receiving site for either increasing or decreasing the value of the parameters of a media service, it will behave differently depending on whether the cooperative session is between two users or more. In the former case, the sending site will immediately act upon the request of the receiving site. In the latter case, the sender will take into account the requests of all sites and will make an average of them. It then will act considering the averaged value.

If the user requires either directly or indirectly quality zero for a media service, the QoS mapper/controller will ask the service manager to stop temporarily receiving that media service and will inform its peer on the sending site that it is not interested in receiving that service. As soon as the QoS mapper/controller of a sending site realizes that nobody is interested in receiving a service, it will stop temporarily transmitting the related media stream, thus avoiding a useless waste of resources.

For implementation reasons, we have split the control mechanism into two parts. A global mechanism (Fig. 4) works on the sending site and checks for potential differences between the sending quality S and the requested quality R computed with the following expression:

$$R = \frac{1}{n}\sum_{i}^{n} R_i$$

where R_i is the requested quality for user U_i ($i = 1, \ldots, n$).

If the global control mechanism senses a disequilibrium,

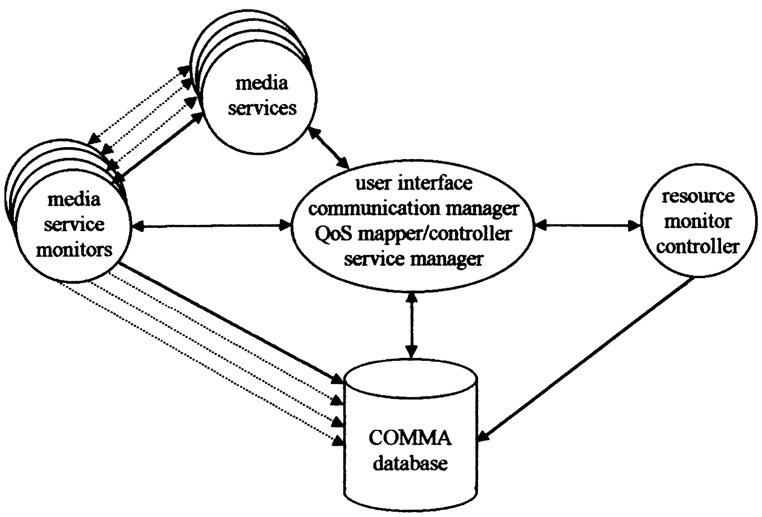


Fig. 5. Process oriented view of a COMMA.

it will verify whether there are losses in the host or in the network and will behave accordingly. On the receiving host, a local control mechanism checks if there are losses locally and in this case it will try to redistribute the host resources taking into account the priorities of the media services.

5. The COMMA experimental prototype

In order to evaluate the architectural framework of the CME presented in the previous sections, we have implemented an experimental prototype. The COMMA prototype accomplishes the main architectural goals. It performs the following functions:

- allows a user to specify QoS requirements for the media services;
- adjusts the media services performance dynamically depending on the user requirements and resource status;
- monitors and controls the resources.

The prototype has been implemented by using the Sun Solaris™ operating system. The programming environment comprises the ANSI-C [20] and TCL/TK [21] programming languages. For storage of persistent data and for interprocess communication within one system, the relational database MiniSQL [22] has been employed. Finally, for interprocess communication between processes on different systems, the Berkeley socket paradigm [23] has been used.

In order to work in a generic environment, the experimental prototype considers a resource scenario where neither the network nor the host offer any QoS guarantees, i.e. hosts with the Unix operating system and the Internet as the communication network. However, the flexibility of the CME architecture allows one to extend the prototype in order to include different resource scenarios.

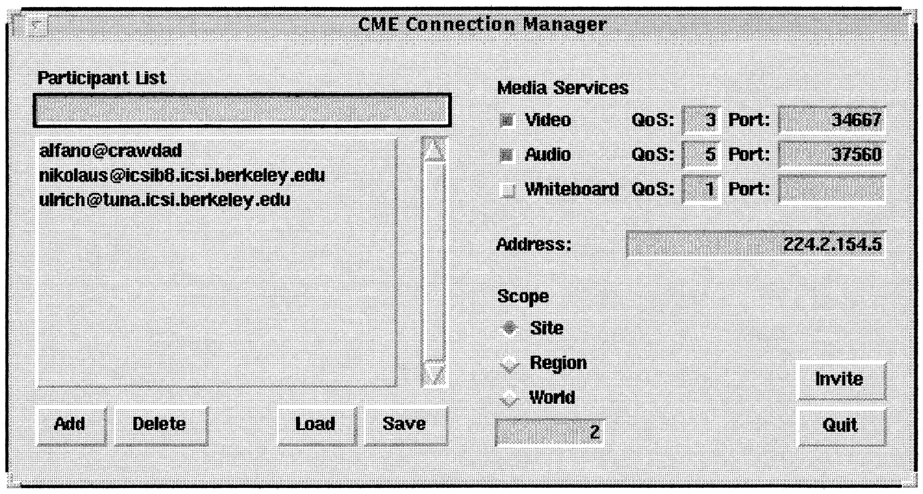
The host and network resource properties can be described as follows. Processes residing on a participant's host offer a time sharing capability. This specific property allows for changing process priorities but does not offer any absolute QoS guarantees. The network resources in turn do not offer any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

As already discussed in Section 2, each session participant runs a COMMA. In the prototype, a COMMA is made up of a set of processes as depicted in Fig. 5, namely a resource monitor/controller, the COMMA database, a set of media services, a set of adjacent media service monitors and a process that contains the connection manager, the QoS mapper/controller, the service manager and the user interface.

We now briefly describe each COMMA component. A complete description of the COMMA prototype can be found in Ref. [19].

5.1. The COMMA database

All COMMA components exchange their data through the COMMA database which consists of a set of tables that are mainly employed to store monitored information and to register the invoked media services and the corresponding monitors. The COMMA prototype uses Mini SQL [22] 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 specification [24]. The most important property of mSQL with regard to the COMMA 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.



Participant List

alfano@crawdad
nikolaus@icsib8.icsi.berkeley.edu
ulrich@tuna.icsi.berkeley.edu

Media Services

▣ Video	QoS:	3	Port:	34667
▣ Audio	QoS:	5	Port:	37560
☐ Whiteboard	QoS:	1	Port:	

Address: 224.2.154.5

Scope

● Site
○ Region
○ World

2

Fig. 6. User interface for initiating a cooperative session.

5.2. Media services — MBone tools

The COMMA prototype uses, as media services, the MBone tools developed at the UC Berkeley and Lawrence Berkeley National Laboratory, i.e. the video tool vic [3] for video, the audio tool vat [1] for audio and the white board tool wb [2] as a white board.

The vic and vat applications are based on the draft Internet standard real-time transport protocol (RTP) [25] developed by the IETF Audio/Video Transport working group. RTP is an application-level protocol implemented entirely within the application.

5.3. 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 COMMA 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. The MBone tools include two properties that allow us to monitor them without modifying their 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 and bandwidth usage) that has been computed by the media services by means of the sent and received RTP packets. The retrieved data is finally written into the corresponding tables of the COMMA database.

5.4. User interface

The COMMA user interface is mainly split into two parts. The first part provides a graphical interface for the connection management (Fig. 6). It is employed by the session initiator who creates a session specific invitation message where he specifies the list of the invited users and the media services to be used for the session along with their initial QoS values.

The second part provides a graphical user interface for the session management (Fig. 3). It is employed by all session participants to control the QoS of the various media services. The slider of a media service indicates, for a selected participant, the service quality the user wants to receive from that participant. If the user himself is selected in the participant list, the slider indicates the averaged quality requirements of the other session participants. 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 Section 3.

5.5. Connection manager

The COMMA prototype employs Unix sockets [23] in order to provide connection management functionalities. 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

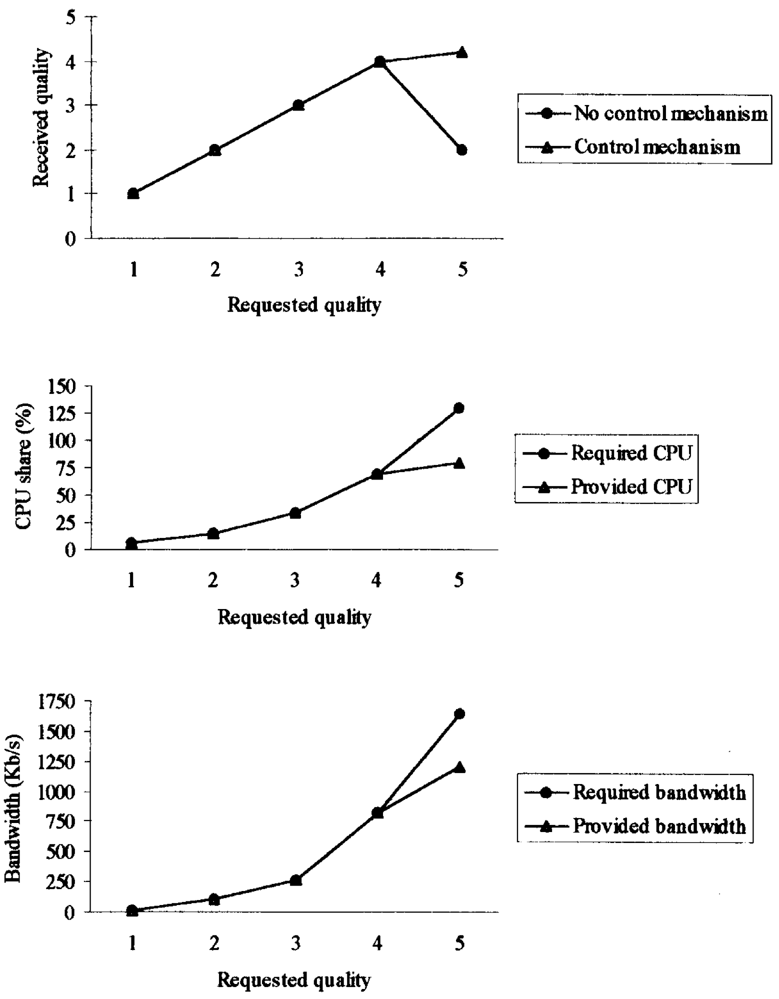


Fig. 7. Results of a set of experiments carried out on the COMMA prototype.

paradigm, this specifically means that a connection manager may act at the same time as a client and as a server.

5.6. Service manager

The COMMA service manager provides functionality for the other COMMA components, mainly for the QoS mapper/controller, in order to start and stop media services, and to set and get media service parameters.

5.7. 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 have already been discussed in the previous two sections since the mapping and control aspects represent a crucial part of the CME architecture.

5.8. Resource monitor/controller

In Section 2 we outlined the tasks of the resource monitor/controller. They mainly comprise monitoring and controlling host and network resources. In the experimental prototype, however, the resource monitor/controller only monitors the CPU load of each media service and the idle CPU by employing the iostat BSD Unix tool. The consumption of network resources has not to be monitored since this task is already performed by the media service monitors. In the prototype, media service processes run under the time-sharing class. By employing the priocntl/priocntl_set library functions, the resource monitor/controller dynamically assigns process priorities to active media services. Although this mechanism allows to privilege certain processes, it does not offer QoS guarantees in absolute terms.

6. First experiments with the COMMA prototype

We now present the results of a first set of experiments that we executed to test the control mechanism of the COMMA prototype. The main focus was on the video service because it is the media that entails the higher resource consumption. We considered an environment for world-wide collaboration with limited but guaranteed network resources. We used the MAY (Multimedia Applications on Intercontinental Highway) network, an ATM network (with the IP protocol implemented on top of ATM) that connects North America to Europe [26]. We executed the experiments at the International Computer Science Institute in Berkeley (where the COMMA prototype was developed) and set-up a loopback in Germany thus obtaining a world-wide ATM link with a fixed transmission rate of 1.5 Mb s^{-1}.

As a sending host, we used a Sun™ sparc20 workstation equipped with a Parallax™ video board. This video board supports JPEG compression in hardware. Thus, we could vary the video frame rate in a wide range without consuming a lot of CPU resources. On the receiving side, we used a Sun™ sparc5 workstation with a Sun Video™ board. This video adapter does not provide the same hardware support as the Parallax board, therefore the decompression has to be done in software and this requires a lot of CPU resources.

We transmitted video (with a slow-motion degree of movement) by means of the video service (vic) of the COMMA prototype. While changing the requested quality, we observed the received quality, the CPU consumption on the receiving host and the transmission rate of the ATM network. The CPU consumption was provided by the monitoring part of the COMMA prototype and the transmission rate of the ATM network was measured by means of the management tools of the local Synoptics™ ATM switch.

By varying the requested quality from one to five, we obtained the results shown in Fig. 7. The first graph shows the received quality vs. the requested quality. The second graph shows the required CPU share for the different quality requests and the actual provided share. The third graph shows the required network bandwidth for the different quality requests and the actual provided bandwidth. On computing the received video quality, we considered slightly more important the temporal quality. Thus, we used the following expression:

$$\text{Video\_Q} = 0.6\ \text{frame\_rate} + 0.4\ \text{resolution}$$

Up to a requested quality of four, the resources were sufficient to supply the requested quality level. For a requested quality of five, both the CPU of the receiving host and the network became saturated. Thus, without any control mechanism, the receiver experienced a dramatic reduction in the received frame rate (0.5 fps) and, hence, a received quality level of two (the resolution did not have the same reduction as the frame rate). By applying the control mechanism, we did not experience the big decrease in the received frame rate because the control mechanism set the sending frame rate and resolution to the proper values (16 fps, 55%) so that the resource consumption stayed on the borderline of the saturation zone.

Note that the curve related to the control mechanism is averaged. The received quality kept oscillating around the average value because the control mechanism kept decreasing and increasing the sending frame rate and resolution depending on whether it was sensing losses or not. Although we did not investigate this issue any further, a more detailed analysis should be carried out in order to examine the potential instabilities of the control mechanism.

These experiments only consider one of all the possible scenarios. They mainly show how the control mechanism that has been implemented in the COMMA prototype works for a particular case. Of course, the simple control mechanism presented here can be improved and a more complete set of experiments could give suggestions on the way to proceed. Nevertheless, even in the case the control mechanism is changed, the structure of the CME and its components would mainly remain the same thanks also to the CME modularity.

7. Conclusions and future work

In this work, we presented the architectural details of a CME that we have developed in order to help the user to set-up and control a cooperative multimedia session. In particular the QoS mapping and the QoS control mechanism have been discussed. We also presented an experimental prototype of a COMMA that provides video, audio and white board services. Finally, we presented the results of a first set of experiments carried out on the experimental prototype.

The work presented in this paper, to our best knowledge, is one of the first attempts in creating an integrated architecture for QoS control of a cooperative multimedia environment that spans from the user level down to the resource level. There are still different open issues that require further investigation. Among them, a better understanding of user requirements is necessary in order to evaluate whether the generic user is comfortable with the quality levels introduced here. In particular, it is important to understand whether a user should have the possibility to express more than one requirement for a media service, e.g. for video he could express his requirements for temporal quality (frame rate) and spatial quality (picture resolution) separately. The user requirements should also take into account the possible relationship between different media as in the case of audio/video synchronization. More work is required in mapping user requirements into media-service parameters and system resources. Other service and resource parameters should be taken into account beside the ones already considered here and their influence on the media-service quality should be evaluated. More work also needs to be done for the control mechanism. In particular other scenarios should be considered beside the one that assumes that a user can control host resources but not network resources. We plan to investigate how to control the different resources in an integrated way in order to guarantee that a user obtains the service quality he is requesting.

Acknowledgements

This work has mainly been developed during my stay at the International Computer Science Institute in Berkeley. Thanks go to Nikolaos Radouniklis and Rolf Sigle who respectively helped in the implementation of the COMMA prototype and in the execution of the experiments on the prototype.

References

[1] V. Jacobson and S. McCanne, vat – LBNL Audio Conferencing Tool On line description, http://www-nrg.ee.lbl.gov/vat/.
[2] V. Jacobson and S. McCanne, wb – LBNL Whiteboard Tool Online description, http://www-nrg.ee.lbl.gov/wb.
[3] S. McCanne, and V. Jacobson, vic: A flexible framework for packet video Proc. of ACM Multimedia'95, San Francisco, CA, November 1995, pp. 511–522.
[4] G. Dermler, et al., A negotiation and resource reservation protocol (NPR) for configurable multimedia applications, Technical Report 11/95, Fakultät Informatik, University of Stuttgart, December 1995.
[5] Lucent Technologies Multimedia Communication Exchange Server (MMCX) Online description, http://www.lucent.com/Business-Works/olc/product/mmcx.html.
[6] K. Nahrstedt and J.M. Smith, The QoS Broker, IEEE Multimedia 2 (1) (1995) 53–67.
[7] E. Schooler, Case Study: Multimedia Conference Control in a Packet-switched Teleconferencing System, Journal of Internetworking: Research and Experience 4 (2) (1993) 99–120.
[8] H. Schulzrinne, H. Dynamic configuration of conferencing applications using pattern-matching multicast, Proc. of NOSSDAV'95, Durham, April 1995.
[9] D. Ferrari et al. Network support for multimedia — a discussion of the Tenet approach, Computer Networks and ISDN Systems 26 (1994) 1267–1280.
[10] L. Zhang et al., RSVP: A new ReSerVation Protocol, IEEE Network 7 (1993) 8–18.
[11] D.P. Anderson, Metascheduling for continuous media, ACM Transactions on Computer Systems 11 (1993) 226–252.
[12] C.J. Van den Branden Lambrecht and O. Verscheure, Perceptual quality measure using a spatio-temporal model of the human visual system, Proc. SPIE Int.l Symp. on Visual Communications and Image Processing '96, Orlando, FL, March 1996.
[13] A. Basso, et al., Study of MPEG-2 coding performance based on a perceptual quality metric, Proc. PCS 96, Melbourne, 1996.
[14] W.R. Daumer, Subjective evaluation of several efficient speech coders, IEEE Trans. on Communications, (1982) 655–662.
[15] W.C.Treurniet and L. Thibault, Perceval — A model for objective perceptual assessment of audio On line publication, http://www.crc.doc.ca:80/crc/branches/DRB/list.html.
[16] ITU-R Recom. BT.500.7. Methodology for the subjective assessment of the quality of television pictures.
[17] L.C. Schreier, and M.B. Davis, System-level resource management for network-based multimedia applications, Proc. of NOSSDAV'95, Durham, April 1995.
[18] M. Alfano, Scheduling features in distributed systems, Proc. of the SBT/IEEE International Telecommunications Symposium, Rio de Janeiro, August 1994, pp. 52–56.
[19] M. Alfano and R.A. Radouniklis, cooperative environment with QoS control: Architectural and implementation issues, ICSI Technical Report TR-96-040, September 1996.
[20] B. Kernighan, and D. Ritchie, The C Programming Language, 2nd edn., PTR Prentice Hall, Englewood Cliffs, NJ, 1988.
[21] J.K., Ousterhout, Tcl and the Tk Toolkit, Addison–Wesley, Reading, MA, 1994.
[22] Hughes Technologies, Mini SQL: A Lightweight Database Engine, Release 1.1, Online Manual, http://Hughes.com.au/product/msql/manual.htm, January 1996.
[23] D. Comer, Internetworking with TCP/IP, Vol. I, Principles, Protocols, and Architecture, PTR Prentice Hall, Englewood Cliffs, NJ, 1991.
[24] ISO/IEC 9075: Information Technology — Database Languages — SQL, 1992.
[25] H. Schulzrinne et al., RTP: A transport protocol for real-time applications, IETF RFC 1889, January 1996.
[26] Multimedia Applications on Intercontinental Highway (MAY) On line description, http://www.icsi.berkeley.edu/MAY/index.html

Marco Alfano received his Laurea degree in Electronic Engineering from the University of Palermo, Italy, in 1988 and in the following two years he was a grant holder of the Italian National Council of Research (CNR) to lead research on computer networks. From 1992 to 1994 he worked at the Universities of Palermo and Catania, Italy, and at the University of Waterloo, Canada for a Ph.D in electronics, computer science, and telecommunications engineering. The Ph.D work mainly dealt with the design of a scheduler for distributed applications with user requirements. From 1995 to 1996, he was at the International Computer Science Institute in Berkeley, USA, where he worked on QoS for distributed multimedia applications and management of high-speed networks. He is currently a senior researcher at CSELT in Torino and the Director of the centre for Communication Studies ''Anghelos'' in Palermo. His research interests include communication systems, distributed computing and QoS for multimedia applications.