Application Design for Cooperative QoS Management[1]

Stefan Fischer, Mohamed-Vall O. M. Salem and Gregor von Bochmann
University of Montréal, Département IRO
C.P. 6128, Succ. Centre-Ville, Montréal (PQ) H3C 3J7, CANADA

1. This work was supported by the CITR Major Project "Broadband Services".

Abstract

QoS management for distributed multimedia applications becomes more complex when a huge number of users are participating, as for instance in broadcasts of major sports events or in teleteaching applications. On the other hand, such an application offers a variety of options to improve resource usage and system performance while decreasing the overall communication cost. We developed a new QoS management scheme called Cooperative QoS management which handles both increased complexity and options. In this paper, we show how this new scheme influences the design of applications based on it, especially concerning the QoS user interface. As an example, we present a teleteaching application developed in the framework of our project "Broadband Services".

Keywords*: QoS management, MM application design, multicast*

1 Introduction

The design of distributed multimedia applications, such as systems for access to remote multimedia databases or teleconferencing, requires carful consideration of quality of service (QoS) issues, because the presentation quality of live media, especially video, requires relatively high utilisation of networking bandwidth and processing power in the end systems. For applications running in a shared environment, the allocation and management of these resources is an important question, although most existing systems are based on a best-effort approach.

In general, best-effort approaches are not suitable for distributed multimedia systems, because some users may be ready to pay some higher price for obtaining a maximum quality, while others may prefer low-cost presentations with lower quality. In addition, for a teleconferencing application involving many users, a single quality of service level may not be appropriate for all participating users, since some users may participate with a very limited local workstation which cannot provide for the quality which is adopted by the majority of the conference participants. We therefore adopt the premise that different levels of quality, often corresponding to different levels of cost, must be provided in the context of distributed multimedia applications.

Much work on QoS has been done in the context of high-speed networks in order to provide for

some guarantee of quality for the provided communication service, which is characterized by the bandwidth of the media stream and the delay, jitter and loss rate provided by the network. More recently, QoS have been considered in a more global context, including also the end systems, such as the user's workstations and database servers. Various global QoS architectures have been developed (for a recent overview see [1]), which include also functions for performance monitoring, resource allocation and QoS management. For instance, in previous work [6], we have developed a framework for QoS management of distributed multimedia applications which stresses two points: (a) the user should define (through a suitable user interface for QoS negotiation) the criteria which are used by the system to select the "best" system configuration for the application at hand, and (b) the selection of an appropriate system configuration is the first step of the QoS management process, followed by resource reservation and commitment, which is performed during the initialization of the multimedia application and each time a QoS renegotiation is required. Renegotiation may be initiated by the user if his/her preferences change, or by the application when some system component does not satisfy the initially agreed QoS characteristics.

A prototype system has been developed which implemented the above ideas for the application of remote access to multimedia databases [5]. In this context, it was assumed that, for a given monomedia component of a multimedia document, such as a video clip, there may be several variants with identical "content", but with different QoS characteristics, possibly stored in different continuous-media file servers. During the initial access to the document, the QoS manager would select the most suitable variant to be presented to the user, and in case of network congestion, for instance, with the video server in question, another variant may be selected which resides in a different server. The negotiation process which leads to the selection of a variant involves three parties: (1) the database server, which contains the meta-information of the documents including all existing variants, (2) the network and (3) the user workstation, which knows the user's preferences and may also impose certain QoS restrictions.

In multimedia applications including multicasting to many users, such as teleconferencing or educational applications, this global QoS management approach which involves a few system components, as e.g. for remote database access [5] of single users, is not workable any more, because the number of users involved is too large for a global management approach. For instance, negotiation of QoS parameters between the sender and every single receiver becomes impossible, since (1) the system would quickly become overloaded and (2) it would have to take into account (and possibly provide) many different qualities requested by users. Instead, a more decentralized approach seems suitable, where QoS management functions such as QoS negotiation, adaptation or renegotiation are distributed over the network. We developed such an approach called *Cooperative QoS Management* [4], where so-called *QoS agents* are installed on the routers and end systems participating in an applications. These agents cooperate with each other in order to provide the QoS levels requested by the application. An interesting new feature, compared to other QoS management schemes, which becomes possible due to this decentralized approach, is the possibility of communication between users resp. their local QoS agents, allowing for a cooperative selection of desired qualities. If users cooperate and decide to request a service in the same quality, less resources have to be reserved, which in turn leads to lower communication costs and higher resource availability for other applications.

In this paper, we show how the design of applications is affected when based on Cooperative QoS Management. We concentrate on the user/QoS interface and the interaction between application and the end system QoS agent. The rest of the paper is organized as follows: in Section 2, we give an introduction into the principles of Cooperative QoS Management. Then, in Section 3, we

compare the functionality of the new scheme with the one we described in [6,7] for presentational applications and show how the differences influence the design of an application resp. the end system QoS agent. Section 4 presents as an example the teleteaching education we develop in the framework of our project “Broadband Services”. Section 5 finally concludes the paper.

2 Principles of cooperative QoS management

Cooperative QoS management has been developed with multimedia applications in mind, in which many users participate at the same time, such as teleeducation systems or life video transmissions of major sports events. We assume that single data streams are multicast to many users and that senders offer the same media stream in several qualities, e.g. a high, a medium and a low quality video stream. There are no individual QoS negotiations between senders and receivers; rather, receivers have to select among the qualities offered by the senders.

The basic idea of our new scheme is to install an application-oriented QoS agent on each router of the underlying network and on every end system participating in an application. These QoS agents are able to communicate with their neighboring agents, informing them e.g. about current QoS values supported in their local area or about possible QoS problems. This knowledge is basically *application-oriented*, i.e. the agents know about QoS requirements and negotiated values for users as well as relationships between streams. This constitutes a main difference of our approach compared to existing QoS management functions on network nodes which deal with lower-layer QoS, such as ATM cell loss priority etc., and which do not have any information about relationships between streams and applications.

In our approach, however, not every agent may contact any other agent. Rather, communication depends on the existing multicast trees, leading to a hierarchical communication structure. For each multicast tree in which a given router is involved, the QoS agent knows its upstream and all downstream neighbors. If the neighboring node is an end system, the agent knows all receivers on this end system. A receiver’s QoS agent knows only its upstream QoS agent(s), and a sender’s agent its downstream neighbors. The information about neighbors may be easily set up during the establishment of the multicast tree, resp. when a member leaves or a new member joins.

As an example, consider the situation displayed in Figure 1 where one sender is multicasting one high-quality video (the regular arrows) and one low-quality video (the dashed arrows) to a group of receivers.

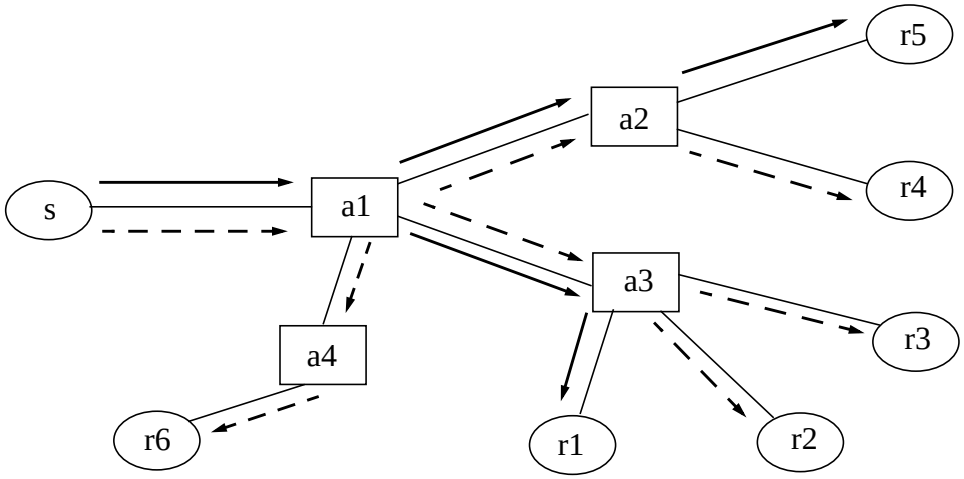

Figure 1. Multicasting streams of different qualities.

Every router in the network has to forward all the streams which are requested by users connected via this router. The QoS agent a2 knows its downstream agents r4 and r5 as well as its upstream agent a1. It also has information about the available resources on its router and the cost associated with reserving them. Finally, it knows all streams available for this application and has access to the multicast routing and resource reservation protocol running on this router. Note that our techniques are independent of underlying protocols and mechanisms and work for different coding and routing techniques, such as hierarchical video encoding [13], multicast routing already including resource reservations as discussed e.g. in [10], traditional MBone routing techniques [3] or video selection using group management protocols [12]. Our initial considerations were based on the multicast routing protocol core-based tree routing (CBT) [2][1] and the resource reservation protocol RSVP [14].

A QoS agent provides the following QoS functions:

- QoS negotiation
  It occurs when a new user requests to become part of the application and receive some of its streams. The multicast routing scheme will forward its request until it arrives at a router that is already participating in the requested application, i.e., supporting its multicast tree. The agent of this router then contacts the new user's agent and sends the information about all available streams (quality and cost). Note that there is no central instance providing quality and cost information, since the cost for available qualities may differ significantly from one region to another. The user may select the streams he desires. Connections are set up by the underlying protocols; the QoS agents update their information about supported streams. We developed protocols to fulfill these tasks [4].
- QoS adaptation
  This functions becomes active when a component is no longer able to support the currently negotiated QoS, e.g. due to overload, failure or other stochastic situations. The QoS management system then tries to find a way to continue providing the service, either by selecting another component or by lowering the service quality within the boundaries negotiated with the client. In the framework of Cooperative QoS Management, we developed a protocol between QoS agents that helps to detect QoS violations and locate their source; furthermore, QoS agents can initiate the adaptation process by several means, one of them being to request the partial reconfiguration of the multicast tree in the area where the problem occurred. More details on the adaptation protocols can be found in [4].
- QoS renegotiation
  Traditionally, there are two types of QoS renegotiation, namely system-initiated and user-initiated. The former occurs when a negotiated QoS was violated and the QoS adaptation was not able to fix the problem. Then the system proposes the user to negotiate a lower quality. The latter happens when users are no longer content with the quality they negotiated. In such a case, they start a new negotiation process to switch to another quality.
  Within Cooperative QoS Management, system-initiated renegotiation may also occur when the management detects an unsatisfying situation concerning resource usage as described in Section 1. Consider again the example in Figure 1. Receiver r1 is the only one on its subtree that receives the high-quality video, i.e., he is the exclusive user of the resources reserved for this stream. QoS agent a3 realizes this, and after checking several other parameters, it

1. In Core-Based Tree routing, there is only one multicast tree per receiver group. All streams are first unicast to the root of this tree (the core) and from there multicast to the receivers. Several optimizations are possible.

decides to propose to r1 to switch to the lower quality. If it already has the necessary information, r1's agent may take the decision on its own, but it may also contact the user and ask if he would like to switch. Certainly, users may forbid their agents to forward any such requests to them, in order to not be disturbed in their session.
If r1 considers to switch to the lower quality, resources for the high-quality video on a3's router could be freed and the communication service cost would be much lower for receiver r1 which would be the motivation for him to switch. Assume that he pays 10 money units for the high-quality stream. If the system is able to offer him the low-quality stream, which is black&white instead of colored, for 1 money unit, it would probably be very tempting to switch.
If r1 finally switches, a new situation for the other agents occurs. Consider agent a1 now. It realizes that only the subtree of agent a3 receives the high-quality video. It may now try to persuade a3 to switch the quality which in turn could lead to a3 sending a switch proposal to r5.

3 Design Considerations for Cooperating Applications

In [7], we developed a framework for QoS management in the context of *presentational* multimedia applications. A set of components that are involved in the QoS provision has been identified, namely the user, the database server, the continuous media file server, the transport entities, the end-systems entities, the network, etc. In this architecture, the QoS manager has the ability to control and monitor each individual component. In order to do so, a QoS interface is defined at each component. It allows the QoS manager and other system components to inquire and control the QoS features of the service provided by the component.

The QoS negotiation process is split into three phases: first, the user specifies his QoS requirements via user profiles. These files can be produced offline and stored for different users or applications. Figure 2 shows the QoS user profile interface in the context of our news-on-demand application [5]. At application startup time, such a profile is passed to the QoS manager which in turn tries to find those audio and video stream, image and text variants of the application that best satisfy the stated requirements. Second, it produces a functional configuration which is necessary to support the transmission and display of all monomedia components of the selected document variant. For an MPEG video, for instance, the functional configuration will include an MPEG encoding function on the end system. Third, this functional configuration is mapped onto a physical configuration. In case of the MPEG movie, the manager could use a software or a hardware decoder to provide the decoding function, depending on availability and required quality. Then, the QoS manager takes the necessary steps to set up the session. If the selected variant cannot be supported due to dynamic overload situations, the QoS manager repeats steps two and three with the second-best variant.

Once a session is established, the QoS manager has to monitor the overall system in order to detect any QoS violations and to notify the appropriate entity in time. The Qos adaptation function is used when the user specified requirement can not be supported or when temporary difficulties rise during a session. The QoS manager in this case, should be able to find alternative solutions to maintain the session ongoing. The QoS manager also provides the user with the ability to dynamically renegotiate the initially specified QoS.

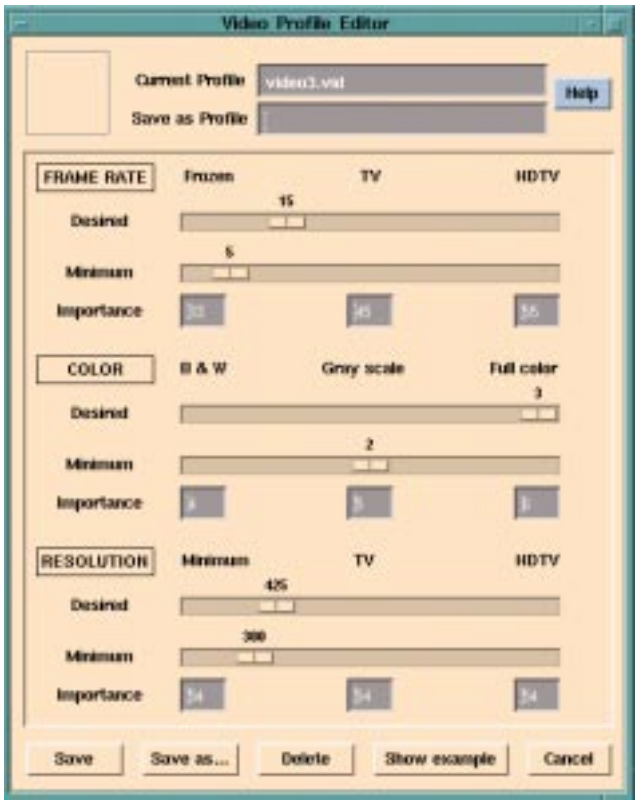

Figure 2. The QoS User Interface for presentational applications

The multicast nature of many *conversational* applications, the participation of a huge number of users and the use of Cooperative QoS Management calls for a revision of this framework. We identified a number of major changes which mainly affect two application components: a user-visible part, namely the QoS interface, and an invisible part, namely the QoS agent.

- QoS user interface
  In our framework for presentational multimedia applications, users resp. QoS managers do not have the information on all existing multimedia document variants, and they access multimedia documents in individual sessions. The QoS user graphical interface was then designed to give users the ability to specify the desired quality of service for each media type (audio, video, etc.) and thus to limit the number of document variants to be checked and offered by the QoS manager. As we already said at the beginning, we believe that in multicast applications, only a limited number of qualities should be offered for a given media type. Using Cooperative QoS Management, the information about all available media types and qualities is available from the network QoS agents. Thus, we do not ask the user to specify his requirements is advance, but offer him all available qualities for the session components. He then simply selects the ones desired. The new QoS graphical interface (see Figure 5) will then be more user-friendly as in the approach for presentational applications (Figure 2).
  As a result, step one of the negotiation process described above is completely eliminated.
  In addition, new interactive possibilities are added to the interface in order to support the new possibilities offered by our QoS management scheme, namely the QoS renegotiation in

case of the arrival of a PERSUADE message from the network. If the user checks the respective box on the interface, he allows his local agent to forward such requests.

- QoS agent
  As described above, the QoS manager in presentational multimedia applications goes through several consecutive operations to implement the negotiation process between a client and a server. This process involves different interactions between the QoS manager and the user on one hand, and the QoS manager and the remote service providers on the other. When the number of users becomes large, as in our teleteaching target application, this process becomes too heavy for the whole system, since every single component would be in steady negotiation processes with new users. Due to the multicast nature of the application and the characteristics of Cooperative QoS Management, however, the design of the QoS manager can be simplified by far. If a certain stream is already supported in the area of the new user, then the QoS negotiation process need not be carried out; rather, the stream is simply multicast to the new user. Thus, an end system QoS agent has only to deal with QoS management on its own system. All issues concerning negotiations with, resource reservations at and adaptation of remote components are part of the work of the network QoS agents. End system QoS agents simply send messages to their upstream agents, stating the types and qualities of documents/media they would like to receive.

The remaining components of what we assume to be the application part of a system (s. also Figure 4) - the application itself and the QoS monitor - remain basically unchanged.

4 An Example: Teleteaching

The teleteaching application we use as an example supports the delivery of a lecture from a given site to students located in several remote locations. The delivery consists of video and audio from the lecturer. In addition, the lecturer may present multimedia documents stored locally or in some other locations. Students have the possibility to ask questions, but they first have to get permission to do so. In this prototype, we allow only one student to talk at a time. The lecturer is always allowed to talk. In addition, it is possible to record a whole session and store it as a multimedia document in the multimedia databases. Thus, lectures may be later reviewed by students when preparing for exams. The overall structure of this application is visualized in Figure 3.

Our implementation environment consists of several heterogeneously equipped machines connected via Ethernet and ATM and communicating via TCP/IP and the MBone multicast and group management protocols. We do not develop a completely new application environment, but use instead the existing MBone tools [11] *vic* for video, *vat* for audio, *wb* for the whiteboard and the MBone VCR [8] to record the sessions. These tools are combined with our QoS interface developed for presentational applications, which has however been modified in order to be suitable for multicast applications and Cooperative QoS management. We transfer at most one stream to a given multicast address, which allows for a finer grained management. The software architecture within an end system is shown in Figure 4.

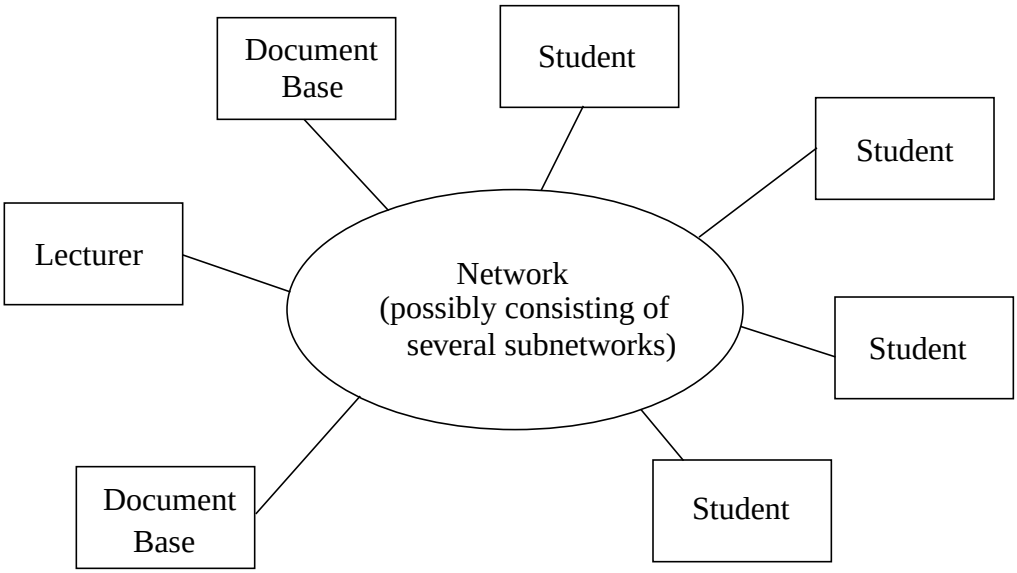

Figure 3. Structure of the remote teaching application.

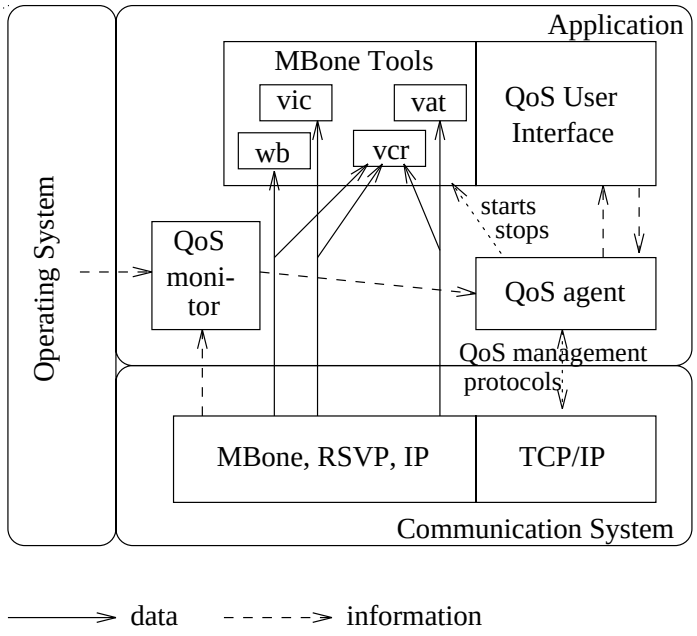

Figure 4. Software architecture in an end system.

We describe the software architecture of the application by describing the functions of its components during typical phases in the application's lifetime, namely application start, normal operation, QoS adaptation and QoS renegotiation.

Start of the application:

1. After being informed by the user about which application to join, the QoS agent collects information about available streams from the QoS agents in the network (Details on this process and the QoS protocols to be used are described in [4]).

2. Based on the answers from the network QoS agents, it assembles a list of available media streams, their qualities and the associated cost and sends it to the user. The list is displayed in the new QoS user interface (see Figure 5).

Figure 5. The new QoS user interface at startup time.

3. The user selects the streams he would like to receive and sends this information back to the QoS agent. As a way of specifying a QoS degradation path, the user may also indicate different qualities for a given stream he is also ready to accept. Thus, if during an adaptation process the quality has to be lowered, a renegotiation can be avoided.
4. The QoS agent knows the multicast group address for each of the selected streams. It starts the respective tools, at the same time informing them at which multicast address to listen. If the user selected the local session recording option in the user interface, the MBone VCR is also started. As a result, incoming data streams are not only sent to the display tools, but are also forked to the VCR. Usually, however, there will be a special multimedia document server which records the session. Then, there is no need for a local recording. In order to start session recording and to allow for a comfortable search for and playback of recorded sessions, we plan to employ a new technology described in [9], also including the MBone VCR.

Normal operation:

During normal operation, the quality of all streams is constantly monitored by the QoS monitor component. It checks the the incoming packets as well as the ability of the operating system to display the data in time. If a QoS violation occurs, the monitor informs the QoS agent. The QoS user interface remains functional in order to allow for renegotiation. It only consists of a renegotiation button. If the user is no longer satisfied with the quality of a stream he receives, he simply initiates a renegotiation process by pressing the button. Then the interface shown in Figure 5 is re-assem-

bled by the QoS agent and put on the screen.

QoS adaptation:

In case the QoS monitor detected a violation and informed the QoS agent, the latter will start the QoS adaptation process. If the problem occurred on the end system, it will try to fix it by informing the operating system. If a real-time operating system is available (which is not the case in our implementation), it could for instance require stronger scheduling bounds for the thread(s) running the violated stream. If, on the other hand, the problem is located somewhere in the net, the QoS agent sends out a violation message and thus initiates the cooperative adaptation process with the network QoS agents. The following outcomes are possible:

- The network is able to adapt, e.g. by selecting a different or modified multicast tree that is able to support the desired quality. Then the quality will improve and the monitor will remove the violation flag set before. Operation continues normally.
- The network is not able to adapt. Then, the QoS agent receives a control message telling him so. In this case, or when the operating system cannot adapt, the agent has two options: (1) If the stream in question is still available in another lower quality, which has earlier been specified as acceptable by the user, it can select this quality. This implies switching to another multicast address. In our implementation, the respective tool is stopped (in terms of the operating system, the process is killed) and is restarted with the new multicast address. It now receives and displays packets of the lower quality stream. (2) If there is no further lower quality which is acceptable, the agent informs the user via the QoS interface and offers him to switch to one of the currently supportable qualities.

Please note that the user is not informed about the ongoing adaptation process as long as the quality remains within the specified bounds.

QoS renegotiation:

Renegotiation has to be initiated in case one of the following events occurs:

- The user is no longer satisfied with the delivered quality of a certain stream. He indicates this by pressing the “renegotiate” button on the QoS interface.
- The agent detects that it cannot provide the currently negotiated quality anymore. Also, there is no lower quality which has been marked as acceptable by the user.
- The agent receives a PERSUADE message from one of its upstream QoS agents asking to switch to another quality in order to optimize system operation and decrease communication cost. Renegotiation is only initiated if (1) the quality to be switched to has not been marked as acceptable by the user and (2) the user did not indicate that he does not want to receive such suggestions. If (2) is not true, the QoS agent simply ignores the PERSUADE message, while if (1) is not true, it is allowed to switch to the new quality without user interaction.

When the user has asked for or decided to accept a new quality for a given stream, the agent again stops the respective tool and restarts it with the new multicast address.

5 Conclusions and Outlook

In this paper, we discussed design issues of conversational multimedia applications which are based on our new Cooperative QoS management. We concentrated on the QoS agent and user interface it provides and compared it to our earlier approach which was developed for presentational multimedia applications. An example - a teleteaching application - showed how these general design issues can be implemented in a concrete application.

Most of the teleteaching application prototype is based on existing code from our news-on-demand application and the existing MBone tools. Since we are interested in a more wide-spread use of our approach, we are currently working on the design of a Web- and Java-based implementation of both the QoS agent user interface and the QoS agents inside the network. This will allow for a more flexible distribution and an enhanced mobility of QoS agents, i.e., there won't be any pre-installed QoS agents on routers. Rather, they will be distributed "on demand" and only run on those routers that provide communication service for the given application.

We are also interested in the technique of hierarchical video encoding [13]. We plan to use the code of one of the existing MBone video tools like *vic* and modify it in order to support this technique. The final goal is to get a flexible implementation which is able to switch between qualities by processing more resp. fewer incoming data streams making up the video. Compared to our current solution with several independent streams broadcast simultaneously, we will no longer have to stop a running *vic* process and start a new one when we switch qualities. Rather, the QoS agent has to direct the respective streams to the new *vic* implementation. This implies that the tool can no longer work directly on a single multicast address, since every single hierarchically encoded data stream will be sent to its own multicast address as described in [12]. The multicast address management will therefore be done by the end system's QoS agent.

References

[1] C. Aurrecoechea, A. Campbell, and L. Hauw. A Survey of QOS Architectures. *Multimedia Systems Journal, Special Issue on QoS Architectures*, 1997. To appear.

[2] A. Ballardie, J. Crowcroft, and P. Francis. Core based trees (CBT) – An Architecture for Scalable Inter-Domain Multicast Routing. In *ACM SIGCOMM '93*, pages 85–95, 1993.

[3] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.

[4] S. Fischer, A. Hafid, G. v. Bochmann, and H. de Meer. Cooperative Qos Management in Multimedia Applications. In N. Georganas, editor, *IEEE International Conference on Multimedia Computing and Systems (ICMCS'97), Ottawa, Canada*. IEEE Computer Society Press, June 1997. To appear.

[5] A. Hafid and G. v. Bochmann. Quality of Service Negotiation in News-on-Demand Systems: An Implementation. In A. Azcorra, T. D. Miguel, E. Pastor, and E. Vazquez, editors, *Proceedings of the Third International Workshop on Protocols for Multimedia Systems, Madrid, Spain*, pages 221–240, Oct. 1996.

[6] A. Hafid, G. v. Bochmann, B. Kerherve, R. Dssouli, and J. Gecsei. On Quality of Service Negotiation in Distributed Multimedia Applications. Technical Report #977, University of Montreal, Montreal, Canada, 1995. (submitted to IEEE JSAC).

[7] A. Hafid and G. von Bochmann. A General Framework for Quality of Service Management. Submitted to Computer Communications, Special Issue on Building Quality of Service into Distributed Systems, 1997.

[8] W. Holfelder. MBONE VCR – Video Conference Recording on the MBONE. In P. Zellweger, editor, *ACM Multimedia '95 (Proceedings)*, pages 237–238, New York, Nov. 1995. ACM.

[9] W. Holfelder. Interactive remote recording and playback of multicast videoconferences. Submitted to 4th European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97), Darmstadt., 1997.

[10] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.

[11] V. Kumar. *MBone – Interactive Multimedia on the Internet*. New Riders Publishing, Indianapolis, Indiana, 1996.

[12] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM SIGCOMM'96*, Stanford, CA, Aug. 1996.

[13] N. Shacham. Multipoint communication by hierarchically encoded data. In *IEEE Infocom'92*, pages 2107–2114, 1992.

[14] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 7(5), Sept. 1993.