Quality of Service Management for Teleteaching Applications Using the MPEG-4/DMIF

Gregor v. Bochmann and Zhen Yang

School of Information Technology and Engineering (SITE), University of Ottawa, Canada
Bochmann@site.uottawa.ca, zyang@site.uottawa.ca

Abstract. In the context of distributed multimedia applications involving multicast to a large number of users, a single quality of service level may not be appropriate for all participants. It is necessary to distribute part of the QoS management process and allow each user process to make certain QoS decisions based on its local context. In order to allow for different QoS options, we assume that each source provides, for each logical multimedia stream, several different *stream variants*, representing different choices of user-level QoS parameters. The paper presents the design of a teleteaching system which uses this paradigm for QoS negotiation, and explains how the Delivery Multimedia Integration Framework (DMIF) of MPEG-4 can be adapted as a session protocol for such an application. In fact, it appears that this DMIF protocol, which is now being extended by ISO (DMIF Version 2) to the context of multicasting, provides some general session management functions which are quite useful for many distributed multimedia applications using broadcasting.

1. Introduction

Over the past several years, there has been a large amount of research focussed on the issue of QoS management techniques. The topics range from end-to-end QoS specification, adaptive QoS architecture, to QoS management agents, etc. The research covered both architecture and implementation issues of QoS management functions, such as QoS negotiation, QoS renegotiation, QoS adaptation, QoS mapping, resource reservation, and QoS monitoring. For instance, [1] investigated the problems related to providing applications with a negotiated QoS on an end to end basis. [2] illustrated some examples of applications that adapt to a certain situation in which the available QoS may be severely limited. In our previous CITR project, "Quality of Service negotiation and adaptation", solutions were developed for applications involving access to remote multimedia database [3], [4].

The concept of multicasting arose years ago [8]. It is playing an important role in distributed multimedia applications, such as remote-education, tele-conferencing, computer supported collaborative work, etc. However, QoS management in the context of multicast has only been addressed recently. Most work in this field has been done for providing QoS-based multicast routing schemes. [9] proposed a multicast

protocol that considers QoS in the routing phase and can create a multicast tree that better meets the needs of QoS sensitive applications. Another study was done to provide QoS control for an existing core-based multicast routing protocol [10]. Stefan Fischer and other researchers proposed a new scheme for cooperative QoS management, which takes a different approach. They considered an adaptive application that suits different QoS requirements of different users in the context of multicast applications [5] [6] [7]. Our project has been greatly inspired by this approach.

Quality of Service management for distributed multimedia applications becomes more complex when a large number of users are involved. A single quality of service level may not be appropriate for all participants, since some of the users may participate with a very limited workstation, which cannot provide for the quality that is adopted by the other users. It is necessary to distribute part of the QoS management process and allow each user process to make certain QoS decisions based on its local context.

We have developed a framework for QoS management of tele-teaching applications. It was assumed that different variants (with the identical content, but different QoS characteristics) of mono-media streams are available to the students' workstations throughout the network by means of multicasting. A so-called QoS agent is installed on each student's workstation. The QoS agent may select the stream that is most appropriate based on the student's preference.

2. Providing QoS Alternatives for Multicast Applications

2.1. General Assumptions

We consider in this paper distributed multimedia applications where most data originates in one system node, called the sender, and is multicast to a large number of receiver nodes. A typical application would be teleteaching where the teacher's node is the sender and the students' nodes are receivers. We note that in such an application, there may also be some real-time data going from a receiver node to the sender and the other receiver nodes. For instance, a student may ask a question which is broadcast to all participants in the teaching session. However, we will ignore this aspect in our discussion.

Figure 1 shows the overall system architecture which we consider for a single instance of our multimedia application. The sender and receiver nodes are connected to one another through a network which supports a multicast service. The video and audio streams originating from the sender are therefore broadcast to the different receiver nodes that participate in the application session.

We assume that the users at the different receiver workstations have different quality of service requirements. These requirements may be due to the following reasons:

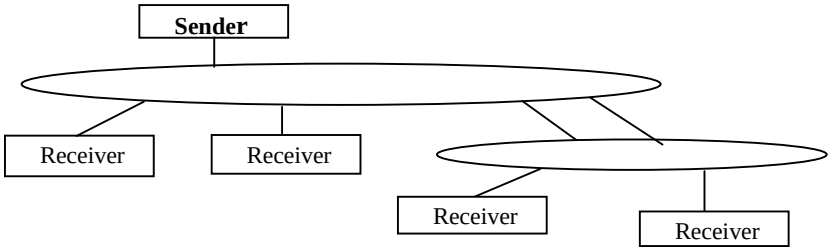

Fig. 1. **Overall System Architecture**

1) different *hardware and/or software resources* available in the workstation,
2) different *user-level QoS parameters* (that is, user preferences), such as requiring low-cost network service or high reception quality (which may imply higher costs); there may be different priorities for different aspects of quality, such as frame rate, color, resolution, disturbances through packet losses, etc.
3) different *transmission-level QoS parameters* (provided by the network) for the different receivers due to the specific network architecture and interconnection structure.

In order to accommodate these different QoS requirements, we assume that the sender node provides for each *logical multimedia stream* (such as for instance "video of teacher", "video of demonstration", and "audio of teacher") different *stream variants*, each representing a specific choice of user-level QoS parameters. Actually, some stream variant may represent several user-level qualities in the case that some form of scalable encoding is used; however, in most cases scalable encoding implies several elementary streams which can be combined in order to obtain a specific quality.

2.2. General System Architecture

Figure 2 shows a general system architecture for multicast applications with QoS alternatives. The figure includes a sender node and two receiver nodes. The receiver nodes communicate with a user profile manager that contains information about the user's QoS preferences and may also be used for user authentification. The sender node may also communicates with a local network QoS monitor which, in the case of best-effort networks, provides information about the transmission quality that is presently available.

The architecture of the sender and that of the receiver nodes are similar. They contain the application module that performs the application-specific functions, including all streams processing (e.g. video capture, coding in several stream variants, transmission, reception, decoding and display, etc.). The transport layer provides an end-to-end data transmission service with multicasting. In our prototype implementation, we assume that this includes the protocols IP (with MBone multicasting), UDP and RTP/RTCP. The session management layer looks after the management of the application session,

including the management of the transport channels for the different multimedia streams and the knowledge about the participating users.

The QoS manager in the sender node determines the list of potential stream variants for each logical multimedia stream. However, not all of these variants will actually be transmitted. A stream variant become active (and is transmitted) if the QoS manager considers that there are enough users that receive that stream, or that have requested that the stream be activated. For the selection of the potential stream variants and the activation of some of these variants, the QoS manager may take into account the information about the presently available network transmission quality that can be obtained from the local network QoS monitor and through the monitoring of the active transport channels. It may also take into account specific requests sent by the users participating in the application. An example of potential stream variants is shown in the table of Figure 3.

Video Stream A			
	Ch1	**…**	**Chn**
Frame Rate	10		30
Color	Grey		Color
Resolution	640*480		640*480
Coding Scheme	h.261		jpeg
Cost	10		20
Active Flag	Yes		No

Audio Stream B			
	Ch1	**…**	**Chn**
Quality	CD		Phone
Coding Scheme	PCM		PCM
Cost	2		1
Active Flag	Yes		No

Fig. 3. QoS Table for Logical Multimedia Streams

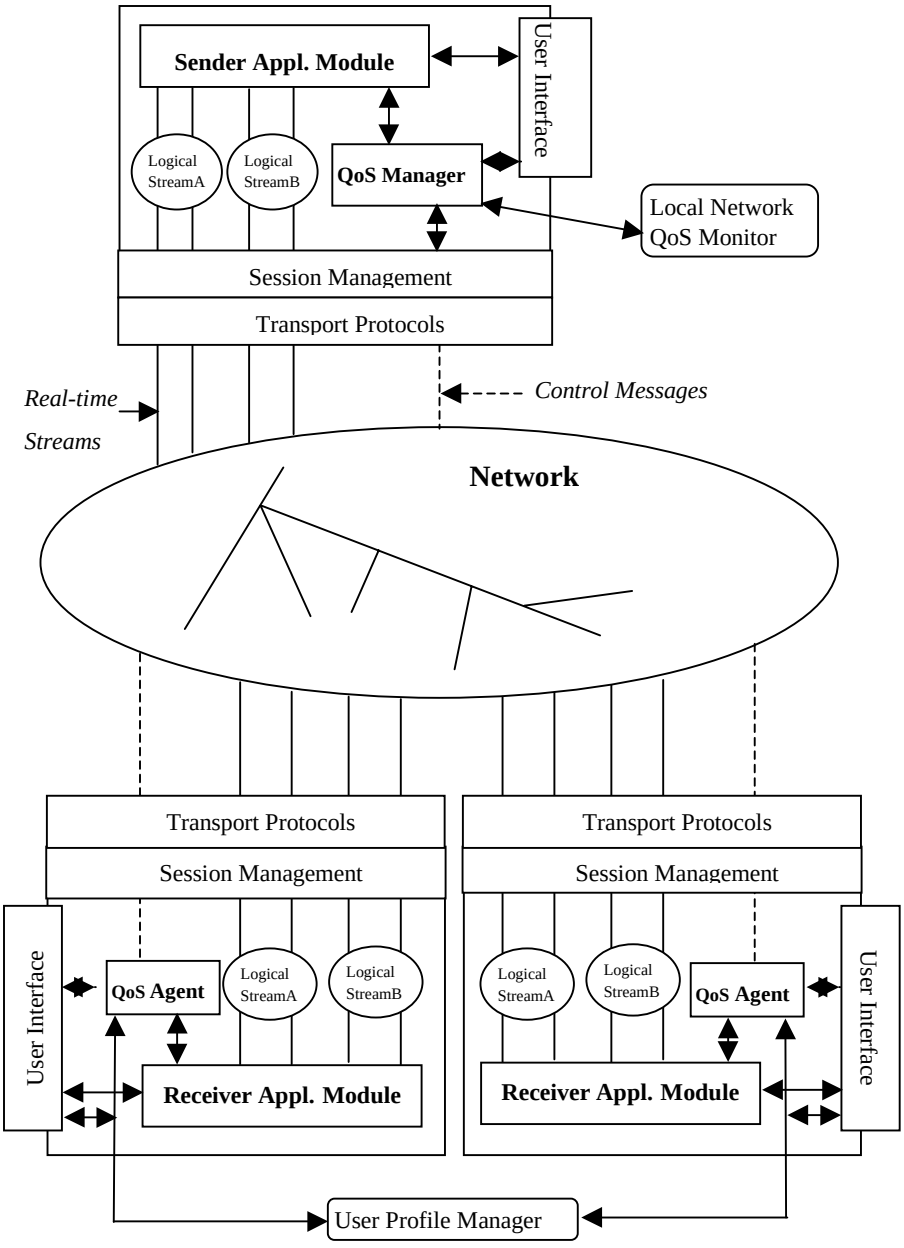

Fig. 2. General System Architecture

The QoS agent in the receiver node obtains information about the user QoS preferences (either directly through the user interface, or by retrieving the user's QoS profile) and selects for each logical multimedia stream a specific stream variants which best fits the QoS preferences of the user. In case that the most suitable stream variant is not active, it sends an activation request to the QoS manager at the sender node.

An example of a possible interaction scenario is shown in Figure 4. Point [1] shows the QoS Manager of the sender that initiates the session and broadcast the information about the available streams. When a new receiver joins the session (point [2]), its QoS Agent sends first an *Ask_QoS_Info (session_ID, user_ID)* to the QoS Manager. The QoS Manager then sends *Give_QoS_Info (session_ID, qos_List)* to the Agent, providing information for all potential stream variants, as shown in Figure 3. According to the user's profile, the QoS Agent then selects the best stream variant (*port1* in our example).

When a QoS violation is detected by the QoS agent, it first checks whether one of the other currently active streams is acceptable. If so, it switches to it (point [3]), otherwise (point [4]) the QoS Agent will ask the QoS Manager to activate one of the other potential stream by sending an *Ask_Add_QoS (session_ID, user_ID, qos)*. If none of the streams, whether active or not, satisfies the user's expectations, the QoS Agent should inform the user about this situation and check whether the user wants to change the preferences in his/her profile.

Whenever there is a change in table of potential streams, the QoS manager should broadcast this information to all the receivers by sending a *Give_QoS_Info (session_ID, qos_List)* message.

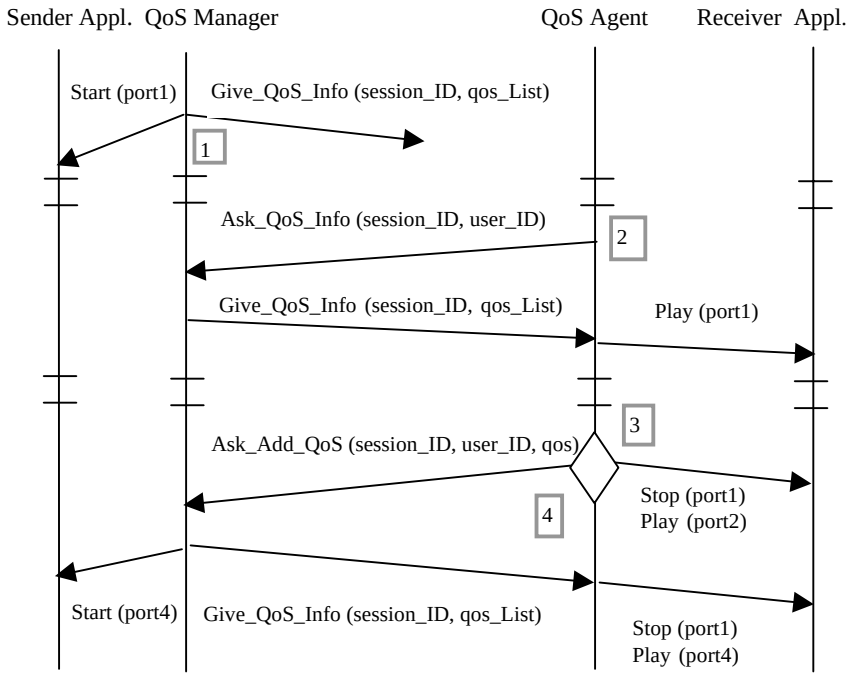

Fig. 4. Example of an Application-Level Interaction Scenario

Notes:
Point [1]: When the QoS manager initiates a session, it should broadcast the information about all potential stream variants. The parameter *qosList* contains the QoS table (see Figure 3) plus the IP addresses and port numbers of all active channels.
Point [2]: Some time later, a new receiver wants to join the session. First, its QoS agent asks the QoS manager for the QoS information (*qosList*) of the session. Then it selects an appropriate stream variant, and asks the Receiver Application Module to start the application with the selected stream.
Point [3]: After some time, QoS violation occurs. If there is another currently active stream that is acceptable, the QoS agent simply asks the Receiver Application Module to switch to that stream.
Point [4]: If no currently active stream is acceptable, the QoS agent asks the QoS manager to activate one of the other potential streams. The scenario above assumes that the QoS manager activates the requested stream; when the receiver's QoS agent is informed, it switches to the new stream. -- If none of the variants that the QoS Manager offers, whether active or not, satisfies the user's expectations, the QoS agent should inform the user about this situation and check whether the user wants to change the preferences in his/her profile.

3. The MPEG-4 / DMIF

MPEG-4 [11] is a new ISO standard on multimedia stream coding which goes beyond the previous MPEG-1 and MPEG-2 standards by providing adaptation to for low-data-rate transmission and support for multiple video and audio streams which can be useful for teleconferencing applications and applications involving virtual environments. In contrast to traditional coding standards, MPEG-4 allows the definition of video streams which represents objects with arbitrary contours, not only the rectangular screens of TV or film presentations.

The suite of MPEG-4 standards include a so-called Delivery Multimedia Integration Framework (DMIF) which is functionally located between the MPEG-4 application and the transport service. The main purposes of DMIF are to define a session level protocol for the management of real time, QoS sensitive streams, to hide the delivery technology details from the DMIF user, and to ensure interoperability between end-systems in the control plane.

DMIF defines an interface called DMIF application interface (DAI), in order to hide the delivery technology details from applications. Also, by using media related QoS metrics at the DAI interface, applications are able to express their needs for QoS without the knowledge of the delivery technology. It satisfies the requirements of broadcast, local storage and remote interactive scenarios. In addition, in case of interactive operation across a network, it ensures interoperability between end systems through a common DMIF protocol and network interface (DNI), which is mapped into the corresponding native network signaling messages. The basic DMIF concepts are defined in MPEG-4 version 1 [12]. Version 2, now being developed, specifies extensions for multicast scenarios [13]. Our project is based on these extensions.

The DAI provides primitives for an application to attach itself to a given session (or create a new session), called *DA_ServiceAttach*, primitives for adding or deleting a transport channel for the session (*DA_ChannelAdd* and *DA_ChannelDelete* primitives). It also includes a *DA_UserCommand* primitive that provides a means for transmitting, between the applications. Similar primitives exist at the network interface (DNI), which includes in addition so-called *SyncSource* and *SyncState* primitives which allow the distribution of the information about the multimedia streams provided by the different senders in the application session. The *AddChannel* primitives at the DNI also include a parameter for requesting different options of QoS monitoring for the transmission service provided by the channel.

In a multicast session, a DMIF terminal can be either a Data Producer DMIF Terminal (DPDT) that is a information source, or/and a Data Consumer DMIF Terminal (DCDT) that is a information receiver. A DMIF multicast session consists of a DMIF multicast signaling channel (C-plane) to distribute the state information of the session, and one or more multicast transport channels (U-plane) to deliver the multimedia data. We will use the DMIF C-plane in our project since it meets our need for message exchange between QoS manager and QoS agents.

The following are some details on the establishment of a multicast session between data produces and consumer DMIF terminals:

- A DMIF multicast session is identified by an DMIF-URL. DMIF-URL is a URL string, whose basic format was defined in DMIF version 1. It is used to identify the location of a remote DMIF instance. In the multicast scenario, this URL is extended with the role of the DMIF terminal in the multicast session (DPDT / DCDT). So that the local DMIF layer is capable of recognizing the role of the application (sender or receiver) in the multicast session.
- The session's signaling channel address (IP address and port number) is available to the interested DMIF terminals by mean of a session directory or through e-mail, etc.
- Each DPDT must explicitly join and leave the DMIF multicast session by sending messages over the signaling channel.
- When a DCDT joins a session, in order to reduce the number of signaling messages, it should listen on the signaling channel to collect the state information from all DPDTs participating in the session. If the DCDT does not acquire the information within a given time period, it should ask the DPDTs for their state information.

4. Using DMIF for Session Management in Tele-teaching Applications

Although the DMIF of MPEG-4 has been designed for use with MPEG-4 applications, it appears that its functionality can be used in a much wider context. It appears to be a quite general session protocol that can be useful for any application using a number of concurrent multimedia streams in a distributed environment. Looking at it from this perspective, we asked ourselves whether it could be used for the teleteaching application described in Section 2. The answer is Yes, and the modalities are explained below.

We consider our teleteaching application in the context of the Internet. Therefore we assume that the underlying transport is provided by the IP/UDP protocols complemented with some multicasting facility. For our prototype implementation, we plan to use the multicasting facility provided by MBone. Each multicast channel, including the signaling channel, is identified by a broadcast IP address plus a port number.

We assume that RTP (and an associated RTCP) is used for each multimedia stream for the purpose of synchronization and for monitoring of the network-level QoS parameters. The DMIF network interface is therefore mapped onto the multicast transport service provided by RTP/RTCP. General guidelines are given in the DMIF specifications.

Over the DMIF layer lies the application. The abstract interactions shown in Figure 4 can be mapped to the DMIF primitives provided to the application through the DAI. A typical example is shown in Figure 5, which corresponds to the abstract scenario given in Figure 4. Specifically, the figure shows the following interactions:

Point [1]: The QoS manager initiates a session. The information on the potential stream variants (see Figure 3) is encoded in the user data field *uuData* of the

DA_ServiceAttach primitive and forwarded by the DMIF through the *SyncSource* and *SyncState* messages which are broadcast over the network through the signaling channel.

Point [2]: The QoS agent sends a *DA_ServieAttach* primitive to inform the local DMIF protocol entity that a receiver wants to join the session. To avoid requesting the information that other receivers just requested, the DMIF entity listens on the signaling channel for a reasonable time period, and collects the *SyncSource* and *SyncState* messages from the sender side. If these messages are not gotten during a given time period, the DMIF entity should request them. The received information is passed to the QoS agent in the *uuData* parameter of the *Attach* confirmation.

Point [2']: The QoS agent selects an appropriate stream variant, and sends a *DA_ChannelAddRsp* primitive. This primitive is originally used to respond to the DA_ChannelAdd primitive, but it can also be used at any time during a session when the application wants to connect to a channel. The DMIF will perform a group join operation at the DNI in order to join the MBone multicast group for the selected multimedia stream. The primitive *DA_ChannelMonitor* is invoked in order to inform the DMIF to start monitoring the selected channel.

Point [3]: The DMIF entity indicates a QoS violation, with detailed QoS information in *qosReport* paramter. If one of the other currently active streams is acceptable, the QoS agent simply decides to switch to that stream.

Point [4]: If none of the currently active streams is acceptable, but an inactive streams is acceptable, the QoS agent sends a *DA_UserCommand* primitive to ask for the activation of that stream, as specified in the *uuData* parameter. If neither active nor inactive streams are acceptable, the QoS agent should ask the user whether he/she wants to change his/her QoS profile or abandon the session.

We found that the DMIF specifications fit well with the session and QoS management functions that were required by our teleteaching application. However, there are certain points where the extensions for multicast applications described in ISO Working Draft of DMIF Version 2 do not fit completely our requirements. We mention in particular the following points:

- **SyncState message parameters:** The information about the potential stream variants (see Figure 3) is encoded in the user data field *uudata* of the *DA_ServiceAttach* primitive invoked by the QoS and session manager in the sender node. When a receiver node joins the application session, this information is passed along in the *SyncState* message from the sender to the receiver DMIF protocol entity. This requires certain changes to the *SyncState* message parameters, as specified in the Working Draft. More specifically, the SyncState message, as defined, contains information about the active channels, however, the QoS information included provides only information about the transmission-level QoS parameters of the channels, but not the user-level parameters contained in the table of Figure 3. These parameters must be added to the *SyncState* message. In addition, the information for the inactive channels must be added.

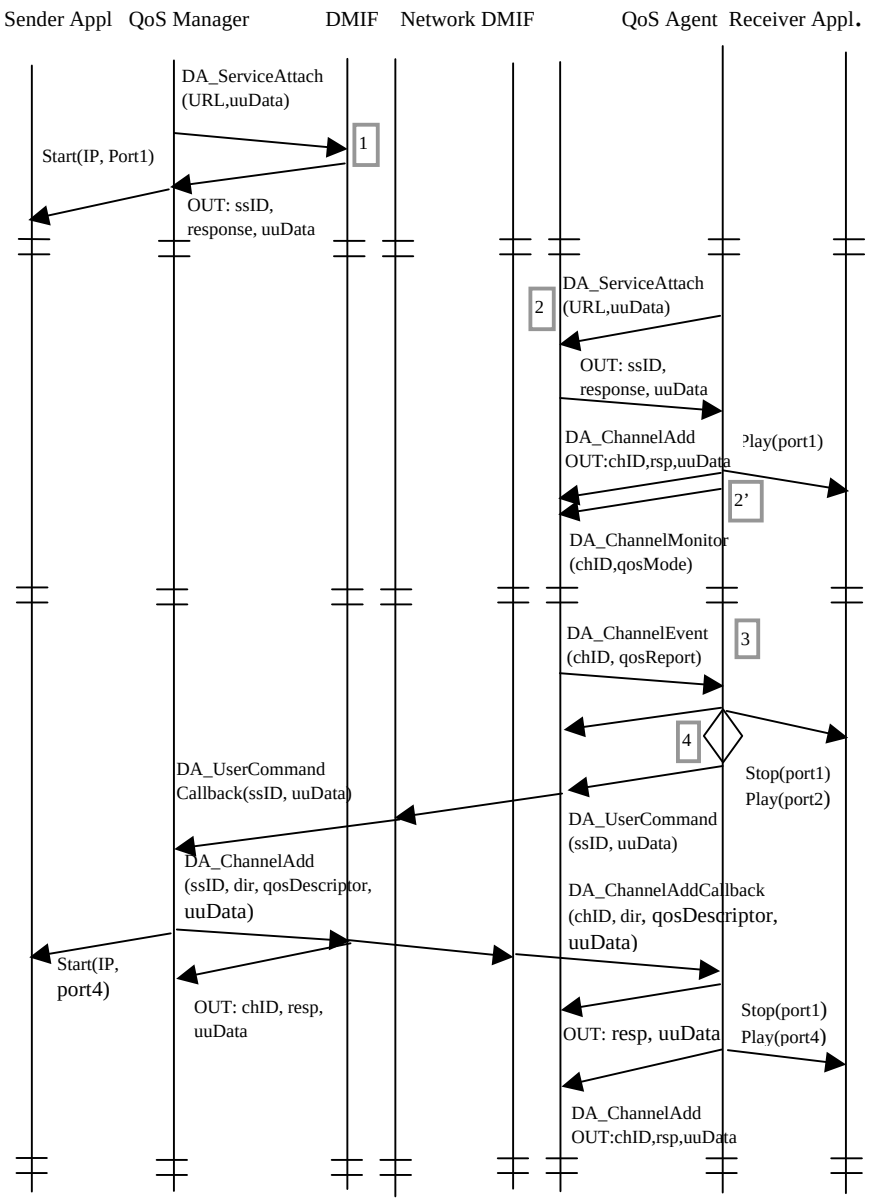

Fig. 5. Multicast Session Interaction Scenario including DMIF

- **Requesting new stream variants:** Another issue is the question how a QoS agent in a receiver node could request that an inactive stream variant be activated. We

have adopted the use of the *DA_UserCommand* primitives for this purpose. This primitive allows the transfer of user information from application to application, in our case from the QoS agent to the QoS manager in the sender node. If the QoS manager decides to activate a new stream variant, it will invoke a *DA_ChannelAdd* that leads to a multicast control message to all participating receiver nodes and a *DA_ChannelAdd* indication to all QoS agents.

5. Prototype Implementation

Our implementation environment is based on MBone. A single sender (teacher) node and several receiver (student) nodes communicate over the Internet via RTP/RTCP, UDP/IP and MBone protocols. At each side, there are two major parts: real-time video transmission and session control message transmission. For the first part, we did not develop a completely new multimedia application, but used an existed MBone video conferencing tool 'vic' at the teacher side, and a Java application 'RTP player' at the student side. 'RTP player' is part of the Java Media Framework (JMF), which specifies a programming interface for time-based media playback. It enables multimedia content in Java applets and applications, and allows for Web-based multimedia solutions that run in any Java compatible environment. However, these tools have to be modified in order to be suitable for multicast application and QoS management. The Java interfaces representing the abstract DAI primitives are specified in the DMIF (version 2) specification. We implemented these interfaces in the sender and receiver applications with the necessary modifications (mentioned above) for realizing the session and QoS management functions. The overall system is described in [14].

6. Conclusions

Session management for multimedia applications with multicasting is a complex task, especially when a large number of users are involved. It appears that the Delivery Multimedia Integration Framework (DMIF) for MPEG-4, which is presently extended for multicast applications, provides interesting session management functions for distributed multimedia applications in general, independently of the question whether MPEG-4 encoding is used.

We have shown how these DMIF functions can be used for session management of a teleteaching application including different QoS alternatives for the participating users. In such an application, the sender node of the teacher provides different stream variants (with different QoS attributes) for each logical multimedia stream. Each user participating as a student may then select one of these variants according to its QoS preferences.

The detailed analysis of the DMIF protocol in the multicasting context has identified certain generalizations that would be useful in order to make it more generally usable

for various distributed multimedia applications. This includes general means for distributing user information from the stream producers to the stream consumers, and some means for sending general user requests from a consumer to a particular producer.
A prototype implementation of a teleteaching application with QoS alternatives is in progress and includes a variant of the MPEG-4/DMIF for the management of the application session and associated QoS management. A demonstration should be available in October 1999.

Acknowledgements

The authors would like to thank Khalil El-Khatib and Qing Zhu for some interesting discussions. This work is supported by research grants from Communications and Information Technology Ontario (CITO) and Nortel-Networks.

References

[1] D. Hutchinson, G. Coulson, A. Campbell, G. Blair. "Quality of Service Management in Distributed Systems", Network and Distributed Systems Management, page 273-303, 1994.
[2] J. Gecsei, "Adaptation in Distributed Multimedia Systems". IEEE Multimedia, 1997.
[3] G. v. Bochmann, A. Hafid, "Some Principles for Quality of Service Management", Distributed Systems Engineering Journal, 4:16-27, 1997.
[4] A. Hafid, G. v. Bochmann, "Quality of Service Adaptation in Distributed Multimedia Applications", ACM Multimedia Multimedia Systems Journal, volume 6, issue 5, 1998.
[5] S. Fischer, A. Hafid, G. v. Bochmann, H. d. Meer, "Cooperative Quality of Service Manaement for Multimedia Applications", proceedings of the 4th IEEE Internatinal Conference on Multimedia Computing and Systems, Ottawa, Canada, june 1997, pp. 303-310.
[6] S. Fischer, M. v. Salem, G. v. Bochmann, "Application Design for Cooperative QoS Management", proceedings of the IFIP 5th International Workshop on QoS (IWQoS'97), New York, May 1997, pp 191-194.
[7] A. Hafid, S. Fischer, "A Multi-Agent Architecture for Cooperative Quality of Service Management", Proceedings of IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS'97), Montreal, Canada.
[8] S. Deering, "Host Extensions for IP Multicasting", IETF RFC 1112.
[9] A. Banerjea, M. Faloutsos, R. Pankaj, " Designing QoSMIC: A Quality of Service sensitive Multicast Internet protoCol", IETF Internet Draft: draft-ietf-idmr-qosmic-00.txt.
[10] J. Hou, H. Y. Tyan, B. Wang, "QoS Extension to CBT", IETF Internet Draft: draft-hou-cbt-qos-00.txt.
[11] "Overview of MPEG-4 Standard", (N2725), http://drogo.cselt.stet.it/mpeg/standards/mpeg-4/mpeg-4.htm
[12] "Information technology – very-low bit-rate audio-visual coding – Part 6: Delivery Multimedia Integration Framework (DMIF)", (N2506), http://drogo.cselt.stet.it/mpeg

[13] “Information technology – very-low bit-rate audio-visual coding – Part 6: Delivery Multimedia Integration Framework (DMIF)”, (N2720), http://drogo.cselt.stet.it/mpeg
[14] Z. Yang, “A tele-teaching application with quality of service management”, MSc thesis, University of Ottawa, expected 1999.