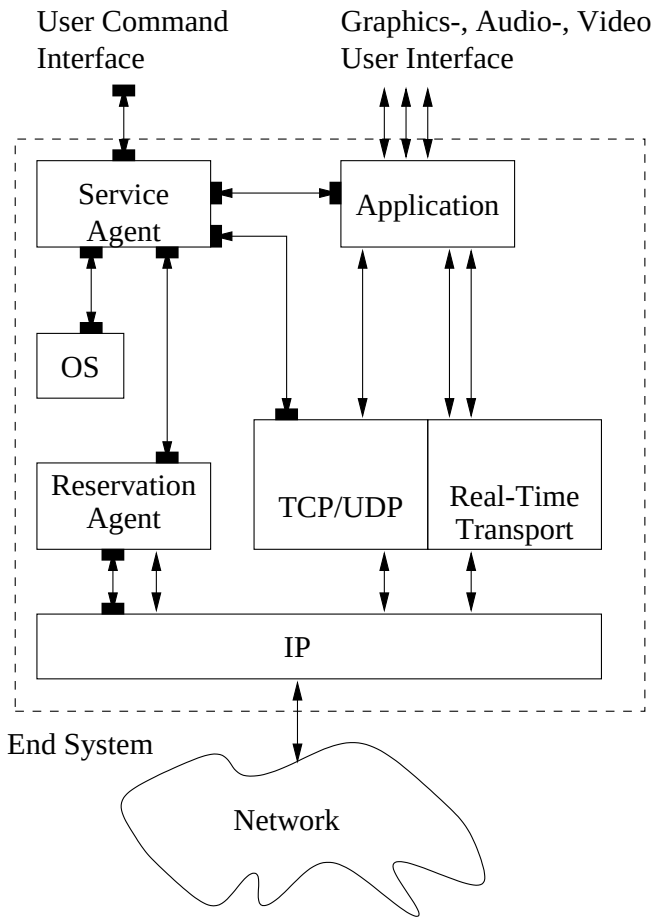
A Service Agent for Connection and QoS Management in Multimedia Systems

Norsk Regnesentral
ANVENDT DATAFORSKNING
Norwegian Computing Center/Applied Research and Development

NOTAT/NOTE



IMEDIA/05/98

Wolfgang Leister
Pål Spilling

Oslo
April 1998

NR ANVENDT DATAFORSKNING

Notat / Notat

Tittel/Title:
A Service Agent for Connection and
QoS Management in Multimedia Systems

Dato/Date: April
År/Year: 1998
Notat nr/:
Note no: IMEDIA/05/98

Forfatter/Author:
Wolfgang Leister, Pål Spilling (UNIK)

Sammendrag/Abstract:

This contribution focuses on connection and QoS management in continuous multimedia systems. We have studied connection and QoS management mechanisms for ATM and IPv6, and propose a framework for negotiation of QoS. A Service Agent will be responsible for QoS negotiation and configuration on an end-to-end basis. We present design characteristics of the architecture. A holistic view of QoS, and the role of the user is discussed.

Emneord/Keywords: Quality of Service, QoS negotiation, Service Agent, IPv6, ATM

Målgruppe/Target group: research institutions, NR

Tilgjengelighet/Availability: Open

Prosjektdata/Project data: ENNCE

Prosjektnr/Project no: 801001

Antall sider/No of pages: 10

Norsk Regnesentral/Norwegian Computing Center
Gaustadalléen 23, Postboks 114, N-0314 Oslo, Norway
Telefon (+47) 22 85 25 00, telefax (+47) 22 69 76 60

A Service Agent for Connection and QoS Management in Multimedia Systems

Wolfgang Leister
Norwegian Computing Center
wolfgang.leister@nr.no

Pål Spilling
Center for Technology at Kjeller
paal@unik.no

Abstract

This contribution focuses on connection and QoS management in continuous multimedia systems. We have studied connection and QoS management mechanisms for ATM and IPv6, and propose a framework for negotiation of QoS. A Service Agent will be responsible for QoS negotiation and configuration on an end-to-end basis. We present design characteristics of the architecture. A holistic view of QoS, and the role of the user is discussed.

Keywords
Quality of service and media scaling, Multimedia-specific intelligent agents, Resource management

1 Introduction

This document is an elaboration of a Quality of Service (QoS) architecture for multimedia systems with end-to-end focus, including the end-users and the user-to-system negotiation of QoS. The document presents a walk-through of point-to-point configuration across a pure ATM network and the Internet, with the aim of elucidating a common efficient QoS architecture.

Much multimedia research [1] has been devoted to study networking and middleware aspects and, how QoS [2, 3, 4, 5] could be managed on an end-to-end basis. In most cases end-to-end means application-to-application and does not include the ultimate end-users and the parts of the end-systems between the user and the applications. Often management of QoS is integrated within the applications. The process of negotiating QoS between the user and his end-system has not been dealt with in much detail.

The novel approach taken in this contribution[1] takes the responsibility for connection and QoS management out of the hands of the application process and places it under control of a dedicated Service Agent (SA). The SA is responsible for the total QoS negotiation process, between the user and his end-system, between the end-system and the network, between other system components in the end-system, and between the end-systems. The SA will also be responsible for the control of connection management. This relieves the

[1] This work has been funded in part by the Norwegian Science Foundation through its program "Basic Telecommunications Research" and project contract ENNCE.

application processes from the burden of dealing with QoS and connection management, and provides a clean and comfortable QoS architecture.

This paper is organized as follows: The next section briefly discusses current research on the handling of QoS in multimedia systems related to the present contribution, emphasizing the key difference with our work. The section thereafter takes a holistic view of the system, to place our work in the right context. The following two sections describe our end system architecture, and how QoS requirements are determined and managed throughout the total system. The last section provides some concluding remarks and views on future directions.

2 Background information

The OMEGA end-system architecture [2] has been designed to provide end-to-end QoS guarantees to distributed applications. The key component is the QoS Broker. The Broker negotiates subjective QoS requirements with the end user and translates these into objective application, transport and network requirements. The objective requirements are then used in negotiation processes with the application, transport, operating system and network, resulting in acceptance or rejection. Finally the Broker performs end-system to end-system negotiation. Hence the Broker combines media and network QoS management into a single entity. This is very similar to our design, but differs in the way our SA performs the QoS negotiation and QoS management in the end systems.

The multimedia enhanced transport system (METS) [3] supports multilayer coded flows in a multicast networking environment, where client workstations may have varying multimedia capabilities. METS offers a flexible QoS-configurable application programming interface, as an extension to the Berkeley socket API. The mechanisms to adapt to fluctuations in the network conditions are embedded in the transport system in a manner transparent to the applications. Hence QoS management is an integral part of the transport system, in contrast to our approach.

The IBM European Networking Center in Heidelberg has developed an end-system QoS architecture [4] built on the Internet Stream-II protocol that provides end-to-end guarantees with respect to QoS. The system supports QoS negotiation and QoS configuration of the system components, under control of an entity separated from the transport system. Hence their design is very similar to ours.

3 A holistic view of the system

Multimedia systems utilize continuous media to a greater extent, as increased bandwidth, greater computing capacity, and special purpose multi-media hardware become available. However, many of the resources being used are not unlimited. Especially continuous media applications decrease in quality in overload situations.

Most research papers on QoS architectures focus on the networking aspects, cf. [1]. This is natural as the most obvious and the most unpredictable influences come from the network. The general observation is that applications using for example a CDROM as a data source behave better. However, the user and his perception of quality should be more

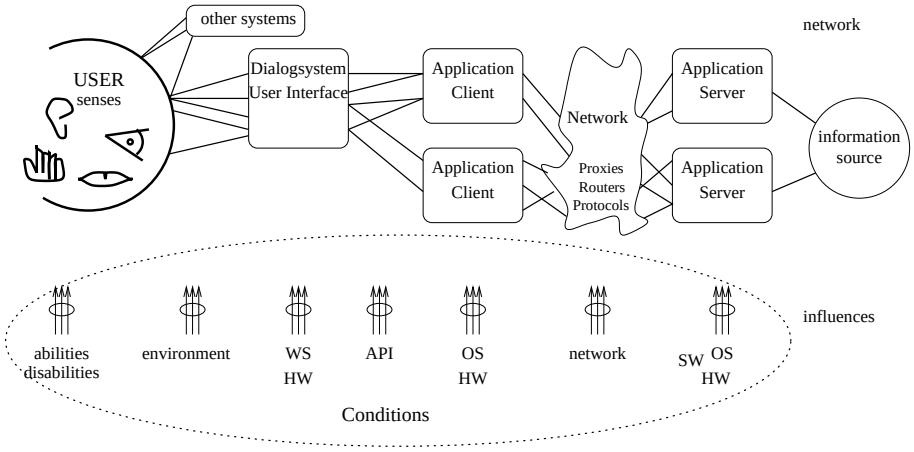


Figure 1: Holistic view of a generic multi media application

in focus. Therefore we take a more holistic approach by including a study of the flow of information from the source to the user.

In general, the information travels from an information source to the user, and undergoes several transformations on its way. Servers, proxies, clients, and networks are examples of elements in the stream between the source and the receiver, that have an impact on the quality of the data that the user perceives. Also the dialogue system, the environment, and even the user's abilities and disabilities should be taken into account. We illustrate in Figure 1 that each of these influences gives its contributions to the final result perceived by the user.

The elements in between the end points of the network are defined as nodes in an abstract networking model. Each node can be consceived of as having a common architecture, which is outlined in Figure 2: A data flow and a control flow are associated with each node. Besides the network layer and middleware, the application and the SA are the most important entities. The application interprets the incoming data flow, and sends its results to the next node in the network. Note the symmetry in the architecture: The concepts used for the network can also be used for rendering, the window system, or even the user.

This concept gives a more adaptive total system, as the SA negotiates QoS on behalf of the application, thus relieving the application from handling some of the adaptation issues. Following the definition from Gecsei [6], adaptation enlarges the area of well-behaved operation. This is exactly one of the most important issues for the SA.

4 End System QoS Architecture

In this section we discuss architectures of end-systems attached to ATM and IP networks. We will show that the tasks for the system components are quite similar. This makes it possible to design a common reference model.

Figure 3 shows the end-system configuration for a point-to-point setup with ATM as the transmission network and IP as the common end-to-end transmission mechanism. The

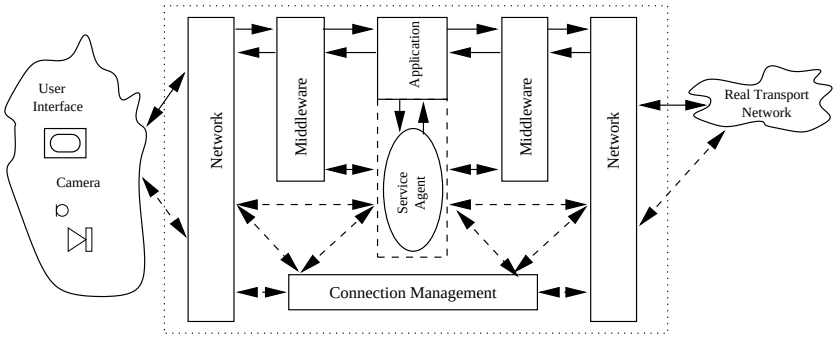


Figure 2: Symmetric architecture of a network node

components of the end-system are the multimedia application, the SA, the middleware consisting of the transport protocols on top of IP, the connection manager, the reservation manager, and the network interface consisting either of the ATM and AAL layers or an interface to the IP layer.

In Figure 3 we differentiate between operational interfaces and communications service interfaces (clean arrows). The operational interfaces are used for QoS management.

Connection Manager: This module builds on ATM and the appropriate AAL layer. Its purpose is to establish and release channels carrying user traffic, on behalf of the SA and the End-User. On request from the SA, it acquires a signalling circuit (a dynamically allocated virtual channel for signalling) from the network. Hence, the ATM layer does the multiplexing/demultiplexing functions between the signalling and user-traffic channels, using different virtual channels.

The Reservation Agent: The Reservation Agent makes use of RSVP (or other IP reservation protocols) to reserve resources along the route between source and destination, hence it has similar functionality as the Connection manager. RSVP does not permit QoS negotiation, only admission control and reservations, hence the outcome of the reservation process will be yes or no.

The Network Interface: The network interface consists of the ATM layer and the appropriate adaptation layer. Currently there is no possibility to multiplex/demultiplex at this level. Hence, user-related traffic will be multiplexed and demultiplexed at the IP level. The purpose of the adaptation layer is to provide a variable-length check-summed packet interface to the IP layer and a segmentation and reassembly interface to the ATM layer.

The Middleware: The middleware consists of the IP layer and a flexible transport layer containing TCP and UDP, and a configurable real-time transmission facility (e.g. DaCaPo [7]). As mentioned earlier, the IP layer interfaces with the adaptation layer, and supports a multiplexed packet service to the transport layer and the SA. The real-time transmission

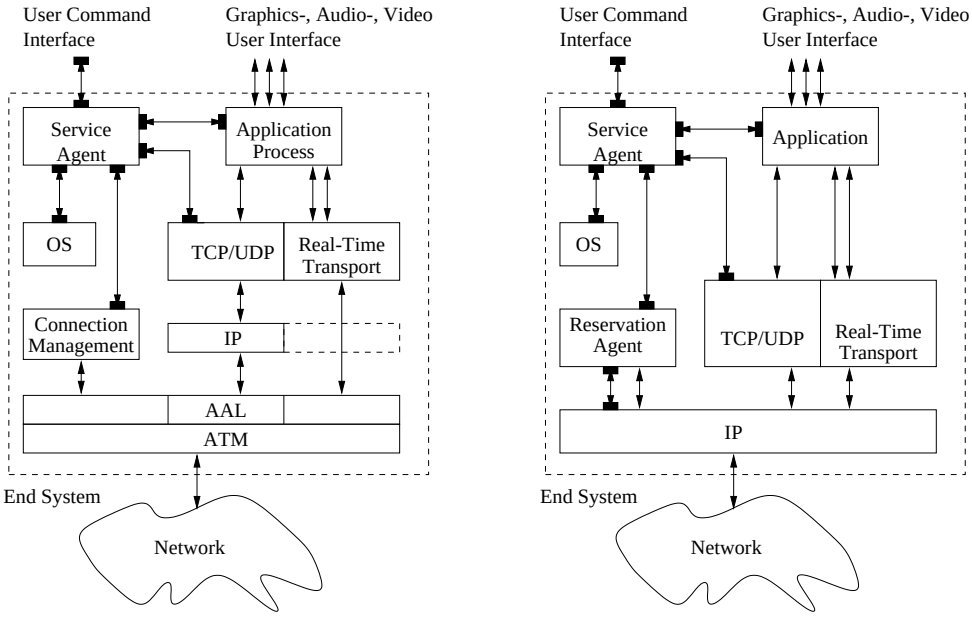


Figure 3: End System Architecture for ATM (left) and IP (right)

facility can be configured according to the QoS requirements set by the SA, and which previously has been negotiated by the end user and the SA.

The Application: The application is assumed to be a multimedia application making use of video, audio, and one ore more graphical input/output interfaces to the end user. These interfaces have to fulfill the QoS requirements negotiated with the user. The transmission facility provides one or more streams (real-time and asynchronous) from the application. In addition the application interfaces with the SA, among others to be configured according to the negotiated QoS requirements.

The Service Agent: The SA has a variety of tasks to perform:

- discern the QoS capabilities of the application media, the middleware, and the network,
- support the user in negotiating QoS requirements,
- configure the end system according to the negotiated QoS requirements,
- establish the network connection with the right QoS properties,
- negotiate QoS requirements with the other end system.

Hence, the SA needs to interface with the application as well as with the middleware, the IP layer and the Connection Manager. In order to configure the end system properly, the Service Agent also needs to interact with the operating system to allocate enough buffer capacity to the various streams and to set priorities and real-time parameters for thread scheduling.

5 QoS Negotiation and Connection Establishment

We briefly outline the steps taken in negotiating QoS requirements, both between the user and his end system, and between the two end systems. We still refer to Figure 3. For simplicity we assume that the application incorporates the media drivers with the corresponding coding algorithms.

5.1 User-to-system negotiation

Step 1. When launched, the application notifies the SA, and implicit triggers the QoS negotiation procedure.

Step 2. The SA sends a request to the application process to obtain a list of coding algorithms that are available for the various media types to be used. The SA will also probe the middleware to get to know its service support capabilities. Further, we assume that the SA has or can obtain knowledge of network capabilities, and the costs involved in using the network services. There may also be costs associated with the QoS requirements in the end system itself. In addition other elements, such as the window system, operating system, etc. may be probed for their capabilities. All information are recorded in appropriate profiles.

Step 3. The SA prompts the end user to state his QoS requirements, or uses a previously determined user profile. The SA now knows the users subjective desire and the corresponding real end system-specific parameters which influence QoS, like algorithms for video and audio, picture rate and resolution, etc. The actual parameters are presented on the user interface. All information are also recorded in a user profile.

In order to support the user in its decision, specific applications are run that play short example video sequences with different quality, and at the same time indicate the associated cost factor. The user makes his choice. A similar procedure is followed for the audio part and possibly also for the graphics part (screen resolution, etc.). This procedure is called *QoS by example* [6].

5.2 Connection Management

In the case where the end system is directly connected to an ATM network, the following connection management steps are taken:

Step 4. The SA now requests the Connection Manager to open up a network connection to the remote end system. If there is a bandwidth-on-demand functionality available in the ATM network, the initial connection capacity may be small, otherwise the connection is established with the fully required bandwidth.

5.3 Path Reservation Management

In the situation where the end system is connected to the Internet, step 4 is replaced by the following step:

Step 4: The Reservation Agent (RA) will use RSVP [8] or other IP reservation protocols to allocate the necessary resources along the flow between source and receiver. The outcome is either negative, meaning that there are not enough resources available, or positive. In the first situation, the initiating user is notified and may terminate the session or renegotiate the QoS requirements. In the latter situation, one proceeds to step 5.

5.4 End system to end system configuration

At this stage we do not go into details on how the remote application process is started up. Various scenarios are possible. For the current discussion we assume that the remote application process is running and listening on one or more transport level access points. Likewise we assume that the remote SA is listening on a well-known TCP port, and is prepared with all necessary profile information for the components of its domain.

Step 5. The local SA establishes a reliable connection to the remote SA, making use of the previously established network connection. The local and remote SAs start to negotiate QoS requirements, i.e. the local SA states its desired QoS requirements, possibly with lower bounds, which the remote SA can accept, modify or refuse. It is assumed that the remote SA has knowledge about the QoS support capabilities at its end, and hence can respond intelligently to the request from the local SA. If the agreed QoS requirements are not equal to the desired ones, the local SA has to request the acceptance from its end user. If not acceptable, the remote SA is notified, the TCP and ATM connections are terminated, and the remote SA goes back to listening. If accepted by the end user, the remote SA is notified about this, and the following steps will be performed.

Step 6. Assuming that the total negotiation process had a positive outcome, both the local and remote SAs now have complete knowledge of all objective QoS requirements for their respective end system components. The local SA notifies the remote SA about this fact, and the two SAs now proceed to invoke the appropriate configuration operations at their respective system components. When these operations are completed, the user to user communications can commence.

5.5 Service Agent Control Protocol

The SA acts as a separate process, and works independently through a separate user interface. The interaction with the SA proceeds through the Service Agent Control Protocol (SACP). SACP is designed to handle all required interactions between the different components of the end system. The exchanged messages are related to both network, application,

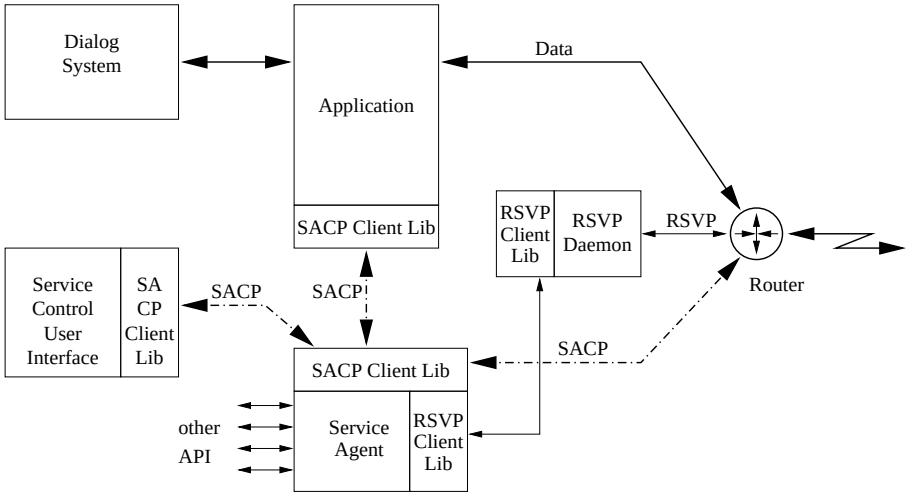


Figure 4: Management concept for IP with RSVP as reservation protocol

user interface, and the user's preferences. SACP provides a syntactical framework, and a protocol. It serves the following purposes:

- passing requests from the user interface to the SA,
- passing status information from the SA to the user interface,
- exchange information with the application,
- provide a syntax for profile information,
- and exchange information with SAs on remote systems.

SACP is not designed to replace RSVP [8] or another reservation protocol. SACP is not used for signalling or routing purposes, as these aspects are handled in other protocols. Actually, in the TCP/IP implementation, RSVP and the RSVP-demon will interface with the SA through RAPI [9]. The architecture is shown in Figure 4. Note, that one SA is supposed to serve several applications. One agent is responsible for an application group, typically a host.

The interactions with other system components, as the network-, reservation-, window system-, and operating system layers, is done with the respective API calls. Otherwise we would have to implement SACP interfaces for all these subsystems. We do not intend to integrate SACP into other parts of the system but the application level processes, and the user interface part of the QoS negotiation.

SACP will be implemented independent from binary formats, and the SACP messages are therefore defined as text. The protocol is on the application level, and effectiveness issues with regard to syntax analysis processing time or message length can be neglected.

Being an application level negotiation protocol SACP it deals also with coding formats, mapping issues, and other QoS parameters for non-networking parts, as for example colors, speed of graphics, and sound capabilities. For renegotiation and settlement of conflicting requests, or requests that cannot be fulfilled in total, the SA acts as arbitrator.

A user interface is provided, in order to keep the user informed about the system's state, and to make an interaction during the negotiation process with the user possible. This user interface will not be integrated in the SA directly, as different user interfaces might be appropriate. We propose an API where independent user interface programs can contact the SA by the SACP protocol. We also have in mind that the window manager can implement the user interface directly, and hence make possible to have an influence on size and arrangement of the programs. An example on such a user interface can be found in [10].

6 Conclusions

This contribution documents the initial results of our effort in studying distributed multimedia systems, and in developing a flexible and efficient architecture for QoS management. Our approach places the responsibility for the total QoS management in a dedicated Service Agent. It is shown how the SA has full control of the QoS negotiation process, in both the local and the remote end systems, and utilizing this capability when the network connection (in case of ATM) or the network path (in the internet case) is established.

Our study so far has concentrated on point-to-point connections across ATM and IP networks. In the continuation we shall enhance the scope of our study to incorporate the user dialogue system, multicasting, and how changing QoS conditions should be handled by the Service Agent to enable application to seamlessly adapt to such QoS changes.

References

[1] C. Aurrecoechea, A. Campbell, and L. Hauw. A Survey of QoS Architectures. *Multimedia Systems Journal*, may 1998(special issue on QoS Architecture), 1998.

[2] K. Nahrstedt and J. Smith. Design, Implementation, and Experiences of the OMEGA End-Point Architecture. *IEEE J. Selected Areas in Comms.*, sept., 1996.

[3] A. Campbell and G. Coulson. QoS Adaptive Transports: Delivering Scalable Media to the Desktop. *IEEE Network*, March/April:18–27, 1997.

[4] C. Vogt, L. Wolf, R. Herrtwich, and H. Wittig. HeiRAT – Quality of Service Management for Distributed Multimedia Systems. *Multimedia Systems Journal*, 1996.

[5] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *IETF*, RFC 1633, 1994.

[6] J. Gecsei. Adaptation in Distributed Multimedia Systems. *IEEE MultiMedia*, April–June:58–66, 1997.

[7] T. Plagemann. *A Framework for Dynamic Protocol Configuration*. PhD Thesis, Swiss Federal Institute of Technology, Zürich, 1994.

[8] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (rsvp) – version 1 functional specification. *IETF*, RFC 2205, 1997. proposed standard.

[9] R. Braden and D. Hoffman. RAPI – Internet Draft. *IETF*, draft-ietf-rsvp-rapi-00.ps, 1997.

[10] G. Michelitsch, M. Ott, D. Reininger, and G. Welling. QoS Aware Browsing in Distributed Multimdia Systems. In R. Steinmetz and L. Wolf, editors, *Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97)*, pages 430–439. Springer Verlag, 1997.