26

AN APPROACH TO QUALITY OF SERVICE MANAGEMENT FOR DISTRIBUTED MULTIMEDIA APPLICATIONS*

A. Hafid and G.v.Bochmann

Université de Montréal, Département d'Informatique et de Recherche Opérationnelle,
Montréal, H3C 3J7, Canada
{hafid,bochmann}@iro.umontreal.ca

Emerging high-speed networks and powerful end-systems give rise to a new class of applications such as video-on-demand and teleconferencing. Such applications are very demanding on Quality of Service (QoS) because of the isochronous nature of media they are using. To provide QoS support on an end-to-end basis, the need for the integration of network, transport, and operating services arises. Thus to support the new emerging services, an end-to-end QoS management is required. This paper deals with QoS management for multimedia applications by taking remote access to multimedia database as a case study. The example application is introduced and the entities involved in QoS provision are identified. QoS management activities are defined and a basic QoS management architecture for multimedia applications is presented. A general framework for QoS (re)negotiation is defined and an instantiation of this framework in the context of the example application is presented.

Key words: multimedia applications, quality of service, quality of service management

1. INTRODUCTION

Currently new emerging services, particularly distributed multimedia (MM) applications, e.g. video conferencing [1,2,3,4] and video-on-demand [5], based on broadband communications, are of great interest in industry, academic research and standardization. The introduction of these new services provides a new quality of communications. The new quality of distributed MM applications is characterized by handling continuous media and by managing various media at the same time. Different types of continuous media require different levels of quality of service (QoS) and they require guarantees for the level of service to be maintained. This implies stringent requirements for the communication systems and the end-systems to support the requirements of MM applications. Hence the new types of application need end-to-end *QoS management* to ensure that the requirements of the users are satisfied.

Several approaches and protocols have been proposed to support QoS at the communication level [6,7]. At the end-system level, system software operating systems, based on earliest deadline scheduling [8] enhanced with priority parameters [9] have been defined to manage the time con-

*. This work was supported by a grant from the Canadian Institute for Telecommunication Research (CITR), under the Networks of Centres of Excellence Program of the Canadian Government

K. Raymond et al. (eds.), *Open Distributed Processing*

straints for the new services. All those approaches provide schemes to support resource reservation in order to guarantee the requested QoS.

A key issue in QoS support is QoS negotiation activity which makes use of most of QoS management functions. However, to date, no framework has been defined to support end-to-end (user-to-user) QoS (re)negotiation for distributed multimedia applications. This paper proposes a basic architecture for QoS management on which a general framework for QoS (re)negotiation is defined.

The paper is structured as follows. Section 2 defines a QoS management architecture for remote access to MM database applications. Section 3 defines different QoS management activities and presents a basic QoS management architecture for MM applications. A general framework for QoS (re)negotiation is presented in Section 4. Section 5 concludes the paper.

2. REMOTE ACCESS TO MULTIMEDIA DATABASES

Remote access to multimedia (MM) database systems enables users to browse, search, request and display MM documents which are digitally stored in one or more high-capacity storage devices [10]. To support such an application, stringent requirements in terms of QoS must be provided by the underlying system.

In the following we give a short description of the components of a simplified architecture of a prototype system for remote access to multimedia databases which we implemented (Figure 1).

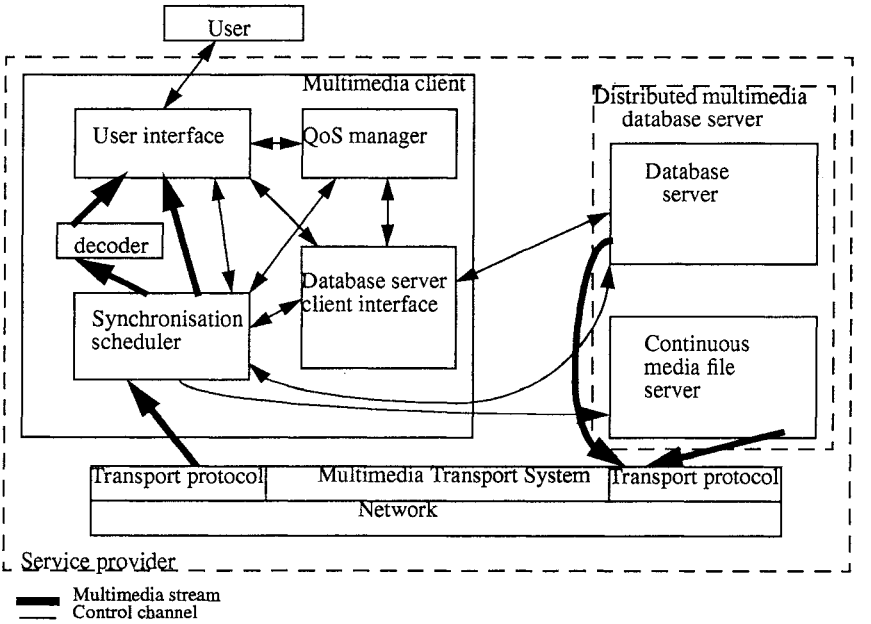

Figure 1. A Simplified Architecture of Remote Access to Multimedia Database Application

- *User interface* enables the user to select a document, to negotiate the QoS of the document to be

played, to display a selected document, and to renegotiate the QoS of the current document.
- *QoS manager* performs a QoS negotiation protocol to find an agreement which is compatible with the client's constraints, e.g. devices characteristics, the user's desires, and the database server constraints, e.g. access delay [11]. Typically the QoS manager must support the QoS negotiation, renegotiation, monitoring, adaptation, etc. by managing the configuration of the concerned objects.
- *Database server* provides reliable and coherent storage of multimedia objects as well as concurrent access to these objects and to their components. Information stored in the database is classified in to two categories: (1) MM information, (2) control information such as synchronization scenarios or QoS parameters.
- *Continuous media file server* provides support for the storage and retrieval of time sensitive continuous media, e.g. video and audio [12].
- *Synchronization scheduler* computes a time flow graph (TFG) and derives the object delivery schedule in order to compensate for variable network delays. The TFG is derived from the information stored in the database server [13].
- *MM transport system* is used for MM data that may have to be transferred between the physically distributed components of the system [14]. It is composed from high-speed transport protocol(s) and a high-speed underlying network.
- *Decoder* is used to decode the coded information, e.g. MPEG video, for presentation to the user [15].

In the context of remote access to MM database applications (Figure 1), we have identified six main objects that are involved in the QoS provision of a given activity: the user, the MM document, the file system, the transport objects, the network, the decoder, the synchronization object and the presentation device (Figure 2).

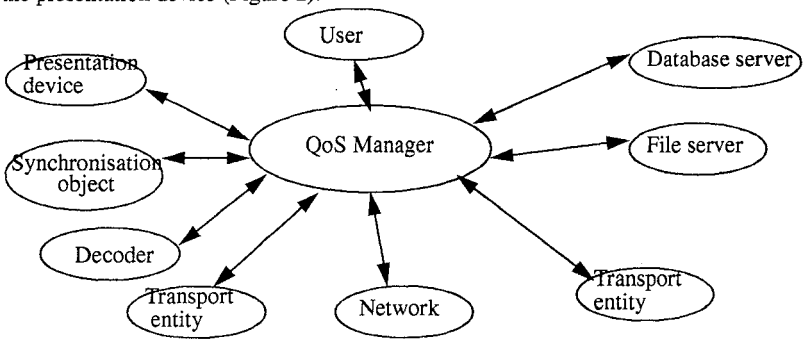

Figure 2. A QoS Management Architecture for Remote Access to Multimedia Database Application

In the following we will give a short description of each object identified, excepted the user, and define the main QoS parameters that must be supported by each entity.

(1) MM document: Multimedia object components of a given MM document, e.g. video and audio, may be stored in a number of file servers at different locations. To handle these documents, MM applications require the specification of document structures which integrate the different components of MM objects such as texts, graphics audio or video. These structures

must be defined in order to represent the control information used during consultation or display. During the consultation phase, information describing the document localization is used for query processing, while information concerning the media type, the size or the frame rate is used for the negotiation of the QoS. During the display phase, the system uses information describing the spatial and temporal relationships to guarantee the synchronization between components of the multimedia documents. More details concerning the integration of QoS parameters in our MM document structure can be found in [11]

(2) *File server:* The file server(s) provides access to MM documents with a requested quality of service. When several streams are being retrieved, guarantee of a desired QoS for a given stream becomes non trivial since all streams are competing for bandwidth, e.g. storage access, workstation bus, and CPU slots. Hence real-time server resources management is required [10]. This means that before accepting a new connection, the server must check if there are enough resources to support this connection without affecting the existing ones. Given the huge quantity of information to be accessed, e.g. 30 Mbits a second for uncompressed video, and the severe temporal guarantees required by continuous media, the file system(s) must guarantee a minimum throughput and a maximum delay to retrieve MM data [16].

(3) Transport entity: Protocol stack processing may introduce a non-desirable delay and jitter, e.g. through error control and segmentation/reassembling. Protocol architectural performance issues have been well studied in [17]. In [18] it was reported that parallel protocol processing is required for end-systems to keep with the improvements in network performance. Some new architectures for protocol development have been proposed, such as the horizontally oriented protocol (HOPS) architecture [19], or the high speed protocol development (HIPOD) architecture [20]. Guarantees for the speed of protocol processing are required to support the desired QoS. Thus, for a requested throughput, the delay and the jitter must be bounded at the transport service level.

(4) *Network:* The network must be able to transfer various media types with complex and variable QoS. The new characteristics of such a network can be summarized as follows: The support of constant bit-rate and variable bit-rate, the suitability of the transport for services with bit-rates varying from tens of bits to tens of millions of bits per second (e.g. full motion compressed video requires at least 2 Mb/s even using the most sophisticated compression schemes), the support of multicast and broadcast services, no distinction among the nature of information, e.g. video and audio are transmitted in the same way, and the support of a rich set of mechanisms to cope with bandwidth management. To support a requested QoS, the MM networks control the flow of new connections and the flow of the existing connections in order to guarantee a maximum transport delay and jitter, a minimum throughput, and a maximum loss rate [21].

(5) Decoder: Due to the huge memory size required to store voice and video data directly, compressions schemes are used to compress the data before storage or transport. To play a compressed multimedia document, it must be decompressed by a decoder before presentation to the end-user. Such an activity may introduce a non-desirable delay and jitter. To provide end-to-end QoS guarantees, the latter must be bounded.

(6) Synchronisation object: Temporal relationships may exist among the media of a given multimedia document. Each of the related media passes through several components, such as network links, and nodes, accumulating some delay at each component. Furthermore the paths used by the related media may be different. Thus the differences in the transit delay which are encountered by related streams may be important and possibly not acceptable to the applica-

tions. The aim of the synchronisation object is to smooth the jitter and skew introduced by the underlying system [14]. Such an entity must be able, by using some buffer management support, to restitute the temporal relationships between the related media of the document, typically by introducing some additional suitable buffering delay.

(7) *Presentation device:* After decompression, the multimedia data is sent to the presentation device, e.g. monitor and speaker, for presentation to the end-user. Such an activity may introduce variable jitter and delay depending on the used device. Thus the latter must be able to provide guarantees to support the requested delay and jitter.

3. QOS MANAGEMENT

Generally, there are three steps during the lifetime of a session, and particularly of a multimedia session: the establishment phase, the active phase and the clearing phase. Hence, to provide a requested service, the service provider has to establish a MM session with the desired QoS, to control the session QoS during the active phase and to clear the session when requested by the user or because of service provider problems, e.g. network congestion. In other words, to support multimedia applications requirements, QoS management functions are needed during the three phases (Figure 3). A brief description of these functions is given below.

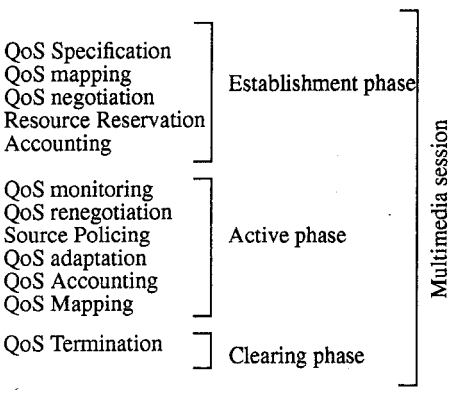

Figure 3. QoS Management Functions

(1) QoS specification and mapping: In the OSI Reference Model, the notion of QoS is defined as a set of qualities related to the provision of an (N)-service as perceived by an (N)-service-user [22]. Hence the QoS concept is associated with parameters to quantify the characteristics of the transfer of data between service access points of an OSI layer.

Mapping functions are required to map the QoS parameters of (N) layer to QoS parameters of (N-1) layer. The layer model of QoS for OSI is only concerned with those aspects of QoS requirements and the flows of QoS data across service boundaries and inside (N)-subsystems that are related to the operation of layer protocols.

More generally, for each component of the system, e.g. human user or the transport system, a set of QoS parameters must be defined [16]. Such parameters must be specified in a language understandable to the corresponding components. As an example, it is not acceptable to

present to the human user the parameter jitter, since it is not meaningful to him/her. Consequently mapping functions are required to translate the user QoS parameters values to the provider ones. Thus the service provider can use such QoS parameters values, results of the translation, to reserve the resources required to support the requested QoS.

(2) *QoS negotiation and resource reservation:* The QoS negotiation activity permits to find a system configuration which might support the requested QoS. Thus a QoS negotiation is required with all components of the configuration. The aim of this activity is to obtain a commitment from the service provider to support the requested QoS. Thus each involved entity is asked about the level of QoS it is able to support. At the end of the negotiation, if the 'sum' of the levels of QoS of each entity corresponds to the QoS required by the user, each entity must reserve the corresponding resources to support its level of QoS. In the case that the negotiation fails, a notification, preferably indicating the failure cause, is sent to the user. The latter has the choice to try another QoS negotiation, or simply to abandon. In the case that the negotiation succeeds (each entity committed to support a specific level of QoS), each entity must dedicate resources to support the requested QoS. In other words, each entity must reserve a certain number of CPU slots and buffers to meet its commitment.

(3) *QoS monitoring and source policing:* Monitoring mechanisms permit to perform a continuous measurement of the QoS actually provided by the underlying system. They have mainly two tasks: (1) To detect and notify any QoS violation (notification task): When the measured value of a QoS parameter does not meet the agreed one, a notification indicating the violation, and preferably the cause, is sent to the QoS manager, and (2) to store information (collection task): a description of the information to be selected when monitored and a description of any computations to be performed of the retained information are required. QoS monitoring requires QoS measurements, measurement procedures and methods must be defined; only measurable parameters must be considered, points where measurements can be affected must be identified, the periodicity of measurements must be determined [23].

Source policing is defined [24] as the set of actions taken by the network to monitor and control users' traffic to guarantee the QoS for any source which keeps within the parameter values agreed upon during the establishment phase, and hence to protect the network resources from user misbehaviour (malicious or not) which can affect the QoS of active connections. An obvious action to take when a user goes beyond the negotiated frame rate, is to discard data to meet the negotiated value. For example, if the negotiated frame rate is 20 frames/s and the user is sending data at 30 frames/s, the network may discard 10 frames/s.

(4) *QoS adaptation:* Overload, e.g. network congestion, will be a common occurrence in communication systems and workstations [9] because of the stringent requirements of the new emerging services. Thus QoS adaptation is required to react to such occurrences. QoS adaptation activity must be able to exhibit graceful degradation reacting adaptively to changes in the environment. Indeed, it may be more desirable to degrade the quality of the affected service (violation of its agreed QoS) rather than to abort it.

(5) *QoS renegotiation:* A renegotiation may be initiated by the user or the underlying system (e.g.communication system). The user initiated renegotiation allows a user to request a better quality, e.g. a user asks for colour quality while the current quality is black&white, or to reduce his requirements from the service provider to reduce, for example, the cost of the current session. On the other hand, renegotiation initiated by the underlying system is generally due to lack of resources (e.g. network congestion) and aims to reduce the provided quality to avoid a service interruption.

(6) QoS accounting: QoS accounting concerns the determination of the cost relative to a service requested by a user. Since distributed multimedia applications provide a vast range of QoS, QoS accounting is a complex activity. It must be based at least on the requested QoS to limit the user's greediness, e.g. if the cost does not depend on the requested QoS, all users will ask for the best QoS.

(7) QoS termination: When the service is terminated, notifications are sent to all entities involved in the QoS provision relative to this service to free the reserved resources.

Toward a general QoS management architecture

We can generalize the QoS management architecture for remote access to MM database application (Figure 2) to more general MM applications. Given an application, the following steps are required to build a framework that supports QoS management functions such as QoS (re)negotiation: (1) to identify the entities involved in the QoS provision relative to this application, and (2) to define their characteristics and functionalities, such as the attributes and the methods supported. Our QoS management framework consists of a QoS manager and a collection of entities required to support the requested QoS. When a QoS manager receives a request from the user, it determines an optimal configuration, whenever possible, that should support the requested QoS and sends requests to each entity to reserve the necessary resources to support the required QoS. Each entity sends a response to the QoS manager. The response may be yes, no, or an alternate proposal. If all parties respond yes, the configuration is established and the QoS management activities of the active phase, such as QoS monitoring and adaptation, can start. If one or more entities respond no, the QoS manager rejects the user request. If any other proposal was sent by one of the entities, the QoS manager will try to find another configuration. Close cooperation between the QoS manager and network managers is required to optimally support the QoS management functions.

4. A GENERAL FRAMEWORK FOR QOS NEGOTIATION

Distributed MM applications provide a vast range of QoS. Hence a user must be able to specify a desired QoS depending on his needs, his end-system characteristics and his financial capacity. In other terms he must be able to negotiate the QoS with the service provider. Furthermore it is not acceptable that the negotiated QoS at the establishment phase applies for the whole session lifetime; a user must be able to decrease or to increase the QoS currently provided, e.g. to receive a less expensive service or a better quality. Consequently a QoS negotiation and renegotiation protocol is required. This protocol must be designed on an end-to-end basis and must not be restricted to the communication systems, as is the case for most existing QoS negotiation protocols [25, 26, 27, 28]. Based on the QoS management framework defined above, we present a general framework for QoS negotiation. For sake of clarity, we will instantiate the framework by considering the application of remote access to MM databases.

Facilities to support QoS (re)negotiation

Upon the receipt of the user request, e.g. user_QoS request, for a specific service with a desired QoS, the QoS manager makes use of the following facilities to answer the user request.

(1) The information facility permits to collect information pertinent for a given user request. Such information concerns QoS parameters relative to the entities involved in providing the QoS. A management information database (MIB) [29] is a candidate to store such information. Two types of information may be considered: (1) static information which does not depend on the

actual system load, e.g. maximum packet size supported by a given network or a compression scheme supported by an end-system, and (2) dynamic information which changes with the system state, e.g. available bandwidth.

The QoS manager communicates with the MIB(s) to get QoS information within the system using the *QoS_Information* service (confirmed).

In the context of our case study, the information collected concerns only the QoS related to the selected multimedia document, the database server and the transport user system. Such information is stored in the database server and concerns the document media-type, format, and size. Additional information are available such as the access time and copyright cost [16]. When the user asks to play a specific document, the client gets the required information from the database server.

(2) *The configuration selection facility* permits to determine an acceptable configuration that supports the requested QoS. Such a configuration is determined based on the information obtained from the information facility. Optimization schemes may be used to find the optimal configuration. Some of the tasks to be performed to construct the configuration are:

(a) Choice of compression schemes: The compression scheme to be used must satisfy the required delay and reliability.

(b) Choice of networks: Such a choice may be based on the cost, resources availability, etc.

(c) Choice of transport protocols: Such choice may be based on error control functions, performance, etc.

(d) Choice of resource locations: If the user's end-system does not have enough resources, e.g. it does not support the compression scheme with which the selected document is stored, then another machine with available resources may be used.

In the context of our case study, this facility permits to get the final QoS parameters values taking into account the user QoS, the local end-system characteristics and the QoS information obtained from the server using the information facility. However the algorithm used to find a configuration that supports the QoS is very simple [11], given that we are using mainly static QoS parameters.

(3) *The resources reservation facility* permits to gain a commitment from the components of the configuration, identified by the configuration selection facility, indicating their agreement to support the requested QoS. Each component must formally commit resources for this purpose. Mapping schemes to transform QoS parameters values to resources, e.g. CPU slots and buffers, are used to provide this facility.

The QoS manager asks the components of the selected configuration to reserve the resources to support the requested QoS using *QoS_reserve* service (confirmed).

Several solutions have been proposed to support guaranteed performance communication. They all adopt a connection-oriented and reservation-oriented approach [6, 7, 30, 31]. At the system level, several systems have been developed to support MM applications requirements such as TRDM [9], SUN OS 5.0 [32], real-time (RT) MACH [33], DASH [34] and extended CHORUS [35]. Most of them use earliest deadline [8] as the scheduling policy.

(4) *The data transfer facility* enables the control entities, e.g. QoS manager, to communicate with the different system components, e.g. server and MIB. Such a facility is used by almost all the negotiation facilities. In the context of our prototype we used remote procedure calls (RPC) to realize this facility.

(5) *The monitoring facility* enables to detect QoS violation because of resources shortage of one or more components involved in the QoS provision. When a QoS violation is detected an indi-

cation is sent to the QoS manager. Depending on the established policies, the session is aborted, the violation is ignored, or a renegotiation is initiated.
The QoS manager sends a QoS violation notification to the user using the *QoS_violation* service (non confirmed)

(6) The renegotiation facility supports a QoS renegotiation initiated by the user or by the underlying system (e.g.communication system). The user initiated renegotiation allows a user to request a better quality or to reduce his requirements, for example, to reduce the cost of the current session. On the other hand, the underlying system initiated renegotiation is generally due to lack of resources and aims to reduce the provided quality to avoid a service interruption. The renegotiation facility uses all the other facilities to provide its service.
The user can renegotiate, with the QoS manager, the current QoS at any moment during the session lifetime using *QoS_renegotiate* service (confirmed).

(7) The termination facility enables a user to terminate a session. All the resources reserved at the components of the configuration are deallocated.
At any moment the QoS manager can terminate the session using the *QoS_terminate* service (non confirmed).

States transitions for the QoS manager

Based on the facilities defined above, the QoS manager always goes through the states shown in Figure 4. The model shown in Figure 4 is a general and abstract model which may be refined depending on the system architecture and the management protocols used. A refinement of this model for the case of remote access to MM database application can be found in [11].

The transitions between the six states of Figure 4 are based on service primitives. Initially the QoS manager is in the *Idle* state. When the user asks for a specific service with a desired QoS (user_QoS_request), e.g. play a specific multimedia document, the QoS manager sends the QoS_information_request primitive(s) to the appropriate MIB(s) and moves to the *waiting to get information* state (transition T1).

When the QoS manager gets the information, related to QoS, from the MIB(s) (QoS_information_confirm) it enters the *selecting configuration* state (T2). At the end of a successful configuration selection the QoS manager asks for a reservation from the involved entities (QoS_resource_request), e.g. database server and video file servers, to support the QoS required by the selected configuration and moves to the *resource reservation* state (T3). In the case where no acceptable configuration is found within a specific interval, the QoS manager moves to the *idle* state (T5).

After receiving resource reservation commitment from the different entities (QoS_reserve_confirm), the QoS manager enters the *active* state (T4). In this state a renegotiation event may be initiated by the user, or the underlying system if a QoS violation occurs. In the case where one or more entities do not commit themselves to support the required QoS, the QoS manager reacts by a transition to the *idle* state (T6).

Depending on the policy used to deal with QoS violation, on receipt of some QoS violation (QoS_violation_indication) the QoS manager moves to *waiting for user response* if the "renegotiation" policy is used (T7), moves to *idle* if the "abort" policy is used (T8), or remains in the same state if the "ignore" policy is used (T9). The QoS manager reacts to the receipt of a renegotiation request from the user (renegotiate_QoS_indication) by a transition to *configuration selection* state (T10).

On the receipt of a *terminate_session_indication ()* primitive from the user, in any state, the

QoS manager enters the *idle* state (T11). When the QoS manager is in *waiting to get information, selecting configuration, resource reservation* or *waiting for user response* states, a timer T is initialized and started. If no transition occurs from the current state before the timer expires, the QoS manager moves (T12, T13, T14) to the *idle* state or (T15) to the *active* state.

The parameters relative to each primitive, e.g. QoS_information request, have been defined for remote access to database application in [16, 11]

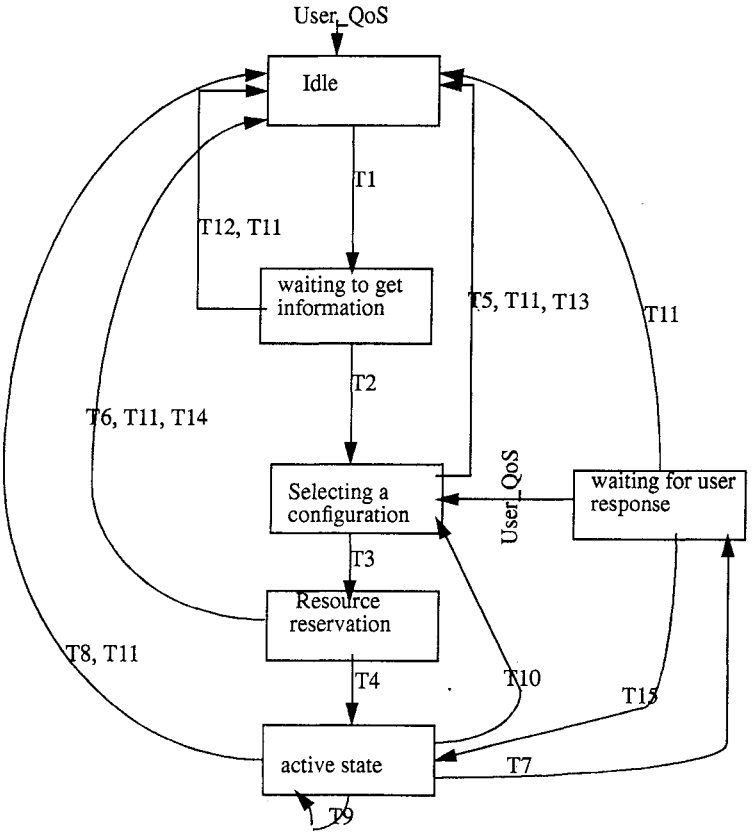

Figure 4. State Transitions for QoS (Re)Negotiation

5. CONCLUSION

The paper describes a remote access to MM database application. A QoS management architecture for this type of applications is defined. The QoS management activities are described and a QoS management architecture for MM applications is presented. A general framework for QoS negotiation is presented. An instantiation of this framework has been built for remote database

access applications in order to show its feasibility by an example. Currently, we are working on elaborating the framework for QoS negotiation, particularly the information and configuration selection facilities. Thus we are developing a performance model that enables us to find an optimal configuration for a requested service. We also study the required MIB(s) and their distribution. More generally we are aiming at specifying QoS management functions such as negotiation, monitoring and adaptation, for remote access to MM database and other applications.

ACHNOWLEDGEMENT: We would like to thank R.Velthuys from Waterloo University for fruitful discussions on a first draft of the paper, and A.Ezust from University of Montreal for helping in the presentation of the last version of the paper.

REFERENCES

1. G.Dermler, T.Gutekunst, B.Plattner, E.Ostrowski, F.Ruge and M.Weber, Constructing a Distributed Multimedia Joint Viewing and Tele-Operation Service for Heterogeneous Workstation Environments, IEEE Workshop on Future Trends of Distributed Computing Systems, Lisboa, 93
2. T.Gutekunst and B.Plattner, Sharing Multimedia Among Heterogeneous Workstations, 2 Inter. Confer. on Broadband Islands, Greece, 93
3. T.Gutekunst, T.Schmidt, G.Schulze, J.Schweitzer and M.Weber, A Distributed Multimedia Joint Viewing and Tele-Operation Service for Heterogeneous Workstation Environments, Workshop on Distributed Multimedia Systems, Stuttgart, 93
4. IEEE Communications magazine, Special Issue: Multimedia communications, May 1992
5. A.Rowe and B.Smith, A continuous Media Player, 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego 1992
6. D.Ferrari, A.Banerjea and H.Zhang, Network Support for Multimedia, Technical report 92-072, International Computer Science Institute, Berkeley, November 1992
7. C.Topolcic, Experimental Internet Stream Protocol, Version 2 (ST-II), Internet RFC 1190, 90
8. C.Liu and J.Layland, Scheduling Algorithms of Multiprogramming in a Hard Real-Time Environment, Journal ACM, Volume 20, Number 1, 1973
9. J.Hanko, E.Kuerner, D.Northcut and G.Wall, Workstation Support for Time-Critical Applications, Second International Workshop Heidelberg, Germany, November 1991
10. V.Rangan, H.Vin and S.Ramanathan, Designing an on-demand multimedia service, IEEE Communications Magazine, July 1992
11. A.Hafid, A.Bibal, G.Bochmann, T.Burdin, R.Dssouli, J.Gecsei, B.Kerherve and Q.Vu, On News-on-Demand Service Implementation, Technical Report, Université de Montréal, Montréal, Canada
12. P.Lougher and D.Sheperd, The Design of a Storage Server for Continuous Media, The Computer Journal, February 1993
13. L.Lamont, Component Interactions and Messaging System for Multimedia Synchronisation, Technical Report, MCRLab University of Ottawa, May 1994and Hypermedia Information
14. B.Metzler, I.Miloucheva and K.Rebensburg, Multimedia Communication Platform: Specification of the Broadband Transport Protocol XTPX, CIO, RACE Project 2060, 60/TUB/CIO/DS/A/002/b2, 92
15. D.le Gall, A video Compression Standard for Multimedia Applications, Communications of the ACM, April 1991

16. B.Kerherve, A.Vogel, G.Bochmann, R.Dssouli, J.Gecsei and A.Hafid, On Distributed Multimedia Presentational Applications: Functional and Computational Architecture and QoS negotiation, High Speed Networks Conference, Vancouver, Canada, August 1994
17. D.Feldmeir, A framework of Architectural Concepts for High Speed Communications Systems, IEEE JSAC, May 1993
18. G.Neufeld, M.Ito, M.Goldberg, M.McCutcheon and S.Ritchie, Parallel Host Interface For an ATM Network, IEEE Network Magazine, July 1993
19. Z.Haas, A Communication Architecture for High Speed Networking, IEEE Network Magazine, January 1991.
20. A.Krishnakumar, J.Knener and A.Shaw, HIPOD: An Architecture for High Speed Protocol Implementation, High Performance Networking IV, 1993
21. D.Towsley, Providing Quality of Service Packet Switched Networks, Joint Tutorial Papers of Performance 93 and Sigmetrics 93, Lecture Notes in Computer Science
22. ISO/IEC JTC1/SC21, Quality of Service Framework, Project JTC1.21.57, Working Draft #3
23. A.Hafid, J. de Meer and A.Rennoch, On JVTOS QoS Experiments, Technical Report, GMD-FOKUS, Berlin, Germany, 1994
24. CCITT I.311, B-ISDN General Network Aspects, 1990
25. ISO/IEC JTC1/SC6/WG4 N831, Multimedia Communication Platform: Specification of the Enhanced Broadband Transport Service, 1993
26. ISO/IEC JTC1/SC6, High Speed Transport Service Definition (HSTS), Preliminary Draft, 6 July, 1992
27. A.Danthine, Y.Baguette, G.Leduc and L.Leonard, The OSI 95 Connection-Mode Transport Service: The enhanced QoS, High Performance Networking IV, Liege, Belgium, 1993
28. I.Miloucheva, XTP and ST-II Protocol Facilities for Providing the QoS Parameters of Connection-Mode Transport Services, Research Note TUB-PRZ-W-1029, Berlin, 92
29. M.Rose, The Simple Book: An Introduction to Internet Management, Prentice Hall, 1994
30. A.Cramer, M.Farber, B.McKellar and R.Steinmetz, Experiences with the Heidelberg Multimedia Communication System: Multicast, Rate enforcement and Performance, High Networking Performance IV, 1993
31. T.LaPorta and M.Schwartz, The Multistream Protocol: A highly Flexible High Speed Transport Protocol, IEEE JSAC, 1993
32. S.Khanna, M.Sebree and J.Zolonowsky, Real-Time Scheduling in SUN OS 5.0, USENIX Winter Conference, SAN Francisco, January 1992
33. H.Tokuda, T.Nakajima and P.Rao, Real-Time Mach: Towards a Predictable Real-time Systems, USENIX Association Mach Workshop, October 1990
34. D.P.Anderson, Support for Continuous Media in the DASH Systems, Proceedings of the Tenth International Conference on Distributed Computing Systems, 1990
35. G.Coulson, G.Blair, B.Stefani, F.Horn and L.Hazard, Supporting the Real-Time Requirements of Continuous Media in Open Distributed Processing, Technical Report 1993, Lancaster University, UK
36. CCITT recommendation G.106, 1984