# A Hybrid QoS Management Scheme for Distributed Multimedia Applications

Anne Fladenmuller[a], Aruna Seneviratne[b], Eric Horlait[a]

[a] *Laboratoire MASI, Université Pierre et Marie Curie*
*4 place Jussieu, 75252 Paris Cedex 05, France*
[b] *School of Electrical Engineering, University of Technology, Sydney*
*P.O.Box 123, Broadway, NSW 2007, Australia*
*e-mail: {fladenmuller,horlait}@masi.ibp.fr or aps@ee.uts.edu.au*

**Abstract**

*This article proposes a model for management of multimedia applications. As management is different, depending on whether resources can be reserved, we study the 2 approaches. We show that a hybrid scheme which uses some form of resource reservation in conjunction with system level support for adaptive application is the most appropriate for this type of QoS management. We also consider the needs of interoperability for distributed multimedia applications by using some generic QoS specification.*

**Keywords:** Multimedia, Quality of Service Management, Adaptive Applications

## 1 Introduction

It is now accepted that Quality of Service (QoS) is a good way of defining the requirements of multimedia applications. Currently there appears to be two approaches to managing the QoS of these applications. The first approach uses reservations. Before running an application, it interacts with a QoS Manager which checks the availability of resources to provide a minimum QoS level. If the resources are available, the QoS manager reserves these resources and launches the application. Thus, with this scheme the QoS manager needs to track the usage and availability of resources. In addition, it needs to map the application's QoS requirements into the available system resources. A number of proposals for schemes based on this model has been proposed in the literature [1, 2, 3].

As the resources are reserved in the above scheme, if changes in their availability occurs[1], the QoS manager need to renegotiate the QoS levels. This can be an involved and time consuming exercise. This can be overcome by incorporating the ability to detect or infer changes in the operating environment, and the changes that need to take place in response to these variations, into the application. A number of implementations which are based on this principle have also been reported in the literature [4, 5, 6]

Proponents of the first scheme [7] argue that adaptive applications are unstable, complex and cannot completely deal with fluctuations in available resources. The proponents of the second schemes [8] argue that adaptive applications provide the highest fidelity, provides graceful degradation and are robust. We believe that a hybrid architecture in which support is provided for centralized management of QoS coupled with application based adpativity can overcome the limitations of both the above scheme.

In this paper we provide some preliminary results of our work in developing such an architecture. First we present the general structure of a QoS Manager based management model which can be used for resource reservation. Then in section 3, we show this model can be used together with specification language such as ASN.1 to provide generic method for QoS specification in this type of system. We present some results which show that application based adpativity alone is not sufficient, and justifies our choice of the hybrid method. And finally we present some initial results which shows the potential of the above hybrid schemes.

## 2 End-to-End QoS Management

As mentioned earlier, there are a number of reservation based schemes that have been reported in the literature. This section provides an overview of this type of QoS management architectures.

[1] this will be common in mobile environments

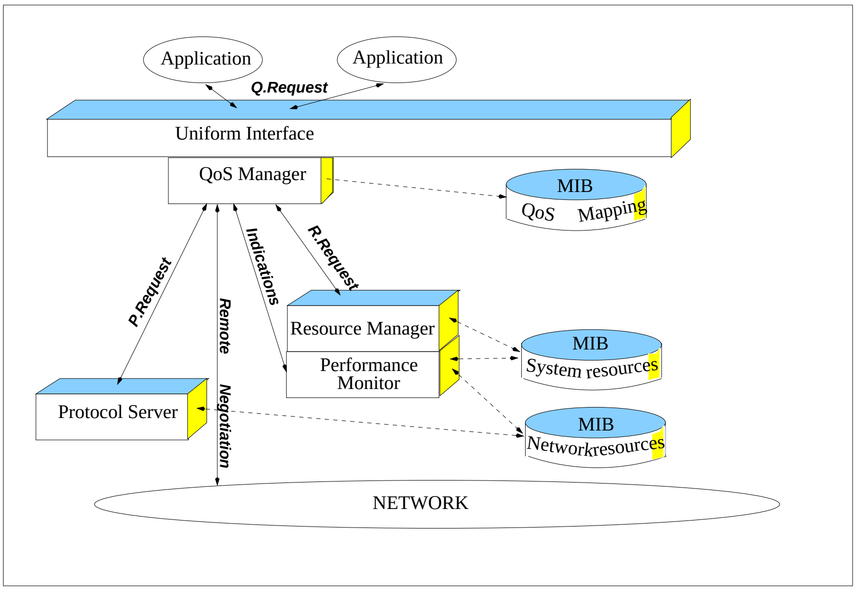


Figure 1: Implementation scheme for an end-to-end management

## 2.1 Architecture

To provide the necessary support for adaptive applications, firstly it is necessary to have a standardized application programming interface (API) [3, 9, 10] through which the applications can carry out QoS management functions. Secondly, QoS management systems must provide a management entity which will handle the translation of the application's QoS requirements into the host system resources (QoS Manager), and a another management entity which will allocate the system resources to applications and police their usage (Resource Manager). Finally these management entities will need to interact with the communication subsystem as the communication systems requirements need to be taken into account when the overall QoS requirements are assessed. An overview of the structure of this system is given in Figure 1.

## 2.2 Operations

*Application QoS Requirement Specification*

The application requirements at this level will be described in very broad terms as described in [2, 11]. At run time, the requirements of a given class will be specified in terms of desirable and minimum acceptable levels.

*Negotiation phase*

Once the QoS requirements have been specified by the application, the system enters the negotiation phase. The negotiation phase either accepts the request for execution and issues a QoS contract between a user and the system, or refuses the request.

To do this, the system uses the QoS specifications of the application and calculates required system resources to support the desirable and minimum levels of QoS requested by the application. If there are not enough resources to run the application at the desirable QoS level, the QoS manager checks whether the application can be supported at the minimum acceptable QoS level or whether some lower priority application can be degraded to accommodate the request. If the application cannot be supported at the minimum specified level of QoS the request is refused. This procedure is referred as the Admission Control.

Once it is established that a QoS contract can be entered into at the originating station, the QoS Manager will contact the remote QoS manager and the remote QoS manager will attempt to negotiate a QoS contract with its host system. The remote manager will only attempt to obtain a contract at the QoS level specified by its peer QoS manager. This will prevent protracted negotiation phase.

*Monitoring*

The monitoring function involves checking if each application has at its disposal the right amount of resources so that it can be executed according to the quality previously defined. If not, the application has to be notified of changes in conditions. Therefore, as soon as a modification in resource allocation is detected that will affect the execution of an application, an alarm is sent to the QoS manager.

Such a problem can arise when a bursty application is running. It can suddenly need and use more resources than its average. For example, a live video application running concurrently with a bursty application could suddenly not have the required resources to display high resolution frames as could have been previously negotiated. In such a case, the manager would notify the application of this degradation of service and the application would react according to its degradation scheme. This can consist of degrading to black and white video frames rather than affecting the speed of the pictures displayed.

The termination of an application implies deallocation of resources. They can then be shared among other applications, especially the ones that have suffered a degradation, depending on their priority.

*Renegotiation phase*

As we have seen previously, a renegotiation phase occurs when an application has to degrade or upgrade its quality of service. Once the new QoS characteristics of the application have been determined, the manager allocates the resources, modifies the threshold concerning this application, updates the resource data bases and checks whether this modification has an impact on the management of the others applications.

*Termination phase*

When an application has finished its execution, the QoS manager is informed by the application. Resources are freed, the information databases are updated and the data concerning this application are removed from the manager.

## 2.3 Implementation

There appears to be no full implementations of this type of QoS Management scheme, as the various components of the systems are still being developed. However, we have investigated how a given application can be extended to utilize such a QoS Management scheme [9]. For completeness, a brief description of this implementation is given below. Details of the scheme can be found in [9].

*QoS Managed Applications*

To decide whether an application can be run, the QoS manager must determine the resources required by the application as well as the communication subsystem. The QoS requirements of different applications will be diverse. To cope with this diversity each application sends a request for a QoS negotiation or renegotiation. The interface between applications and the QoS manager is done with some **Q.Request** [9]. The QoS Manager sends a notification to indicate the result of a **Q.Request**, and an alarm to notify a modification in the resource allocation. When an application receives an alarm, it applies a degradation or upgrading script to modify its quality.

*QoS Manager*

The role of the QoS manager consists first of mapping applications requirements into a set a QoS parameters of a lower level. As users generally use the same applications without varying their environment, the QoS Manager keeps track of the use of these applications. Information concerning QoS parameters for an application can be stored in a database to facilitate the mapping process. This type of caching will speed up the negotiation or renegotiation phases.

The QoS manager sends these new parameters to the protocol server in a **P.Request** and to the resource manager in a **R.Request**. The protocol server will determine the best communication protocol for each of the application streams, and the resource manager will request the required resources.

If the system has enough resources locally, a remote negotiation is started with the remote sites. The multicast negotiation is still open research and will not be considered in our model.

*Resource Manager*

The resource manager is in charge of reserving the host system resources. It uses information given by the QoS manager, such as the minimum and acceptable QoS level for each stream and their periodic or aperiodic nature, to determine the amount of resources required by each stream of the application.

If the desirable level can not be supported, a schedulability analysis is carried out with the minimum level for all active connections. A notification of the result is provided to the QoS manager to indicate whether under these conditions the request can be supported. The exchange of information between the QoS manager and the resource manager is done for each stream with a **R.Request**.

*Protocol Server*

The role of the protocol server is to provide the appropriate standardized protocol implementations to application streams. The choice of the most suitable protocol stack is determined by the information sent by the QoS manager. For example, if an error control is required, it will determine whether it should be based on an error correction or automatic repeat request scheme. An implementation of a protocol server is given in [12]. In this paper, it is assumed that the application will choose from a set of predefined protocols such as RTP, TCP, and UDP. The interaction between the QoS manager and the protocol server is done with the **P.Request**.

*Performance Monitor*

The performance monitor polls the data bases concerning the system and network resources and sends an alarm to the QoS manager when the value of any resource can affect the QoS of an application. The interaction between the performance monitor and the QoS manager is performed by the use of **Indication** messages.

# 3 Abstractions for the management

## 3.1 QoS Specification and abstractions

Description of application and user needs, and specification of what the system offers in terms of available resources, can be done by using a set of QoS parameters. This has been done at a transport layer [13], or by classifying applications into classes according to their time constraints or their reliable or unreliable nature [11, 14]. However these specifications do not address QoS specification at the application level. Therefore we present in this section a generic method of specifying QoS using a specification language, namely ASN.1.

Abstract Syntax notation (ASN.1) [15] is an formal notation for specifying abstract syntaxes. It has been standardized internationally as a language for describing structured information. Given an ASN.1 description a representation can be derived mechanically by applying a set of encoding rules, which is also standardized. ASN.1 has been conceived to allow the designer to specify some aspects of distributed applications.

The ASN.1 language has also been chosen to specify some management information bases (MIB) [16]. The MIBs defined in our model contain information about the system and network resources available. The use of a MIB manager such as SNMP (Simple Network Management Protocol) [17] allows the manipulation of the MIBs, to obtain statistics and send alarms.

## 3.2 definition of MIBs

We have defined three types of MIBs to help the management of multimedia applications.

**MIB for the management of applications**

An approach given in [18] defines some Network Service Monitoring (NSM) MIB to provide information about the currently running applications. The generation of these MIBs is done by a new language: Quality of Service Assurance Language (QuAL), where QuAL specifications are compiled into run time components that monitor the actual QoS delivered.

An extension of such an approach can be used to keep track of information concerning multimedia applications already running on a computer. The calculation task of the QoS manager would then be reduced to an application cache management. In fact, as users usually run the same application, statistics about the applications needs can easily be obtained. The use of such a method would improve the admission control, above all when burst application are involved but would not overload the system, because this does not need to be done in real time.

**MIB for the mapping of QoS parameters**

Constant development of new multimedia applications makes it difficult for any envisaged QoS Manager to have a knowledge of every applications. However, the list of coding types is fixed by agreed standards. For example, the main audio coding types are [19] G777, G721, G722, G723 and MPEG audio. In the same way, we can consider that there is a finite set of coding and/or compression standards will be in place, such. group3, JBIG and JPEG for still pictures, and H261, MPEG-1 and 2 for video. As these standards become available, system Management Information Bases (MIBs) can be developed which will store resource requirements to support a given media type within the host system. The role of the QoS management entity will be to translate the application's requirements into system resource requirements, including the communication system, using an appropriate MIB and request these resources from the resource manager.

The type of abstraction is used by the QoS manager to determine the system and network resources for each stream. Each entry in the QoS Mapping MIB defines the profile of a stream.

The extract of the MIB of the Figure 2 gives an example of the specification for an audio stream. The type (audio) and the name (G722) of the stream are transmitted by the application in a Q.Request to identify the entry in the MIB.

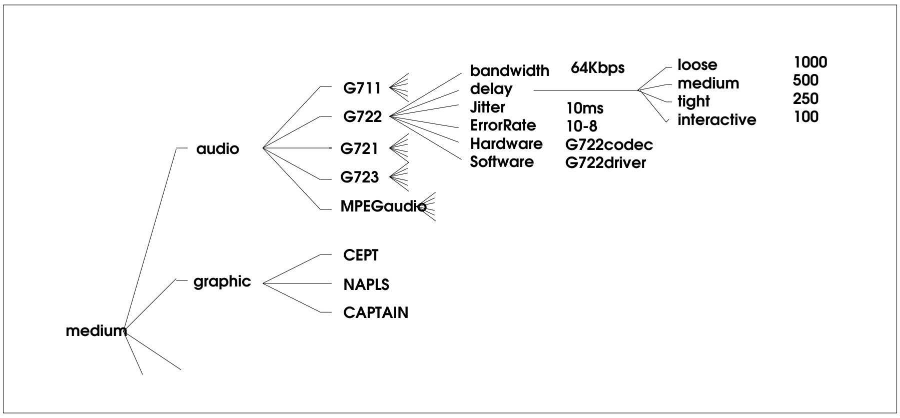


Figure 2: management information base

This entry provides the QoS manager with communication profile of this stream: its bandwidth, the intra-stream delay according to the expected quality (specified in the Q.Request), the jitter, the error rate and the hardware and software required on the system. The information contained in this database, is stable. The database can be updated, taking into account the evolution of coding or compression types. These databases do not contain information about the system on which its is installed and can be easily moved from one site to another.

**MIB for the monitoring of system and network resources**

The third type of MIB is used to monitor the system and network resources of the system. The amount of resources required for a stream depends on the characteristics of the computer and the underlying network. Each system has its own characteristics, but resources to be managed are identical on every computer. System and network MIBs reflect the use of these resources. These databases are managed by the Performance Monitor which is in fact an SNMP agent. This agent can provide statistics on the use of resources, but also is responsible for sending the indication messages, when necessary.

### 3.3 A need of specification for Applications

The model presented here, enforces the notion of adaptive applications, and the necessity for application programmers to take part in the QoS management by providing some renegotiation scripts and interacting with the QoS manager to reserve resources for these applications. It also shows how the specification of abstractions can help at different phase of the QoS management. Nevertheless, It is not obvious that every system and network will have capabilities of reserving resources. In such a context it is necessary to provide applications with the best quality given the currently available resources.

## 4 Adaptive Applications

Adaptive applications basically change their execution pattern to suit the operating environment. To enable adpativity it is necessary to build in the infrastructure to detect the changes in the operating environment into the application. This is necessary because to provide system level support the system will need to know every applications needs. The video play back and video editor example provided in [6] illustrates this clearly. The main requirement of the video player is to keep good synchronization between voice and video, while the video editor expects precise editing and therefore needs to be guaranteed that every frame is to be seen by users. The expected degradation scheme should then be different depending on the application. The conclusion is that a single degradation policy can not satisfy the requirements of diverse applications.

Recently two systems that utilize system level adpativity have been reported in the literature [20, 21]. IVS is a video conferencing system that uses feedback from the receiver to adjust the transmission of video frames. They have shown, by using a scheme similar to slow start, that it is possible to make it adapt to congestion in the network. However, its has been shown [23], that if there is contention for end system resources the feedback based scheme misbehaves.

The Vu system uses a receiver based scheme to adjust the operation of applications contending for end system resources [21]. Again, it has been shown that the scheme works well for single invocations, but fails when more than one application is running concurrently.

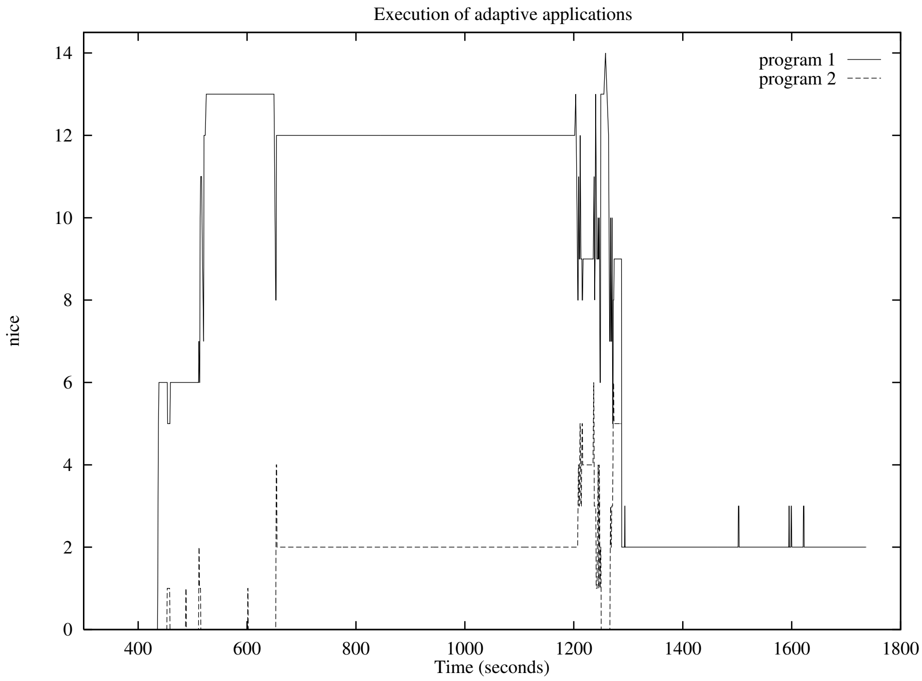


Figure 3: 2 Adaptive applications running concurrently

These results show that firstly, the adaptive application need to take into account the total resource requirements of the application. Secondly, applications need to adjust the QoS levels in a collaborative fashion. These two requirements imply that the QoS management should be done in a centralized fashion. However, the best form of adaptation for a given condition can only be determined by the application. Therefore we propose a hybrid scheme in which the monitoring of system resources is done external to the application, and the adaptation is done within the application. The following section show the results of an experiment we conducted to investigate the viability of this split architecture.

### 4.1 Application Based Adpativity

In this type of scheme, as mentioned earlier, applications adapt their quality according to the system load. So when the load increases, applications reduce their resources utilization and if the load of the system decreases they increase their quality. The aim of the simulations made here is to study the behavior of several applications running concurrently and to determine if the approach of using adaptive applications without global management can address the issue of resource allocation for multimedia applications.

Firstly we have implemented a simple adaptive application that modifies its priority according to the system load. This program executes several times the same function and measures the interval of time for each execution of this function. It compares the *i*-th execution of the function ($E$i) with the *i-1*th, ($E$i-1). As other programs can run concurrently to ours on the system, the execution time of the function will vary. If another program uses more resources, the execution time of our function increases $E$i $>$ $E$i-1. If this occurs, the model measures the decrease in available system resources and reduces its priority thus in effect lowering its QoS. Priorities fall in the range between 0 for the highest and 14 the lowest and are set with the nice Unix command. We have considered execution time modifications as significant if they vary by more than 20 percent.

The Figure 3 shows the behavior when we concurrently run two of these adaptive applications. The graph shows that even if both applications are started at the same time and in the same environment, one of the applications has a priority globally higher than the other (lower value of the nice parameter).

In this simulation, we found also that applications do not degrade their priority uniformally. This shows that the behavior of the adaptive application model is not deterministic. Thus, without global quality-based management, you run the risk of one application lowering its quality while the other other benefits from these freed resources unfairly. In order to demonstrate whether there is a need of a quality based manager associated with adaptive applications, we have implemented a supervisor that can be used with the above adaptive applications.

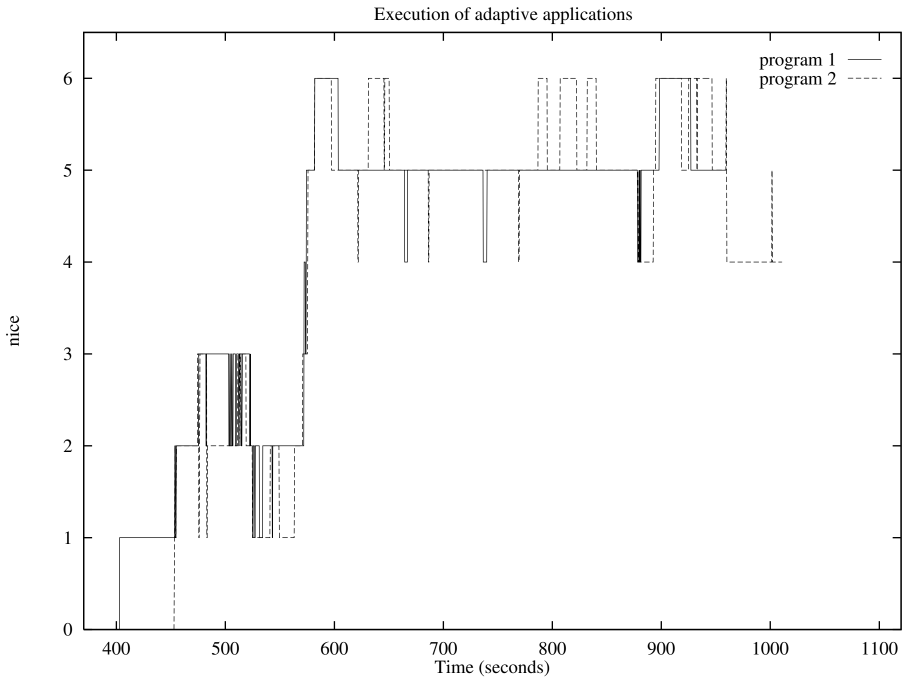


Figure 4: Role of the AQ Supervisor

## 4.2 Supervisor for the Quality Management of Adaptive Applications

As explained previously, the concept of adaptive applications seems well suited to multimedia environments as these applications are very demanding in terms of resources. In such environments, where traditional applications are also executed, it seems normal to share resources among diverse applications and run the demanding applications in a degraded mode where appropriate. Nevertheless, as shown in the previous section, adaptive applications are not capable of managing their resources in isolation. There needs to be some sort of global control to prevent applications from taking blind decisions. This can be performed by a *Applications Quality (AQ) Supervisor*. The role of this supervisor is to make applications converge to the same execution quality.

Communication between the adaptive applications and the *AQ supervisor* is done using messages. The types of messages used are:

- *INITIALIZATION* from the application to the server,
- *MODIFICATION* to specify either a AQ modification request, or
- *TERMINATION* to notify when an application has finished its execution.

The *receiver* and *sender* are the pid of the adaptive application or the identifier of the *AQ Supervisor* depending on which one is sending the message. It also contains the quality level at which the application is started for the opening of the connection or the expected quality level for a modification type message.

```
struct  message  {
        long     receiver;
        long     sender;
        integer  message_type;
        integer  quality;
}
```

The supervisor, keeps track of the quality level of each current application and determines which one is allowed to be degraded or upgraded. Only applications with the lowest quality can be improved and only the one with the highest quality can be degraded. Results of simulations run on the same Unix system with the *AQ Supervisor* can be seen in Figure 4.

Comparing the execution times between cases with or without a supervisor, Figures 3 and 4, shows that the *AQ supervisor* gives a much more predictable execution time. Both applications settle at the same priority level

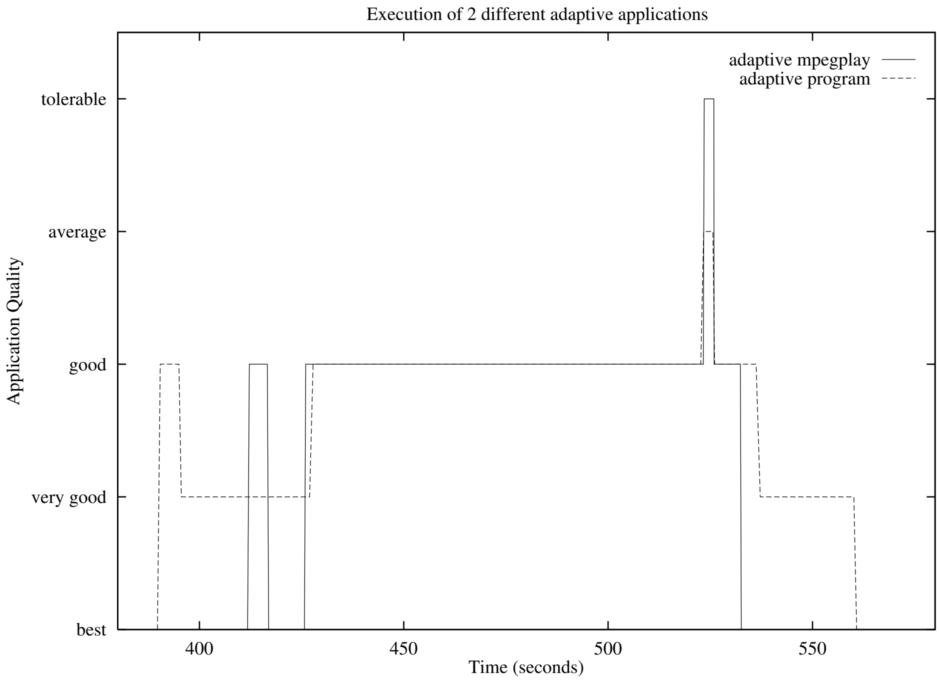


Figure 5: Role of the AQ Supervisor with several types of adaptive applications

and have very similar execution times. So even if the execution is still not deterministic, there is a convergence in the priority plots for two or more programs running concurrently.

Nevertheless, for other adaptive applications the quality level can refer to some parameters different than the application priority. It can be a video quality, a synchronization degree between voice and video, etc. Each application programmer can determine the quality levels associated with its application.

At this step of our experiments, we have then modified the *AQ supervisor* interface to specify a range of quality levels that can refer to something different than applications priority. A range of five quality levels (*best, very good, good, average, tolerable*) has then been made available for applications programmers. Each quality level can be associated with some characteristics of an application. Thus, for an adaptive application running only video, the *average* quality level can be associated with a black-and-white video, and the *good* quality level to a colour video; whereas a different adaptive audio and video application would associate the *average* quality level to a level of synchronization between the two media and the *good* quality level to another level of synchronization.

In order to validate the behaviour of our *AQ Supervisor* with several types of adaptive applications, we have implemented another adaptive application different than the previous one. For our first adaptive application we have assimilated the application priority to the quality level defined previously. So we have reduced to 5 priority levels to match with the *Applications Quality Supervisor* interface. Thus, the value 0 of the nice parameter corresponds to the highest quality level and 4 to the lowest one.

The second application is based on mpeg_play [22]. It modifies the display of video to adapt to the system load. With such a model, we intend to keep a minimum frame rate. Adaptive mpeg_play starts with a colour double-sized video, which is the *best* quality level of this application. If a minimum frame rate (5 frames/sec) can not be provided the quality is then degraded to *good* quality level (coloured normal-sized video) or to a *tolerable* level one (black-and-white normal sized) if there is not enough available resources to obtain the minimum frame rate. Thus, we have defined three quality levels for this application.

Results of these simulation are also very good as shown in Figures 5. It allows us to be very optimistic for development of new adaptive applications. These adaptive applications are running locally on the system, but it is easy to imagine some remote adaptive applications such as IVS, running concurrently with the local one. The AQ Simulator should then provide some similar results.

# 5 Conclusion

In this article, we provide a methodology for the development of multimedia applications. These applications usually have an interactive and dynamic nature. Moreover they handle continuous media and therefore they should be given the guarantee that a minimum quality will be provided to the application.

In some environments where resource reservation can be done, such as ATM networks, it is necessary for applications to specify their needs. The model given in this paper shows how this specification can be achieved with the use of the ASN.1 language and the creation of appropriate MIBs. The given model, also provides a methodology for developing multimedia applications which can be easily adapted to different host systems with a minimum of modification.

When no resources reservation can be done, a good alternative consists in giving applications the possibility of modifying their quality to make the best use of the available resources. Thus, applications programmers can determine which scheme should be adopted when the application quality decreases or increases. With such an approach it is not always necessary for applications to be aware of which resources are available. As these applications often use continuous media, a measure of the execution time of the display of pictures, or of a synchronization between several media can provide enough information to evaluate the quality. Our observations shows that the absence of a global *AQ supervisor* for adaptive applications makes the environment even more unstable. It becomes more difficult to predict the load of the system and a fair repartitioning of the resources between applications will not occur. The *AQ supervisor* gives some good results for the management of different adaptive applications.

The schemes developed here need to be combined in a common architecture to provide a complete management scheme for multimedia applications.

## References


[1] A. Campbell, G. Coulson, D. Hutchinson "A Quality of Service Architecture", ACM Computer Communication Review, Vol. 24, No. 2, April 1994.

[2] A. Seneviratne, M. Fry, V. Withana, V. Saparamadu, A. Richards, and E. Horlait, "Quality of service management for distributed multi media applications", IEEE Communication Society Phoenix Conference on Computers and Communications, (Phoenix-Arizona) 1994.

[3] L. Besse, L. Dairaine, L. Fedaoui, W.Tawbi, K. Thai, "Towards an Architecture for Distributed Multimedia Applications Support", International Conference on Multimedia Computing Systems, Boston, May 1994.

[4] S. Deering, "Reservations or No Reservations ?", INFOCOM'95 Panel Session, Boston, MA, April 1995.

[5] N. Davies, G. S. Blair, K. Cheverst, and B. Friday, "Supporting Adaptive Services in a Heterogeneous Mobile Environment".

[6] B. D. Noble, M. Price, and M. Satyanarayanan, "A Programming Interface for Application-Aware Adaptation in Mobile Computing", 2nd USENIX Symposium on Mobile and Location Independent Computing, Michigan, April 95.

[7] D. Ferrari, "To Share Rather Than to Pay", INFOCOM'95 Panel Discussion, April, 1995.

[8] C. Huitema, "Application Adaptivity", INFOCOM'95 Panel Session, Boston, MA, April 1995.

[9] L. Fedaoui, A. Seneviratne, A. Fladenmuller, and E. Horlait, "Implementation of a End-to-End Management Scheme",

[10] C.Vogt, G. v. Bochmann, R. Dssouli, J. Gecsei, A. Hafid, and B. Kerherve, "On QOS Negociation in Distributed Multimedia Applications", Technical Report, University of Montreal, Canada, 1993.

[11] R. Gopalakrishna, and R. Parulka, "Efficient Quality of Service in Multimedia Computer Operating Systems", Department of Computer science, Washington University, Report WUCS-TM-94-04, August 1994.

[12] A.Richards, "A Universal Transport Service: An adaptive End-to-End Protocol System", PhD Thesis, University of Technology, Sydney (submitted).

[13] A. Campbell, G. Coulson, F. Garcia, D. Hutchinson, and H. Leopold, "Integrated Quality of Service for Multimedia Communications", IEEE INFOCOM'93, 1993.

[14] M. Zitterbart, B. Stiller, and A. Tantawi, "A Model for High Flexible Performance Communication Subsystems", IEEE Journal on selected Areas in Communications, May 1993.

[15] D. Steedman, "ASN.1 The Tutorial and Reference", Technology appraisals, 1993.

[16] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "RFC 1442 Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", April 1993.

[17] M. Rose, "RFC 1418 SNMP over OSI", March 1993.

[18] P. Floressi, and Y. Yemani, "A Quality of Service Assurance Language", Columbia University Technical Report, 1993.

[19] M. Guglielmo, G. Modena, and R. Montagna, "Speech and Image Coding for Digital Communications", ETT, vol 2, no 1, February 1991.

[20] J-C Bolot, T. Turletti, "A rate control mechanism for packet video in the Internet", IEEE INFOCOM'94, June, 1994.

[21] D. Compton and D. Tenenhouse, "Collaborative Load Shedding", ICMCS, Boston 1995.

[22] K. Patel, B.C. Smith, and L. A. Rowe, "Performance of a Software MPEG Video Coder" Computer Science Division, University of California, Berkeley, http://www.cs.berkeley.edu.

[23] A. Seneviratne and H. S. Cho, "Quality of Service Mapping in Distributed Multimedia Systems" IEEE MMNet'95, Aizu, Japan, Septemebr 1995.